KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
VERSLO INFORMATIKOS KATEDRA

Mantas Vaitkus

Miesto viešo transporto keleivių srautų ir bilietavimo sistemos tyrimo programinė įranga

Magistro darbas

Vadovas
doc. dr. V. Pilkauskas

KAUNAS, 2005

TURINYS

Lentelių sąrašas

Paveikslėlių sąrašas

1. ĮVADAS

1.1. Dokumento paskirtis

Dokumento tikslas – apibrėžti magistro darbo problemą, pateikti galimus sprendimus ir juos išanalizuoti, pasirinkti sprendimą, pagal pasirinktą sprendimą atlikti eksperimentą: pateikti sistemos projektą, tyrimą ir rezultatus.

Skaitytojo patogumui dokumentas išskaidytas į skyrius. Šio dokumento antrajame skyriuje analizuojamos programinės įrangos kūrimo valdymo priemonės, kokybės užtikrinimo būdai, analogiškos pasaulyje naudojamos sistemos. Trečiame skyriuje pateikiami pagrindiniai projekto aspektai: projektavimo metu sukurti modeliai bei jų aprašymai. Ketvirtajame skyriuje išdėstyti kokybės analizės rezultatais ir rekomendacijos sistemos patobulinimui. Penktame skyriuje nagrinėjama sistemos eksperimentinė analizė, tiriama sistemos įtaka vartotojo darbo našumui bei pateikiami analizės rezultatai.

1.2. Santrauka

Šiuolaikinių technologijų progresas įpareigoja ieškoti efektyviausių sprendimų optimizuojant įmonių/organizacijų veiklos planavimo procesą. Automatizuoti įmonės ar organizacijos procesų, svarbių įvykių planavimas ir kontrolė, išteklių apskaitos planai gerina įmonės darbo kokybę, mažina valdymo kaštus. Ne išimtis ir viešo transporto sektorius. Kuriant programinę įranga MunLT išanalizuotos įvairios, pasaulyje naudojamos panašios paskirties programinės įrangos modifikacijos. Palyginti pagrindinai jų privalumai ir trūkumai.

Šis darbas skirtas keleivių srautų ir bilietavimo sistemos tyrimo programinės įrangos tobulinimui naudojant naujausias programinės įrangos kūrimo technologijas ir projektavimo įrankius.

Darbe nagrinėjamos programinės įrangos kūrimo planavimo priemonės, leidžiančios sumažinti riziką ir užtikrinti produkto kokybę. Sistemos architektūros kūrimas ir dokumentavimas atliktas pagal keliamus programinės įrangos kūrimui reikalavimus [1]. Atliktas sistemos kokybės testavimas naudojant esminius metodus testinių atvejų identifikavimui: funkcinis ir struktūrinis testavimas [2]. Gautas produktas atitinkantis vartotojo keliamus reikalavimus Naudojamos technologijos užtikrino trumpą sistemos sukūrimo laiką, sumažintą savikainą ir investicijas pakartotiniam panaudojamumui.

Raktiniai žodžiai. Viešas transportas, keleivių srautai, bilietavimo sistema, programų kūrimas.

Summary

The progress of technologies obliges us to look for the most effective solutions to optimize design and development process of enterprise. Automated process and crucial events handling makes enterprise's work quality better and reduces management costs. There is no exception in the segment of public transport too. A similar purpose software modification was analyzed during MunLt development process. Main advantages and disadvantages were compared.

This thesis is concerned with use of the newest software development technologies ant tools in software development of passengers' flows and ticketing system research.

Tools for software management and planning were analyzed in this work. They let us to reduce risk and guaranty software quality. Development of the system architecture fulfills all requirements for software documenting. Software quality tested using structural and functional test cases. The final product satisfies all users' requirements. Faster time-to-market, reduced costs for investment or reuse are reason why we use new software development tools.

Keywords. Public transport, passengers flows, ticketing system, software development.

2. PROBLEMŲ SPRENDIMO BŪDŲ ANALIZĖ

2.1. Problemos aprašymas

Viešasis keleivinis transportas – svarbi vietinio ūkio sritis. Keleivinis transportas – svarbi socialinė paslauga be kurios sunkiai suvokiamas miesto gyvenimas. Patogus, ekonomiškas ir saugus žmonių judėjimas iš vienos vietos į kitą yra transporto politikos kūrimo pagrindas. Svarbu yra integruota pozicija, kuri apima strategiją, siekiančią padidinti naudojimąsi viešuoju transportu. Viešojo transporto problemos yra nagrinėtos įvairiais pjūviais, bet dažnai tai būdavo užsakymai iš įmonių, kad gauti atsakymus į konkrečius klausimus. Dažniausiai būdavo ieškoma atsakymų, kaip pateikti geriausią paslaugą su minimaliomis biudžetinėmis lėšomis. Pastarąjį dešimtmetį mieste daug kas keitėsi. Pasikeitė žmonių traukos centrai, išnyko didžiulės pramonės įmonės, kurių poreikiams buvo suprojektuoti viešojo transporto maršrutai. Natūralu, kad specialistai ieško kelių, kaip pritaikyti šį svarbų kompleksą šiuolaikiniam miestiečiui. Siekiant tinkama kryptimi vystyti viešąjį transportą ir plėtoti jo maršrutų tinklą būtina remtis detaliais tyrimais. Viešojo transporto planavimas ir organizavimas vykdomas remiantis visuomenės poreikių viešajam transportui detaliais tyrimais. Pagrindinės tyrimų kryptis apima: keleivių srautų, bilietavimo sistemos, transporto talpos panaudojimo, keleivių kaitos koeficiento, keleivio vidutinio kelionės ilgio, kelionės trukmės [3].

Nenaudojant šiuolaikinių priemonių, viešojo transporto tyrimams reikalingus duomenis galima surinkti naudojant vizualinius, anketinius duomenų surinkimo būdus. Tokie duomenų surinkimo metodai užima daug laiko, nėra tikslūs, priklauso nuo duomenų surinkėjo sąžiningumo.

Naudojant šiuolaikiškas priemones, šį procesą galima automatizuoti ir priversti patį vartotoją dalyvauti tyrimo duomenų rinkimui. Tereikia įvesti elektroninį bilietą viešajame transporte.

Pasaulyje duomenų, surinktų iš viešo transporto, apdorojimui ir analizavimui yra sukurta pakankamai nedaug programinės įrangos. Taip yra todėl, jos tai yra specializuota, nepaprastai brangi ir kuriama dažniausiai pagal individualius užsakymus, programinė įranga. Kadangi tokios programinės įrangos atsinaujinimas yra brangus ir retas dalykas, jos palaikymas, funkciniai pakeitimai brangiai kainuoja.

Naudojat naujausias technologijas, galima vartotojui pateikti kokybišką ir prieinama kaina produktą. Kuriant programinę įrangą, jos kodavimas užima tik mažą dalį darbo, kurį reikia padaryti iki programinė įranga pateikiama vartotojui.

Pasaulyje, siekiant valdyti programinės įrangos kūrimo procesus, kuriami nauji arba tobulinami egzistuojantys modeliai. Projekto valdymo metodai, taikomi programinės įrangos projektų valdymui, tačiau PĮ turi tik specifinių, tik jai būdingų savybių:

- Produktas yra nematerialus. Neįmanoma pažiūrėti kiek produktas išbaigtas. Projekto vadybininkas, sekdamas projekto vykdymą gali remtis tik dokumentacija.

- Nėra aiškaus supratimo apie PĮ kūrimo procesą. PĮ kūrimo procesas tik dalinai standartizuotas.
- Didelės PĮ sistemos dažnai yra unikalūs projektai.
- Lankstumas, Programos gali būti lanksčiai pritaikytos, modifikuotos, t.y. turi aukščiausią keitimų laisvę.

Yra esminis skirtumas tarp mažos, asmeniniam naudojimui skirtos programos rašymo ir PĮ produkto kūrimo ir modifikavimo. Nėra nei analizės, nei projektavimo nei testavimo planavimo, o tik kodavimas.

PĮ projektavimas yra žymiai platesnė sąvoka, apimanti analizę, projektavimą, kodavimą ir testavimą. Bendriausi programinės įrangos inžinerijos proceso modeliai yra:

- Krioklio modelis – aiškiai atskirtos proceso veiklos.
- Evoliucinis modelis – proceso veiklos persidengia.
- Formalusis modelis – matematinis sistemos modelis aprašytas formaliomis specifikacijomis transformuojamas į realizaciją.
- Pakartotinio panaudojimo modelis – sistema komplektuojama iš jau egzistuojančių komponentų.

Projektuotojas privalo įvertinti projektuojamos sistemos sudėtingumą, reikalavimų aiškumą, pritaikymo sritį ir parinktį optimaliausią kuriamai sistemai modelį. Projekto valdymas – integruotas procesas. Veiksmai reikalingi vieniems darbams dažniausiai turi įtakos kitiems darbams Projekto valdymo sistema susideda iš valdymo objekto (projektas) ir valdymo subjekto (projekto komandos). Projektų valdymo sistemoje realizuojamos dvi procesų grupės:

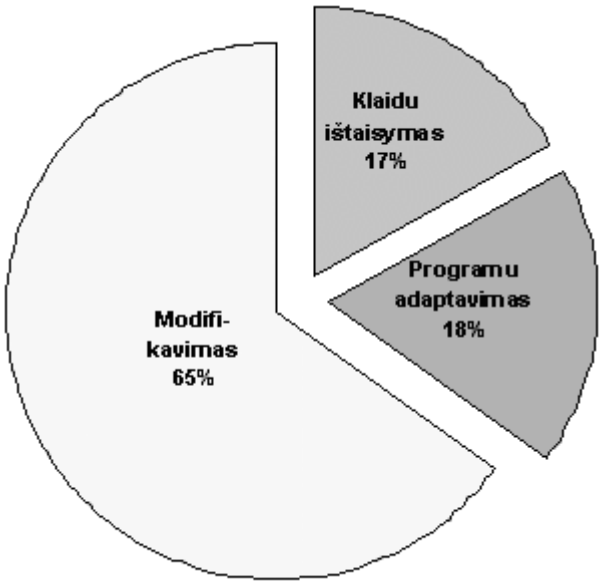
- Projektavimo procesai, susiję su projekto valdymo objektu. Šiuos procesus realizuoja projekto darbų vykdytojai.
- Projekto valdymo procesai, susiję su projekto valdymo subjektu arba projekto vykdymo komanda ir apima planavimo, organizavimo ir koordinavimo darbus.

Projekto valdymo procesus galima suskirstyti į šias grupes:

- Inicijavimo procesai – idėjos generavimas ir atranka, projekto pradžios formalus pripažinimas.
- Planavimo procesai – projekto plano, kuris leistų pasiekti projekto tikslus, parengimas.
- Vykdymo procesai – projekto įgyvendinimo eigos stebėjimas, vertinimas.
- Analizės procesai – projekto įgyvendinimo eigos stebėjimas, vertinimas.
- Valdymo procesai – projekto koreguojančių veiklų nustatymas, vykdymas, nepalankių veiklų pašalinimas.
- Užbaigimo procesai – formalus projekto užbaigimas, rezultatų įvertinimas.

Projekto valdymo procesai gali sekti vienas paskui kitą, persidengti, gali vykti skirtingu intensyvumu. Be to projekto valdymo procesai yra susieti vienas su kitu savo rezultatais, t.y. vieno proceso rezultatai tampa pagrindiniais duomenimis kitam procesui.

Svarbus parametras, užtikrinantis programos gyvavimą ir sėkmingą integraciją bei konkurencingumą yra programos priežiūros užtikrinimas. Programos priežiūros kaštai palaipsniui auga ir nuo 20% 1970 m. išaugo iki 70% programos gyvavimo ciklų kaštų [4]. Todėl vien JAV programų priežiūrai ir palaikymui išleidžiama 43 milijardai dolerių [5]. Pagal IEEE-STD 1219 standartą [6], programos priežiūra tai programinio produkto modifikavimas po pristatymo siekiant ištaisyti klaidas, pagerinti charakteristikas arba pritaikyti produktą prie pasikeitusios aplinkos.



2.1 pav. Priežiūros pastangų pasiskirstymas

2.2. Problemos apibrėžimas ir motyvacija

Transporto keleivių srautų ir bilietavimo sistemos tyrimo programinė įranga nėra naujiena programinės įrangos produktų rinkoje. Nors kompanijų, siūlančių savo produktus, nėra daug, panagrinėsime keletą naudojamų programinės įrangos produktų:

2.2.1. Programinė įranga „MUNICOM“

- ✓ Veikia net ant 486 DX4/100 asmeniniu kompiuteriu (AK) su 16MB atminties (RAM), operacinė sistema (OS) – Microsoft Windows 95 ir aukštesnė.
- ✓ Analizuojamų duomenų filtravimo galimybė.
- ✓ Galimybė pažiūrėti analizuojamų duomenų periodo ribas.

- ✓ Pažymėtų bilietų ataskaitos įvairiais pjūviais.
- ✓ Parduotų bilietų ataskaitos įvairias pjūviais.
- ✓ Keleivių srautų ataskaitos įvairias pjūviais.
- ✓ Sistemos tam tikrais komponentais gali naudotis tik vartotojai, kuriems suteiktos teisės jais naudotis.
- ✓ Sistemos vartotojų valdymo modulis [7] .

Trūkumai:

- ✓ Sistema realizuota tik DOS aplinkoje.
- ✓ Nėra realizuoti vartotojo priėjimo lygiai.
- ✓ Neteikiamos jokios išvados iš analizuojamų duomenų.
- ✓ Tik angliška ir lenkiška vartotojo sąsaja.
- ✓ Nėra sistemos palaikymo.

2.2.2. Programinė įranga „PIKAS“

- ✓ Patogus valdymas.
- ✓ Programa yra lietuvių kalba.
- ✓ Paprasta vartotojo sąsaja.
- ✓ Sistemos palaikymas.
- ✓ Pagal surinktus duomenis automatiškai generuojamas transporto priemonių važiavimo grafikas.
- ✓ Nurodomas optimalus transporto priemonių skaičius maršrute [8].

Trūkumai:

- ✓ Nėra realizuoti vartotojo priėjimo lygiai;
- ✓ Nėra duomenų filtravimo galimybės.
- ✓ Nėra galimybės analizuoti keleivių srautus;
- ✓ Nėra galimybės analizuoti panaudojamas bilietų rūšis.

2.2.3. Programinė įranga „PTV INTERPLAN“

- ✓ Analizuojamų duomenų filtravimo galimybė.
- ✓ Patogus valdymas.
- ✓ Automatinis išvažiavimo laikų sudarymas;
- ✓ Keleivių srautų ataskaitos įvairiais pjūviais [9].

Trūkumai:

- ✓ Nėra galimybės analizuoti panaudojamas bilietų rūšis.
- ✓ Nėra lietuviškos vartotojo sąsajos.
- ✓ Nėra realizuoti vartotojo priėjimo lygiai;

- ✓ Reikalauja didelių kompiuterio resursų.

Atliekant šių produktų analizę, matoma nemažai trūkumų, susijusių su sistemos pakartotiniu panaudojamumu, darbo efektyvumu, greito funkcionalumo papildymo stoka. Todėl atsiranda poreikis apjungti šias sistemas panaudojant naujausias technologijas, tuo pačiu siekiant orientuoti produktą į šiuolaikinį programinės įrangos vartotoją. Vienas iš galimų sprendimų – naudoti kliento/serverio technologiją.

Sistema nespręsta realizuoti komponentais, kurie atlieka tam tikrą užduotį. Bendras sistemos funkcionalumas sudaromas susiejant šiuos komponentus atitinkamais ryšiais – naudojama į paslaugą orientuota architektūra.

Sistema, atsižvelgiant į klientų naudojamas OS, realizuojama Windows XP platformai. Kliento/serverio architektūrai realizuoti bei darbui su vietinėmis ir nutolusiomis DB, naudojamas mechanizmas, vadinamas Borland duomenų bazės varikliu (BDE - Borland Database Engine). Priėjimui prie nutolusios DB, pvz. SQL, BDE, naudojant SQL valdiklį (ODBC), jungiamasi prie SQL bazės serverio.

Analizuojant panašias sistemas, padaryta išvada, kad dauguma šių sistemų turi nepakankamą palaikomumą. Šiuolaikinė programinė įranga privalo turėti palaikomumą ir tęstinumą. Kitu atveju ji „pasmerktą“ būti nekonkurencinga preke rinkoje. Todėl projektuojant kuriamą sistemą didelis dėmesys skiriamas sistemos palaikomumui. Programinės įrangos kūrimo procese vykdomas griežtas sistemos kūrimo proceso dokumentavimas, kas ateityje leis sumažinti programinės įrangos palaikymo ir atnaujinimo kaštus.

Atsižvelgiant į ribotą programinės įrangos potencialių klientų kiekį, nuspręsta atsisakyti automatinio (savaiminio) programinės įrangos atsinaujinimo būdo, taip sumažinant programinės įrangos kainą vartotojo kainą ir kūrimo kaštus.

2.3. Darbo tikslas

Sukurti viešo transporto keleivių srautų ir bilietavimo sistemos tyrimo programinę įrangą panaudojant kliento/serverio architektūrą.

2.4. Sistemos realizavimo savybės

Pagal atliktą galimų sprendimų analizę, pasirenkamas sprendimas

1. Sistema kuriama naudojant krioklio modelį..
2. Eksperimente bus naudojamas objektiškai orientuotas programavimas.
3. Sistema bus realizuojama atsižvelgiant į:
   - ✓ Apkrovimo balansavimą.
   - ✓ Tinkamą reakcijos į sistemos klaidą.

- ✓ Duomenų saugumą.
- ✓ Patikimumą.
- ✓ Pritaikomumą siekiant užtikrinti greitą naujų vartotojo poreikių realizaciją.

4. Sistemoje bus realizuoti šie privalumai:
   - ✓ Administravimo įrankis - grafinis administravimo įrankis nutolusių vartotojų ir serverių darbo sekimui.
   - ✓ Integruota vystymo aplinka.
   - ✓ Sujungimų su DB "baseino" technologija ir sujungimų laikinu saugojimu atmintyje.
   - ✓ Verslo ir taikomosios programos logikos sąsaja.

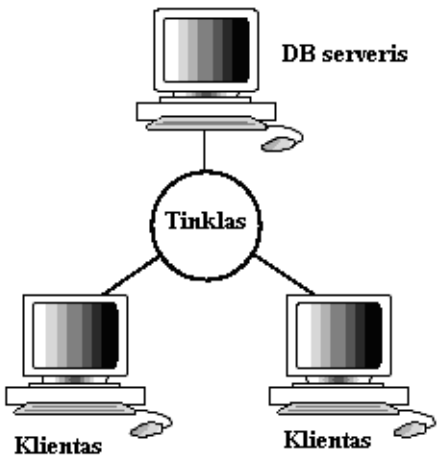
2.5. Darbo naujumas

Programinės įrangos projektavimas ir realizacija panaudojant naujausias programinės įrangos kūrimo ir projektavimo technologijas bei būdus.

2.6. Pasirenkamų metodų apžvalga

Kliento/serverio architektūros aprašymas

Terminas klienas/serveris pirmą kartą buvo panaudotas 1980 m. informacijoje asmeniniams kompiuteriams tinkle. Dabartinis kliento/serverio modelis priimtas 1980 m. pabaigoje. Kliento/serverio programos architektūra yra įvairiapusė, leidžianti pagerinti programos panaudojamumą, pernešamumą. Klientas atlieka paslaugų užklausų siuntimo paskirtį, o serveris laikomas kaip paslaugų teikėjas. Viena sistema gali būti ir klientas ir serveris, priklausomai nuo sistemos konfigūracijos.



2.2 pav. Kliento/serverio architektūros paveikslas

Kliento programa neatlieka jokių duomenų apdorojimo. Ji tik laukia vartotojo paliepimo pareikalauti duomenų iš serverio. ir tada analizuoti ir atvaizduoti duomenis naudojant atvaizdavimo savybėmis.

Kliento programa nepriklauso nuo fizinės duomenų vietos. Netgi jei duomenys perkeliami iš vienos vietos į kitą (pakeičiamas serveris), programa vis tiek gali funkcionuoti [10].

Apibendrintas projektavimo metodas (RUP)

Programinės įrangos kūrimui pasirinkta RUP technologija dėl to kad tai panaudojamų atvejų valdomas, architektūra grindžiamas, iteracinis, vykdomas palaipsniui, riziką mažinantis projektavimo procesas. RUP užtikrina programos kūrimo įgyvendinimą, su standartine pritaikoma proceso aplinka. Kad sumažinti riziką, kūrimas atliekamas iteracijos būdu. Kiekviena iteracija baigiasi veikiančiu produktu. Kas tai yra iteracinis vystymas? Projektas kuriamas naudojant iteracijas, susideda iš kelių iteracijų gyvavimo ciklų. Iteracija apjungia laisvai išplaukiantys veiksmų aibės modeliuojant verslo taisykles, reikalavimus, analizė ir projektavimą, realizaciją, testavimą ir įdiegimą skirtingomis proporcijomis, priklausomai nuo to, kokiame gyvavimo cikle randasi iteracija [11].

Kodėl iteracinis projekto vystymas yra toks patrauklus? Žemiau išvardinti iteracinio projekto kūrimo metodo privalumai:

- Ankščiau įvertinama rizika, kadangi sistemos elementai integruojami palaipsniui.
- Funkcinių ir nefunkcinių reikalavimų pasikeitimai netampa kritiniais.
- Palengvintas produkto gerinimas ir tobulinimas, kas leidžia sukurti geresnį produktą.
- Proceso vykdytojai gali reaguoti į pasikeitimus ankstesniuose procesuose ir patobulinti savo procesą.
- Padidintas panaudojamums.

Kaip programos procese sumažinama rizika naudojant iteracinį projektavimo proceso metodą? Iteracinis metodas leidžia sumažinti riziką anksčiau, kadangi daug rizikos yra nustatoma įdiegimo metu. Einant per visus projekto proceso iteracijas, peržiūrimos visos sritys, tobulinami daugelis projekto aspektų: programos, žmogaus sugebėjimai ir t.t. Surasta rizika gali nepasiteisinti, o nauja nenumatyta rizika gali pasirodyti.

Vartotojai tobulėja kartu tobulėjant sistemai, labai svarbi savybė, leidžianti keisti funkcinius ir nefunkcinius reikalavimus sistemai.

Naudojant iteracinį metodą, sukuriama tobulesnė programinės įrangos architektūra, kadangi klaidos ištaisomos per kelias iteracijas. Be to sistema testuojama kiekvieno ciklo pabaigoje, kad užtikrina, kad sistema bus ištestuojama daug detaliau nei naudojant kitą programinės įrangos kūrimo metodą.

Projektas gali būti patobulintas kūrimo metu. Įvertinant projekto kūrimo būseną iteracijos pabaigoje, planuojama projekto grafiko ateities perspektyva bei analizuojamas reikalingi pakeitimai proceso organizaciniais klausimais, kad užtikrinti geresnį produktyvumą sekančioje iteracijoje.

Daug lengviau identifikuoti pagrindines dalis kaip jos yra pusiau sukurtos arba įdiegtos. Sunku identifikuoti ir vystyti panaudojamas programos dalis. Proceso peržiūra praėjusiose iteracijose leidžia programos architektui identifikuoti nenumatytus, potencialius panaudojamus atvejus ir sekančiose iteracijose juos vystyti ir įvertinti. Iteracijos metodas leidžia lengviau integruoti kitų gamintojų integruojamas kodo dalis, bei jas integruoti ir įvertinti kuriamoje sistemoje.

Programinės įrangos projektavimo kalba (UML)

UML tai pramonės standartizuota projektavimo ženklų sistema, naudojama objektiškai orientuotoms sistemoms ir yra pirmaujanti platforma greitam programų kūrimui. UML sudaryta iš 4 modelių tipų [12]:

- ✓ Panaudojimo;
- ✓ Statinio;
- ✓ Elgsenos;
- ✓ Architektūros.

Kiekvienas modelis turi savo diagramos tipus. Panaudojimo modeliai turi panaudojimo atvejų diagramas. Statiniai modeliai turi klasių ir objektų diagramas. Elgsenos modeliai turi būsenos, veiklos, sekų ir bendradarbiavimo diagramas. Architektūriniai modeliai turi realizacijos, komponentų ir įdiegimo diagramas.

Panaudojimo atvejų diagramos

Panaudojimo atvejų diagramos vaizduoja aktorių ryšius su panaudojimo atvejais ir apibrėžia sistemos ribas. Daug informacijos gali būti surinkta žiūrint panaudojimo atvejų diagramą. Ji parodo sistemos funkcionalumą. Vartotojai, projekto vadovai, analitikai, testuotojai ir kiti suinteresuoti asmenys gali pažiūrėję į šias diagramas suprasti ką turi atlikti sukurta sistema.

Veiklos diagramos

Veiklos diagramos parodo sistemos veiklos procesus. Jos gali būti panaudojamos verslo taisyklių modeliavime, kad parodyti verslo procesus. Jos gali būti panaudotos reikalavimų rinkimo metu, kad pavaizduoti įvykių procesus esančius panaudojimo atvejuose. Šios diagramos parodo kur veiklos procesas prasideda, kur baigiasi, kokie veiksmai atliekami proceso pradžioje ir pabaigoje, kuris parodo proceso pabaigą. Veiklos diagramos nebūtinai kuriamos kiekvienam procesui, bet jos yra galingas bendravimo įrankis, ypač esant didelėms sistemoms su daug ir sudėtingais veiklos procesais.

Sekų diagramos

Sekų diagramos naudojamos pavaizduoti objektų sąveikas laike. Jų pagalba galima aprašyti sąveikas ne tik tarp klasių objektų, bet ir sistemų, posistemių ar programinių komponentų

Bendradarbiavimo diagramos

Bendradarbiavimo diagrama vaizduoja tą pačią informaciją, kaip ir sekų diagrama, tik kitu pjūviu. Jei sekų diagramoje matoma pranešimų eilės tvarka, tai bendradarbiavimo diagramoje išryškėja bendradarbiaujančių objektų architektūra, o pranešimų eilę rodo tik jų numeriai. Priklausomai nuo poreikių galima naudoti vieną ar kitą diagramos tipą. Testuotojai ir sistemos architektai naudoja šias diagramas, kad pastebėti procesų išsidėstymą tarp objektų.

Klasių diagramos

Klasių diagrama parodo sąveiką tarp sistemos klasių. Sistemos analitikai naudoja klasių diagramas parodyti sistemos detalų vaizdą. Taip pat klasių diagrama naudojama sistemos struktūros peržiūrai. Pažymėtina, kad netgi tame pačiame modelyje egzistuoja aktyvios ir pasyvios klasės, kurių ryšiai turi skirtingą prasmę. Aktyvios klasės gali aktyvuoti kitas klases, jų tarpusavio asociacijos ar asociacijos su pasyviomis klasėmis reiškia sąveikas. Pasyvios klasės pačios negali aktyvuoti kitų klasių, jų tarpusavio ryšiai atspindi dalykinės srities konceptų sąryšius.

Būsenų diagramos

Būsenų diagramos parodo įvairias modelio būsenas, kuriose gali egzistuoti objektas. Jei klasių diagrama parodo statinį klasių ir jų ryšių vaizdą, tai būsenų diagramos parodo sistemos dinaminį vaizdą ir elgseną. Būsenų diagramos nenaudojamos pavaizduoti kiekvieną klasę, jos vaizduoją klasių kompleksą. Jei klasės objektas gali egzistuoti keliose būsenose ir elgtis skirtingai kiekvienoje būsenoje, tam pavaizduoti naudojamos būsenos diagramos.

Komponentų diagramos

Komponentų diagramoje pavaizduojamas fizinis modelio vaizdas, t.y. sistemos komponentai ir ryšiai tarp jų. Yra dvi komponentų rūšys: vykdomi komponentai ir bibliotekos. Komponentų diagrama naudojama kompiliuojant sistemą. Ji parodo kokia seka reikia kompiliuoti. Be to ji parodo kokie vykdomi komponentai bus sukurti kompiliacijos metu.

Įdiegimo diagramos

Įdiegimo diagrama parodo fizinį tinklo sluoksnį ir kaip išdėstomi įvairūs komponentai.

Krioklio modelis

Krioklio modelis naudojamas kai vartotojo reikalavimai yra aiškūs, t.y. vartotojas arba programos užsakovas žino ko nori gauti iš programos. Tačiau programinės įrangos kūrimui naudojant krioklio modelį, susiduriama su sekančiais sunkumais [13]:

1. Nelankstus projekto skirstymas į stadijas.
2. Tai sukelia sunkumus, kai tenka taisyti klaidas arba kai tenka reaguoti į vartotojo reikalavimų pasikeitimą.

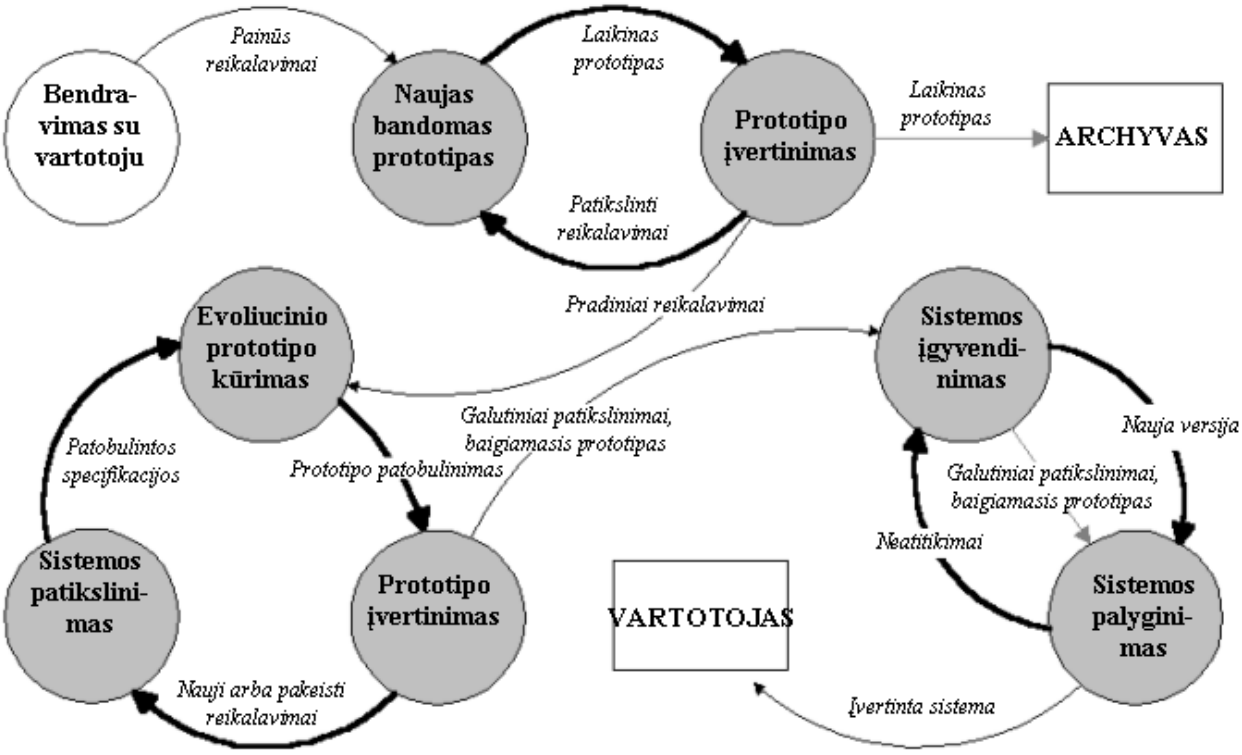
3. Vykstant procesui, sunku daryti pakeitimus, paprastai jie atliekami po paskutinės fazės, grįžtant į kurią nors fazę.

Evoliucinis modelis

Evoliuciniame modelyje projektas vystosi nuosekliai, pradedant panaudojimo atvejų poaibio realizavimu ir nuosekliai prijungiant naujus panaudojimo atvejus arba jų grupes. Modelio esmė – darbas su užsakovu nuo pradinės specifikacijos iki galutinio produkto pateikimo. Reikalavimų supratimas auga realizavimo eigoje [14].

Evoliucinio modelio trūkumai:

- „Suveltas procesas“.
- Programinės įrangos sistema gaunasi blogai struktūrizuota..
- Reikia “specialių” specialistų, sugebančių įkalbėti vartotoją dirbti su nepilnai suprojektuota programinės įrangos sistema.
- Labai svarbus yra išdirbėjo bendravimas su vartotoju, gera psichologinė aplinka.



2.3 pav. Evoliucinis modelis

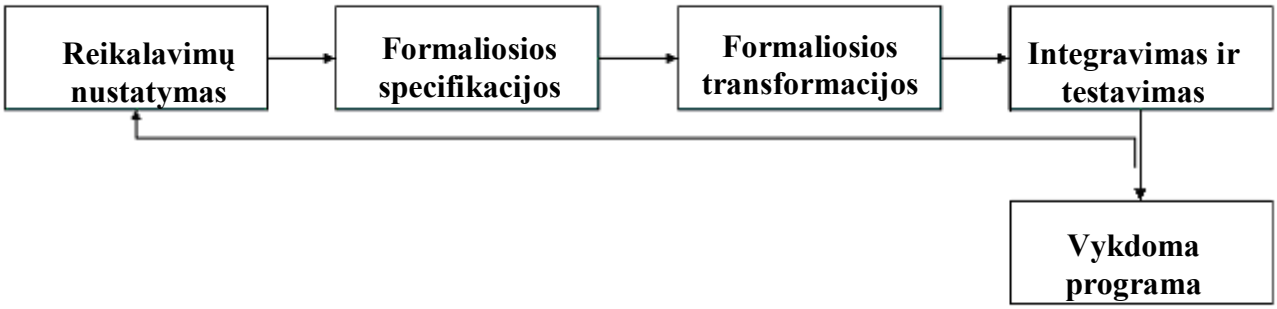
Formalusis metodas

PIS aprašoma matematiškai ir formalių metodų pagalba generuojamas galutinis jos variantas (vykdoma programa). Tokiu būdu galima sugeneruoti tik vykdomą programą, tačiau lieka parengti

dokumentaciją, pagalbos tekstus. Tinka nedidelėms sistemoms bei sistemoms, keliančioms aukštus reikalavimus patikimumui [15].

Trūkumai:

- Procesas labai formalus ir sudėtingas.
- Sunku formaliai aprašyti sistemas.
- Reikia aukštos kvalifikacijos specialistų.
- Nepavyksta visko gauti vien formalių transformavimų kelių, tenka įdėti ir rankinio darbo, ypač – projektuojant vartotojo sąsają.



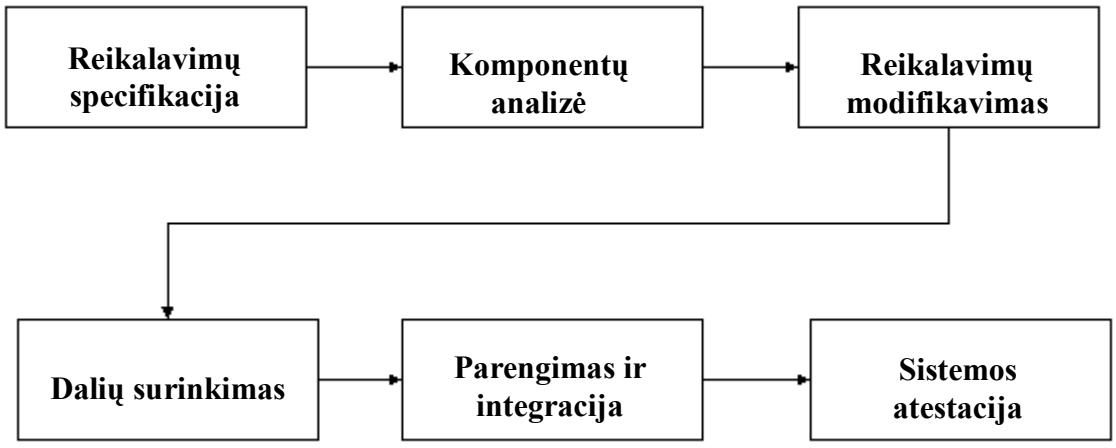
2.4 pav. Formalios specifikacijos modelis

Pakartotinio panaudojimo modelis

Modelio esmė – esamų sistemų, posistemių ar programų panaudojimas iškilusiems uždaviniams spręsti. Naujas uždavinys – į vieningą sistemą integruoti skirtingų gamintojų sukurtas dalis [16].

Trūkumai:

- Reikia gerai orientuotis produktų rinkoje.
- Reikalingi specialistai, gebantys aukščiausiu lygiu įsisavinti svetimus produktus, ne visuomet turint išeities kodus.
- Sudėtinga posistemių sąveika, ne visuomet efektyvios sąsajos tarp posistemių, skirtingi darbo su posistemėmis stiliai.



2.5 pav. Pakartotinio panaudojimo modelis

Atvira priėjimo prie duomenų bazių sąsaja (ODBC)

ODBC tai standartas, sukurtas firmos Microsoft, kad leisti bet kuriai programai bendrauti su duomenų bazės vidine programa. Be to ODBC sąsaja yra daugiaplatfomė ir veikia su kitomis operacinės sistemomis kaip Unix, Macintosh ir OS/2. Technologija leidžia valdyti dideles DB bei didelius užklausų skaičius. Standartas sukurtas bendradarbiaujant techninės įrangos, programinės įrangos ir tinklinių sprendimų pardavėjams. Jie nustatė pagrindinius priėjimo prie duomenų būdus kad supaprastinti kliento/serverio sistemos veikimą [17].

Microsoft panaudojo pardavėjų sudarytus priėjimo prie duomenų būdus ir sukūrė šaukinių lygio sąsają, vadinamą atvira priėjimo prie duomenų bazių sąsaja (ODBC). Dabar dauguma DB stočių pardavėjų ir tiekėjų tiekia produktus su integruota ODBC sąsaja, per kurią galutinis vartotojas gali prisijungti prie saugomų duomenų tiesiogiai iš nutolusios darbo vietos [18].

ODBC standartas numato tris prisitaikymo lygius taikomųjų programų sąsajai (API):

- ✓ Pagrindinis (23 funkciniai šaukiniai);
- ✓ 1 lygio (15 funkcinių šaukinių);
- ✓ 2 lygio (16 funkcinių šaukinių).

3. PROGRAMINĖS ĮRANGOS TECHNINĖ–PROJEKTINĖ DOKUMENTACIJA

3.1. Sistemos aprašymas

MunLt - tai programa skirta surinktų duomenų iš transporto priemonės analizei. Programa analizuoja duomenis įvairiais pjūviais ir pateikia rezultatus.

MunLt programos tikslas – palengvinti surinktų duomenų apdorojimą ir pateikti išvadas iš surinktų apdorojimui rezultatų.

Programos kompiuterizuojamos funkcijos:

- ✓ Keleivių srautų analizė;
- ✓ Optimaliausių kontrolės vietų parinkimas;
- ✓ Bilietavimo sistemos analizė;
- ✓ Optimalaus transporto skaičiaus maršrute skaičiavimas;
- ✓ Važiavimo grafikų automatizuotas sudarymas;

3.2. Sistemos vartotojai

Programine įranga gali naudotis kiekvienas, mokantis dirbti asmeniniu kompiuteriu. Tačiau darbas bus efektyvus, jei vartotojas turės bent minimalias žinias apie taikomąją sritį. Vartotojams būtinos lietuvių kalbos žinios. Programa yra skirta lietuviškai kalbančių vartotojų grupei, todėl visi paaiškinimai pateikiami būtent šia kalba.

Išskiriamos šios vartotojų grupės: administratorius, paprastas vartotojas.

3.2.1. Administratorius

Vartotojo sprendžiami uždaviniai:

- ✓ Užregistruoti naują vartotoją.
- ✓ Keisti užregistruotą vartotoją.
- ✓ Pašalinti užregistruotą vartotoją.
- ✓ Užregistruoti naują vartotojų grupę.
- ✓ Keisti vartotojų grupės priėjimo prie sistemos lygį.
- ✓ Pašalinti vartotojų grupę.
- ✓ Keisti sistemos nustatymus.

3.2.2. Paprastas vartotojas

Vartotojo sprendžiami uždaviniai:

- ✓ Keleivių srautų analizės ataskaitos generavimas.
- ✓ Optimaliausių kontrolės vietų ataskaitos generavimas.
- ✓ Pažymėtų bilietų pagal tipą ataskaitos generavimas.
- ✓ Optimalaus transporto skaičiaus maršrute ataskaitos generavimas.
- ✓ Važiavimo grafikų ataskaitos generavimas.

3.3. Produkto veiklos sfera

3.3.1. Sistemos funkciniai reikalavimai

1. Sistemoje turi būti numatyta vartotojų ir priėjimo prie sistemos lygių administravimo galimybė. Tam turi būti sistemoje numatytas vartotojas, turintis teisę keisti kitų sistemos vartotojų prisijungimo duomenis, sudaryti vartotojų grupes, keisti grupių priėjimo prie sistemos lygį.
2. Sistemos vartotojas atpažįstamas pagal sistemai pateiktus prisijungimo duomenis: vartotojo vardą ir slaptažodį.
3. Galimybė vartotojui keisti prisijungimo duomenis.
4. Formuoti keleivių srautų ataskaitą:
    - ✓ Galimybė filtruoti analizuojamų duomenų periodą;
    - ✓ Analizuojamų duomenų periodo ribos pateikiamos programos;
    - ✓ Galimybė filtruoti maršrutus.
5. Formuoti optimaliausių kontrolės vietų ataskaitą:
    - ✓ Galimybė filtruoti analizuojamų duomenų periodą;
    - ✓ Analizuojamų duomenų periodo ribos pateikiamos programos;
    - ✓ Galimybė filtruoti maršrutus.
6. Formuoti pažymėtų bilietų pagal tipą ataskaitą
    - ✓ Galimybė filtruoti analizuojamų duomenų periodą;
    - ✓ Analizuojamų duomenų periodo ribos pateikiamos programos;
    - ✓ Galimybė filtruoti maršrutus.
7. Formuoti rekomenduojamo transporto skaičiaus ataskaitą
    - ✓ Galimybė filtruoti analizuojamų duomenų periodą;
    - ✓ Analizuojamų duomenų periodo ribos pateikiamos programos;
    - ✓ Galimybė filtruoti maršrutus.
8. Formuoti išvažiavimo laikų ataskaitą
    - ✓ Galimybė filtruoti analizuojamų duomenų periodą;

- ✓ Analizuojamų duomenų periodo ribos pateikiamos programos;
- ✓ Galimybė filtruoti maršrutus.

9. Informuoti sistemos vartotoją apie klaidas sistemos darbo metu atitinkamais pranešimais.
10. Vartotojo atliekamų veiksmų korektiškumo automatinis kontroliavimas

3.3.2. Sistemos nefunkciniai reikalavimai

1. Sąsaja turėtų būti patraukli vartotojui: spalvos turi būti neryškių spalvų, suderintos tarpusavyje.
2. Vartotojo sąsaja privalo būti grafinė, naudojami tokie sąsajos elementai kaip:
   - ✓ Langai;
   - ✓ Dialogai;
   - ✓ Meniu
3. Neįkyri sąsaja. Vartotojo neturi erzinti pašaliniai ir dažni sistemos užklausimai.
4. Lengvai skaitoma sąsaja. Kompiuterio ekrane visa informacija turi būti vaizduojama tvarkingai. Tekstinės informacijos išvedimui turi būti parinktas lengvai skaitomas šriftas.
5. Vartotojo sąsaja turi atspindėti taikomąją sritį: naudoti meniu ir mygtukus, pavadintus dalykinės srities sąvokomis.
6. Nesudėtingas panaudojamumas. 80% - 90% sėkmingo panaudojimo dirbant su sistema pirmą kartą.
7. Programa orientuota į Lietuvos rinką, t.y. sukurta programinė įranga turės lietuviškus tekstus.
8. Kiekvienai operacijai, kuri pasirenkama iš meniu, priskirti klavišų kombinaciją, kad patyrę sistemos vartotojai galėtų greičiau iškviesti norimas operacijas.
9. Programa turi veikti stabiliai, nepadarant žalos aplinkai, vartotojo programinei bei techninei įrangai.
10. Sistemos turi būti suprojektuota taip, kad būtų galima ją lengvai papildyti naujo funkcionalumo komponentais (komponentinis programavimas).
11. Sistema gali naudotis daug vartotojų vienu metu
12. Sistema turi veikti Windows NT/ 2000/XP/ME platformose.
13. Sistemos verslo logika turi būti atskirta nuo taikomosios logikos.
14. Sistema turi būti apsaugota nuo nesankcionuotų priėjimo prie jos resursų, t.y.: prisijungimas prie duomenų bazės turi būti apsaugotas slaptažodžiu, norint atlikti svarbesnį redagavimą ar valdymą reikia identifikuoti vartotoją.
15. Sistemos kontekste neturėtų būti žargonų, barbarizmų ar necenzūrinių žodžių.
16. Dokumentacijos standartizavimas. Naudoti tokią dokumentavimo sistemą, kuri leistu keičiantis sistemos reikalavimams su minimaliomis sąnaudomis ją atnaujinti.

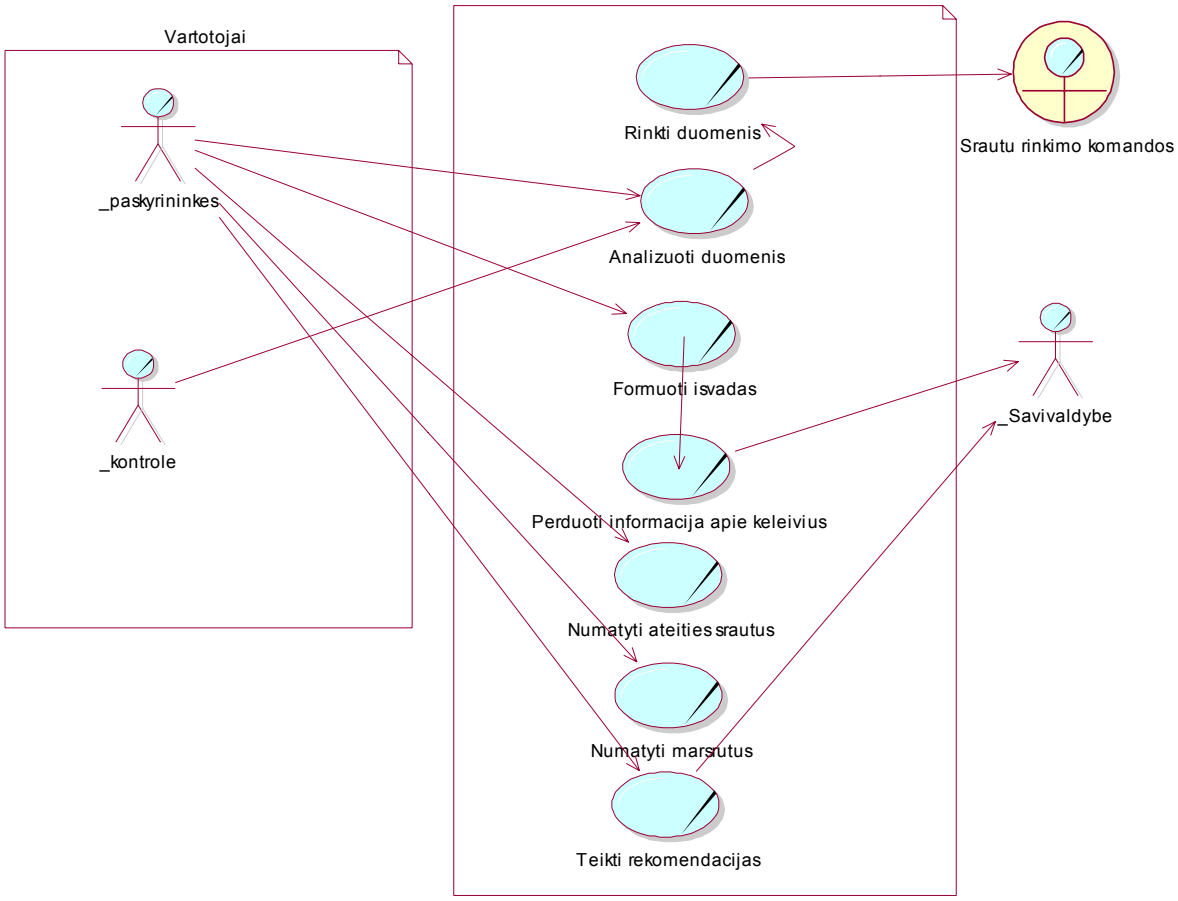
17. Sistemai kurti naudoti tik licencijuotą programinę įrangą.
18. Sistema turi nepažeisti Lietuvos Respublikoje galiojančių įstatymų bei teisės normų.
19. Visos teisės į produktą priklauso užsakovui.

3.4. Sistemos architektūra

3.4.1. Panaudojimo atvejų diagramos

3.4.1.1. Veiklos kontekstinė diagrama

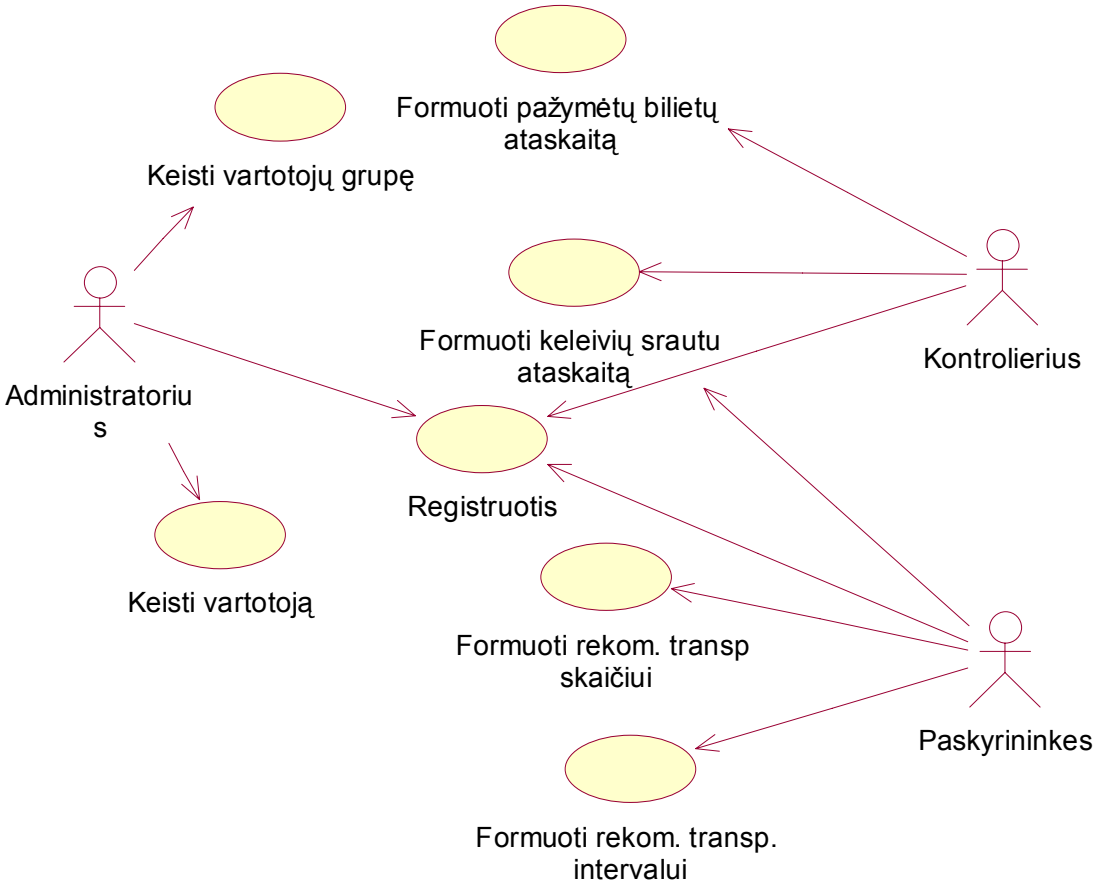
Panaudojimo atvejų diagramos taikomos, modeliuojant veiklą (veiklos panaudojimo atvejų diagramos) ir modeliuojant vartotojų poreikius reikalavimų modelyje Todėl analizuojant funkcinius sistemos reikalavimus, suformuotus vartotojo, pirma sudaromos veiklos panaudojimo atvejų diagramos, kurios atspindi analizuojamoje organizacijoje vykdomus veiklos procesus.



3.1 pav. Detalaus lygio veiklos kontekstinė diagrama

3.4.1.2. Aukšto lygio panaudojimo atvejų diagrama

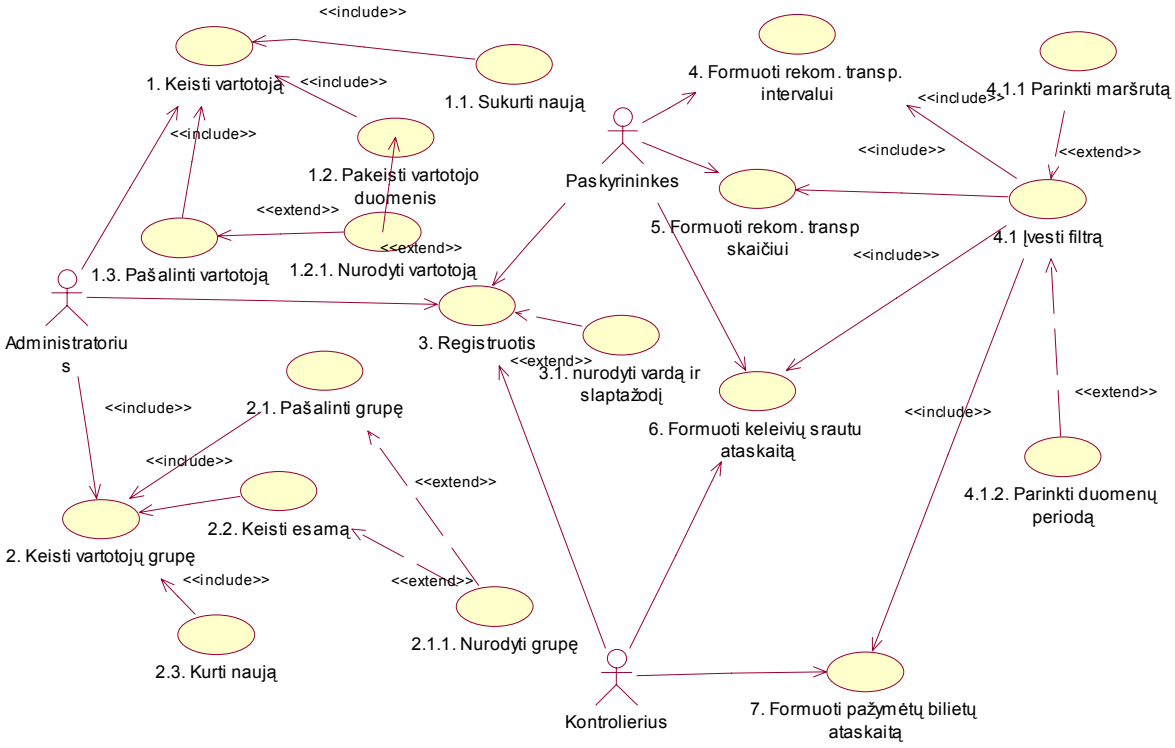
Naudojant veiklos panaudojimo atvejų modelį pradedamas formuoti reikalavimų panaudojimo atvejų modelis. Rengiamasi kompiuterizuoti ataskaitų formavimą, ateities srautų numatymą, maršrutų numatymą, duomenų analizavimą. Sudaroma aukšto lygio panaudojimo atvejų diagrama.



3.2 pav. Aukšto lygio panaudojimo atvejų diagrama

3.4.1.3. Detalizuota aukšto lygio panaudojimo atvejų diagrama

Sistemos funkcionalumo reikalavimai yra pateikiami atitinkamais panaudojimo atvejais. Toliau taikomas architektūros projektavimo „iš viršaus“ metodas – pagrindiniai panaudojimo atvejai yra padalinami į mažesnius panaudojant išplėtimo ir įtraukimo ryšius:



3.3 pav. Detalizuota aukšto lygio panaudojimo atvejų diagrama

3.4.1.4. Apibendrintų panaudojimo atvejų aprašymas

Apibendrintų panaudojimo atvejų aprašas pateikiamas lentelėje. Išskiriami aktoriai su kuriais panaudojamo atvejis sąveikauja, jo aprašas, ir tenkinimo kriterijus – įvykdymo požymis:

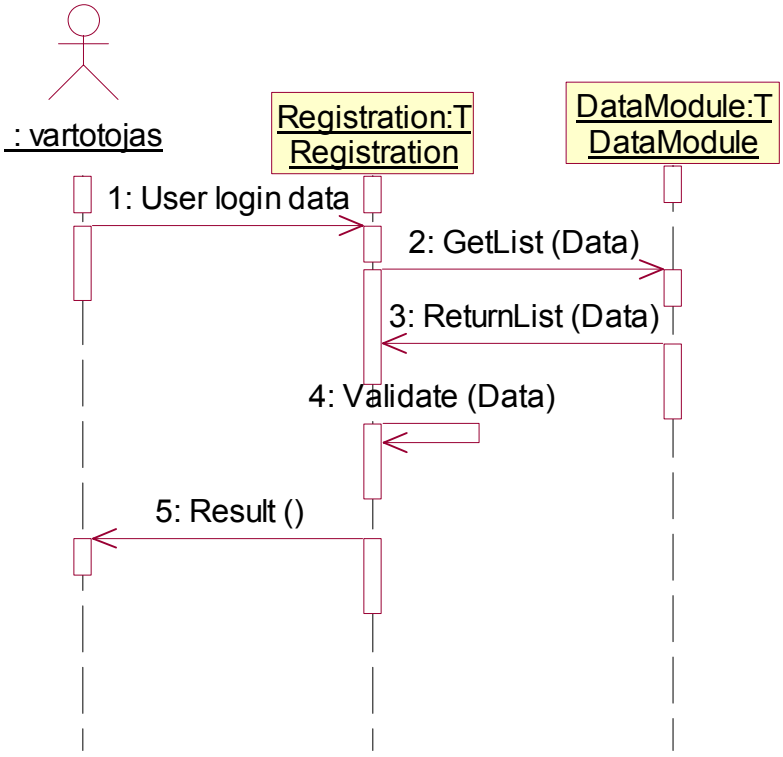
3.1 lentelė

Apibendrintų panaudojimo atvejų sąrašas

Panaudojimo Atvejo Nr.	Aktoriaus pavadinimas	Aprašas	Tenkinimo kriterijus
1.	Administratorius	Keisti vartotoją	Pakeistas įrašas sistemos DB
2.	Administratorius	Keisti vartotojų grupę	Pakeistas įrašas sistemos DB
3.	Administratorius, Vartotojas	Registruotis	Sistema paruošiama darbui
4.	Vartotojas	Formuoti rekomendacijas transporto intervalui	Naudojant sistemą sukuriamas rekomenduojamas transporto važiavimo grafikas
5.	Vartotojas	Formuoti rekomendacijas transporto skaičiui	Naudojant sistemą sukuriamas rekomenduojamas transporto skaičius maršrute
6.	Vartotojas	Formuoti keleivių srautų ataskaitą	Naudojant sistemą sukuriama keleivių srautų ataskaita
7.	Vartotojas	Formuoti pažymėtų bilietų ataskaitą	Naudojant sistemą sukuriama pažymėtų sustojimuose bilietų ataskaita
1.1.	Administratorius	Sukurti naują vartotoją	Pakeistas įrašas sistemos DB
1.2.	Administratorius	Pakeisti vartotojo duomenis	Pakeistas įrašas sistemos DB
1.3.	Administratorius	Pašalinti vartotoją	Pakeistas įrašas sistemos DB
2.1.	Administratorius	Pašalinti grupę	Pakeistas įrašas sistemos DB
2.2.	Administratorius	Keisti esamą	Pakeistas įrašas sistemos DB
2.3.	Administratorius	Kurti naują	Pakeistas įrašas sistemos DB
3.1.	Administratorius, Vartotojas	Nurodyti vardą ir slaptažodį	Prisijungimo dialoge įrašomi vartotojo prisijungimo duomenys
4.1.	Vartotojas	Įvesti filtrą	Sistemoje suformuotas filtras duomenų užklausoje
1.2.1.	Administratorius	Nurodyti vartotoją	Sistemai nurodomas vartotojas
2.1.1.	Administratorius	Nurodyti grupę	Sistemai nurodoma grupė
4.1.1.	Vartotojas	Parinkti maršrutą	Sistemoje suformuotas maršruto filtras
4.1.1	Vartotojas	Parinkti duomenų periodą	Sistemoje suformuotas duomenų periodo filtras

3.4.2. Veiksmų sekos diagramos

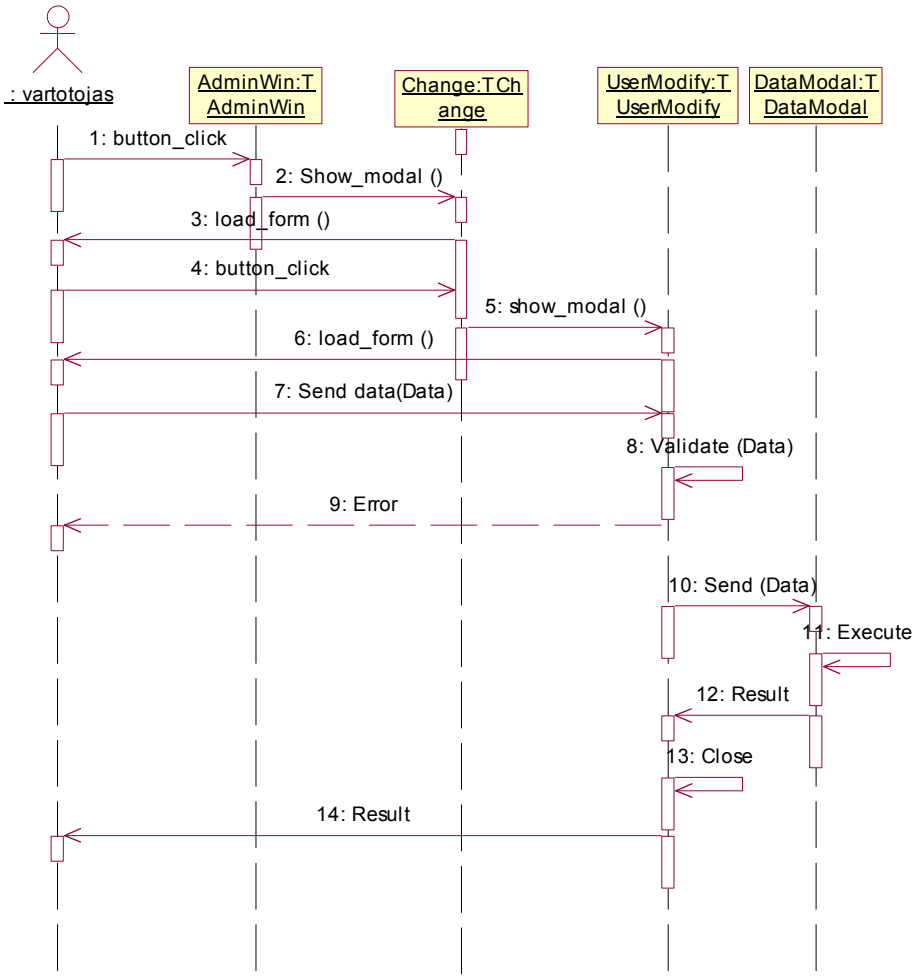
Scenarijai pateikiama pagal pagrindinę darbo su sistema seką. Pirmiausiai, prieš pradėdamas darbą, vartotojas turi prisijungti prie sistemos. Tam reikia suvesti vartotojo vardą ir slaptažodį. Taip sistema apsaugoma nuo nesankcionuoto priėjimo. Vartotojų vardai ir slaptažodžiai saugomi sistemos DB.



3.4 pav. Vartotojo prisijungimo prie sistemos sekų diagrama

Norėdamas prisijungti prie sistemos, vartotojas įveda savo vartotojo vardą ir slaptažodį. Paspaudus prisijungimo klavišą, sistema siunčia užklausimą į DB, gautas atsakymas tikrinamas ir išvedamas atitinkamas rezultatas. Administratorius pasirenka meniu punktą keisti vartoją. Iššaukiamas keitimo dialogas, kuriame nurodomas tikslus norimas atlikti veiksmas. Iššaukiamas duomenų redagavimo dialogas. Užpildžius formą vykdomas duomenų tikrinimas. Jei tikrinimo metu klaidų nerasta, atnaujinama DB.

Identiškai vykdama ir sistemos vartotojų grupių keitimas, bei priėjimo lygio grupėms suteikimas.



3.5 pav. Vartotojo keitimo sekų diagrama

Vartotojas pasirenka ataskaitos formavimo meniu punktą. Sistema siunčia užklausą, apie esančias surinktų duomenų datas sistemoje. Vartotojui išvedama forma su leidžiamom pasirinkti datomis. Vartotojui pasirikus duomenų periodą ir papildomus filtrus, sistema atrenka duomenis pagal nurodytus filtravimo parametrus. Duomenys yra išanalizuojami ir rezultatas pateikiamas ataskaitos šablono formai. Forma užkraunama vartotojui.

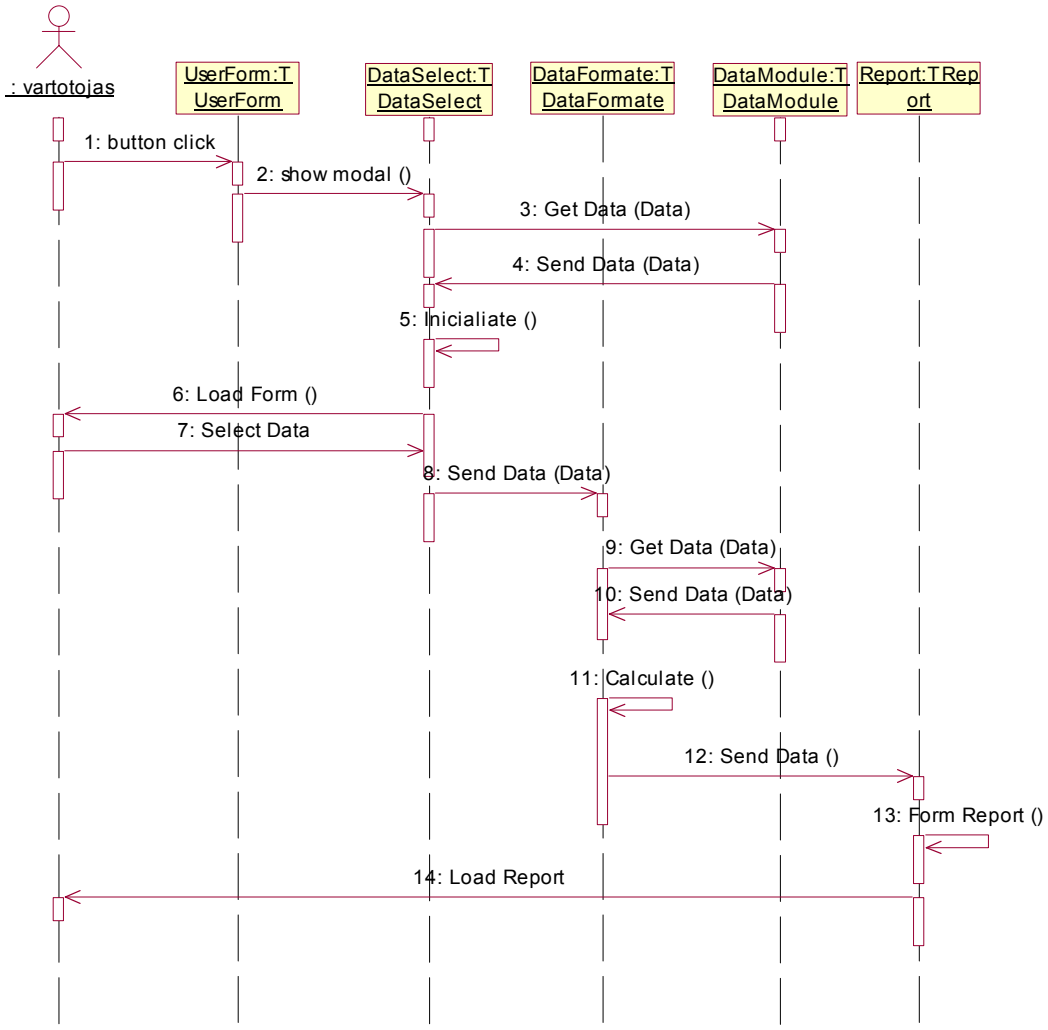
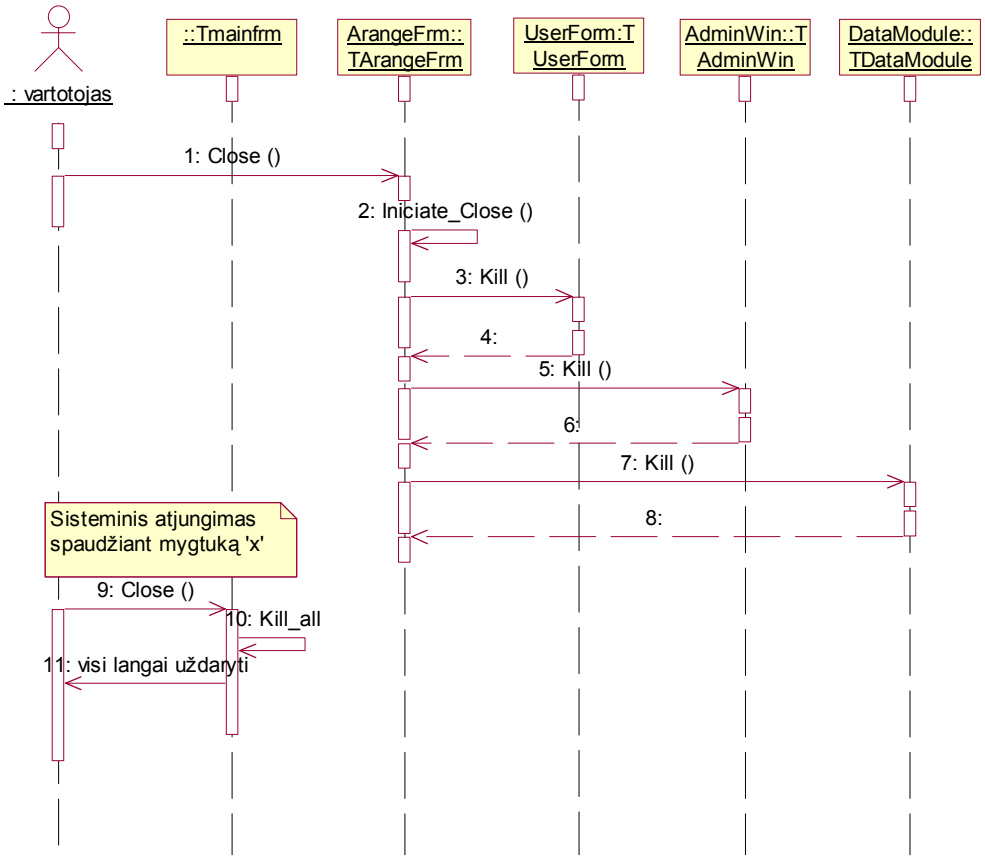


Figure 3.6. Ataskaitos formavimo sekų diagrama

Baigiant darbą su sistema, uždaromi visi dialogo su vartotoju laukai ir inicijuojamas atsijungimo veiksmas.

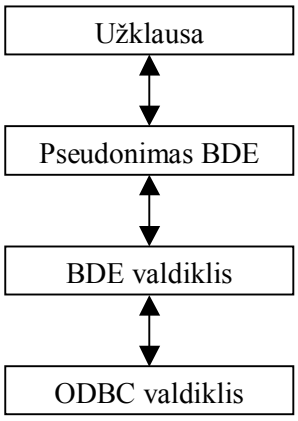


3.7 pav. Programos uždarymo proceso sekų diagrama

3.4.3. Architektūros detalizavimas

Sistema realizuojama komponentais, kurie atlieka tam tikrą užduotį. Bendras sistemos funkcionalumas sudaromas susiejant šiuos komponentus atitinkamais ryšiais – naudojama į paslaugą orientuota architektūra.

Šiuo metu sistemos platforma yra Windows. Darbui su vietinėmis ir nutolusiomis DB, naudojamas mechanizmas priėjimui prie duomenų, vadinamą Borland duomenų bazės varikliu (BDE - Borland Database Engine). Priėjimui prie nutolusios DB, pvz. SQL, BDE, naudojant SQL valdiklį (ODBC), jungiamasi prie SQL bazės serverio. Scheminis komponentų bendradarbiavimas pavaizduotas sekančiame paveiksle.

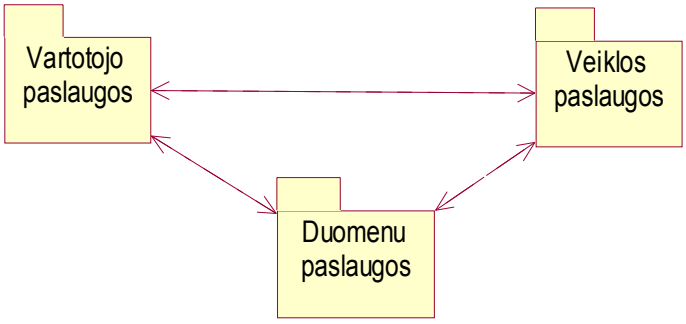


3.8 pav. Scheminis komponentų bendradarbiavimas

3.4.4. Sistemos loginiai komponentai

Sistema projektuojama panaudojant 3T architektūrą. Kiekviena pakopai sudaromas atskiras klasių, sąsajų ir modulių loginis rinkinys – UML paketas. Taip pasiekiamas lygių silpnas susietumas ir suprantamumas:

:



3.9 pav. Architektūros modelis

Kiekvienas paketas atitinka veiklos tipą:

- Vartotojo paslaugų paketas – vartotojo sąsajos projektavimas.
- Veiklos paslaugų paketas – sistemos funkcionalumo projektavimas.
- Duomenų paslaugų paketas – duomenų struktūrų projektavimas.

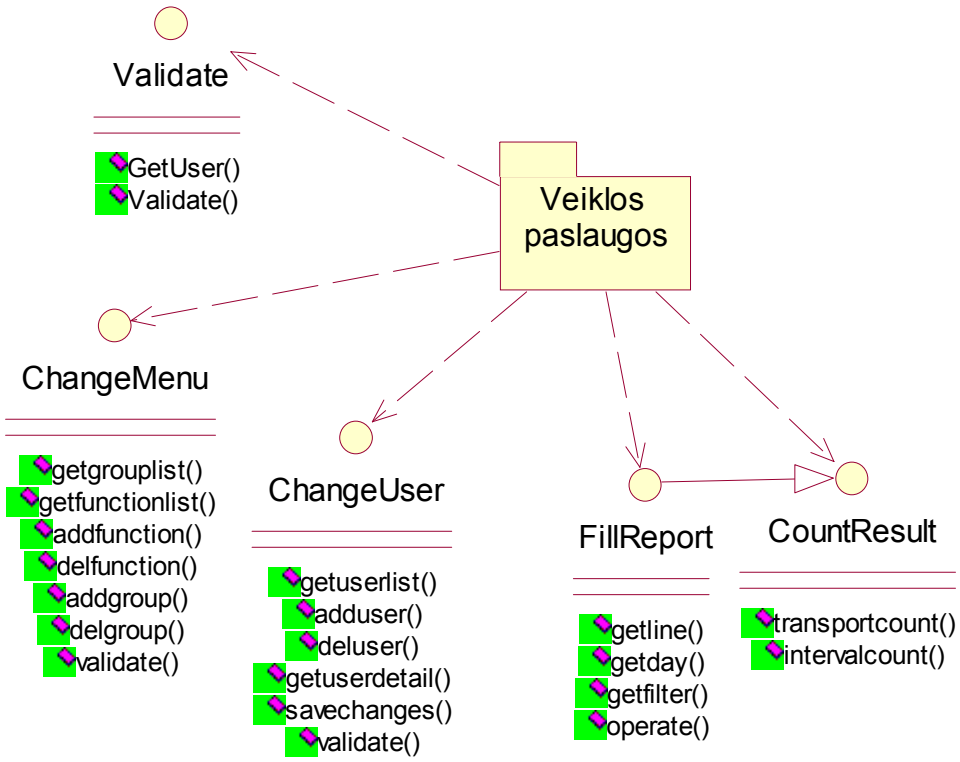
Vartotojo paslaugų paketas

Vartotojo paslaugų paketas susideda iš dviejų pagrindinių funkcijų – atvaizdavimo logikos ir atvaizdavimo išdėstymo. Atvaizdavimo išdėstymas išreiškia vartotojo sąsają (langai, iškrentantys sąrašai, ir pan.).

Pagrindinės formos moduliai:

- MainForm. Realizuoja prisijungimo prie sistemos formos grafinę sąsają. Atlieka vartotojo registraciją ir grupės priskyrimą.
- AdminWin. Realizuojama grafinė sąsaja į veiksmų su sistemos vartotojais grupes ir sistemos nustatymus aktyvizuojant atitinkamus grafinius elementus.
- UserForm. Realizuojama grafinė sąsaja su pagrindiniais sistemos panaudojimo funkciniais elementais. Atliekami apribojimai vartotojo veiksmams.
- Addgroups. Grupės koregavimo formos modulis. Sąsaja su vartotoju. Atliekami duomenų pakeitimo veiksmai.
- AddUserMenu. Vartotojo koregavimo formos modulis. Sąsaja su vartotoju. Atliekami duomenų korektiškumo tikrinimo ir pakeitimo veiksmai.
- BestControlPl. Ataskaitos generavimo formos modulis. Atliekama duomenų išdėstymas pagal iš anksto suformuotus reikalavimus.
- ChangeGroup. Grupės koregavimo formos modulis. Sąsaja su vartotoju. Atliekami duomenų pakeitimo veiksmai.
- ChangeMenu. Vartotojo koregavimo formos modulis. Sąsaja su vartotoju. Atliekami duomenų korektiškumo tikrinimo ir pakeitimo veiksmai.
- PassStreamRep. Ataskaitos generavimo formos modulis. Atliekama duomenų išdėstymas pagal iš anksto suformuotus reikalavimus.

Veiklos paslaugu paketas



3.10 pav. Veiklos paslaugų klasių paketas

Validate – verslo objekto, tikrinančio vartotoją ir jo teises klasė.

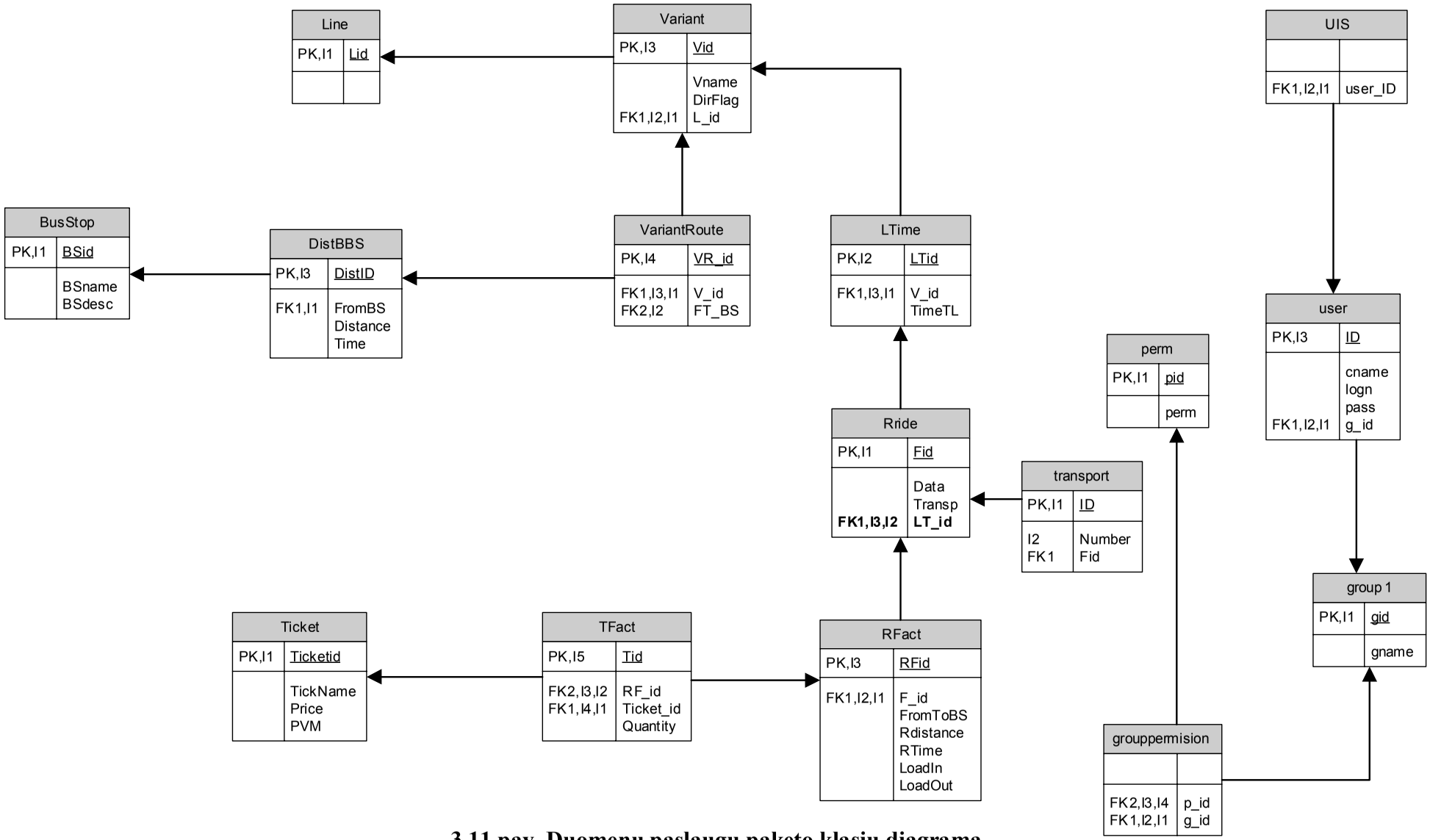
ChangeMenu, ChangeUser– standartinio verslo objekto, grąžinančio kursorių į verslo įrašų sąrašą, elgsena. Sąrašo nuskaitymas, veiksmai su įrašu klasė.

FillReport – verslo objekto realizuojančio sąrašo nuskaitymą, analizę ir apdorojimą klasė.

CountResult – verslo objekto, apdorojančio galimus duomenis ir formuojančio išvadas klasė.

Duomenų paslaugos paketas

Šis paketas susideda iš pačių duomenų ir informacijos apie jų konfigūraciją. Tai reliacinė SQL DB, suteikianti taikomajai programai universalumą ir darbo efektyvumą. Ši pakopa atsakinga už duomenų valdymą. DB architektūroje minimizuojamas ryšių ir elementų skaičius, paliekamos tik lentelės su atributais.



3.11 pav. Duomenų paslaugų paketo klasių diagrama

Line – Esybė skirta transporto maršrutams. Sudaryta iš atributo *Lid* nurodančio unikalų identifikatorių ir kartu maršruto numerį.

Variant – maršruto važiavimo kryptis. Atributai: *Vid* – unikalus esybės identifikatorius; *Vname* – krypties pavadinimas; *DirFlag* – krypties loginė reikšmė; *L_id* – maršruto numeris.

VariantRoute – maršruto važiavimo krypties sustojimų grupės. Atributai: *VR_id* – unikalus esybės identifikatorius; *V_id* – maršruto važiavimo kryptis; *FT_BS* - sustojimo grupės atkarpa.

DistBBS – sustojimo grupės atkarpa. Atributai: *DistID* – unikalus esybės identifikatorius. *FromBS* – pradinė, galinė stotelės; *Distance* – atstumas tarp sustojimų; *Time* – važiavimo laikas.

BusStop – stotelių pavadinimai. Atributai: *BSid* – unikalus esybės identifikatorius; *BSName* – stotelės pavadinimas; *BSDescription* – stotelės aprašymas.

LTime – maršruto išvykimo iš pradinio taško laikai. Atributai: *LTid* – unikalus esybės identifikatorius; *V_id* – maršruto važiavimo kryptis; *TimeTL* - išvykimo laikas.

Transport – individualios transporto priemonės identifikacinis numeris. Atributai: *ID* – unikalus esybės identifikatorius; *Number* – transporto identifikacinis numeris; *Fid* – transporto statusas.

Rride – važiavimo fakto identifikacija. Atributai: *Fid* – unikalus esybės identifikatorius; *Data* – važiavimo data; *Transp* – transportas; *LT_id* –išvykimo iš pradinio taško identifikatorius.

RFact – fiksuojamas atstumas tarp sustojimų grupės atkarpos ir įlipančių bei išlipančių keleivių skaičius. Atributai: *RFid* – unikalus esybės identifikatorius; *F_id* – papildomas atributas; *FromToBS* – sustojimo atkarpos identifikatorius; *Rdistance* – atkarpos realus atstumas; *Rtime* – atkarpos važiavimo laikas; *LoadIn* – įlipusių keleivių kiekis; *LoadOut* – išlipusių keleivių kiekis.

TFact – važiavimo metu tam tikroje atkarpoje pažymėti bilietai. Atributai: *Tid* – unikalus esybės identifikatorius; *RF_id* – fiksuojamų duomenų identifikatorius; *Ticket_id* – bilieto rūšies identifikatorius; *Quantity* – pažymėtų bilietų kiekis.

Ticket – sistemoje esamos bilietų rūšys. Atributai: *Ticketid* – unikalus esybės identifikatorius; *TickName* – bilieto pavadinimas; *Price* – bilieto kaina; *PVM* – PVM mokestis.

User – informacija apie sistemos vartotojus. Atributai: *ID* – unikalus esybės identifikatorius; *cname* – vartotojo vardas ir pavardė; *pass* – prisijungimo slaptažodis; *logn* – prisijungimo vardas; *g_id* – grupės identifikatorius.

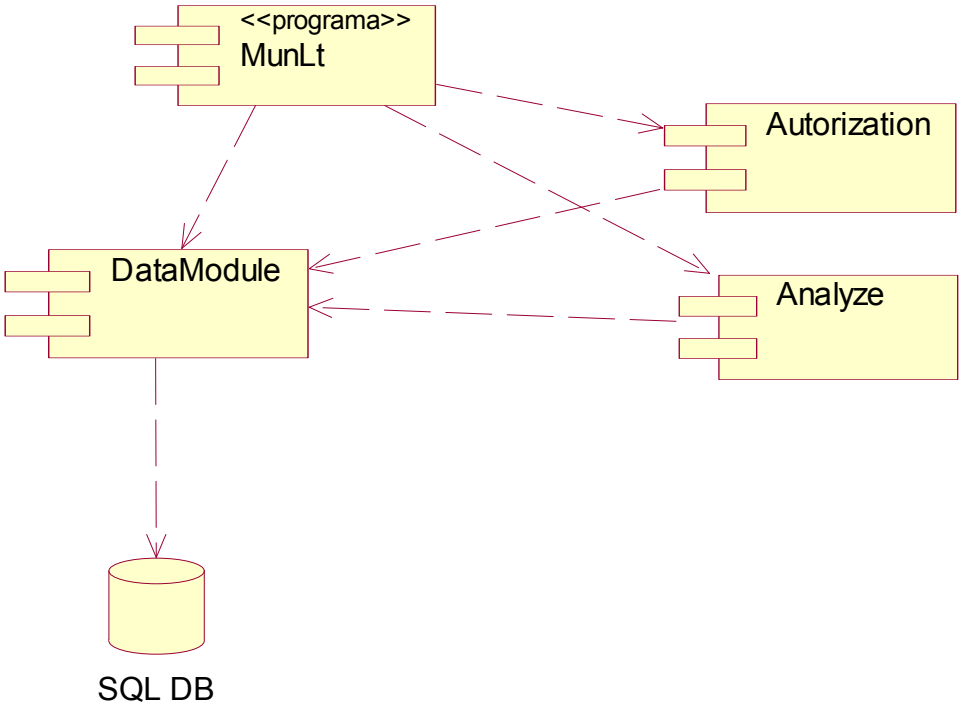
Group1 – informacija apie vartotojų grupes. Atributai: *Gid* – unikalus esybės identifikatorius; *Gname* – grupės pavadinimas.

Grouppermission – grupei leidžiami veiksmai. Atributai: *p_id* – grupei leidžiamo veiksmo identifikatorius; *g_id* – grupės identifikatorius.

Perm – vartotojui leidžiami veiksmai. Atributai: *pid* – unikalus esybės identifikatorius; *perm* – pavadinimas.

3.4.5. Sistemos komponentai

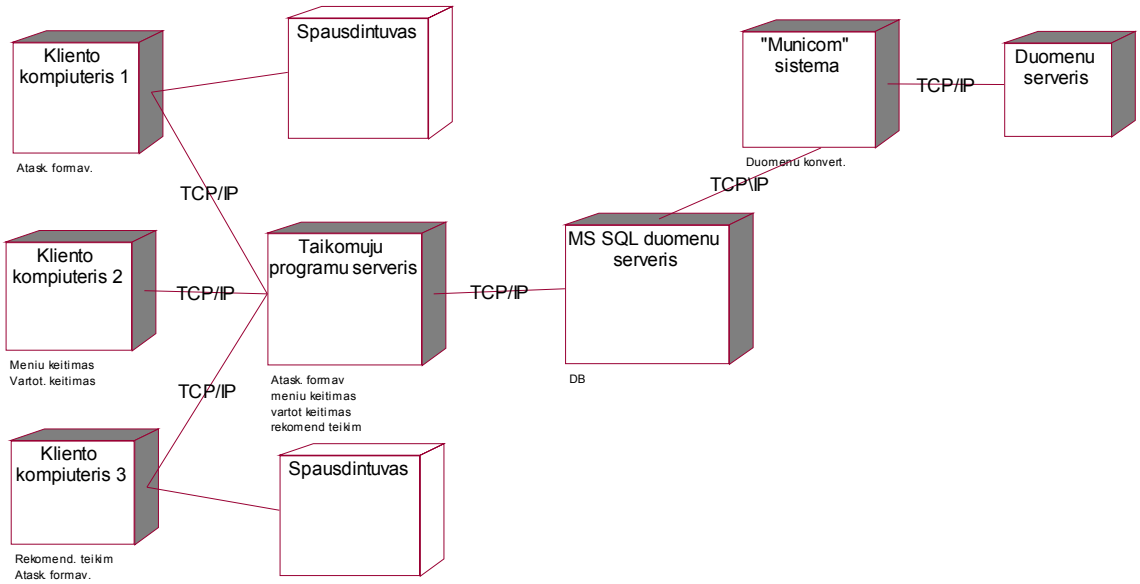
Sistemos komponentų diagramoje :



3.12 pav. Sistemos komponentų diagrama

1. MunLT – tyrimo programinė įranga. (paleidžiamasis failas).
2. Analyze – komponentas, kuriame saugoma Verslo logikos elgsenos aprašymo branduolys,
3. DataModule - sujungimų su DB,, transakcijų mechanizmo, verslo bibliotekų dispečerio funkcionalumas.
4. Autorization – vartotojo atpažinimo ir teisių suteikimo logikos elementų branduolys.
5. SQL DB – DB Serveris.

Galimas sistemos išdėstymo pas klientą vaizdas:



3.13 pav. Sistemos išdėstymo vaizdas

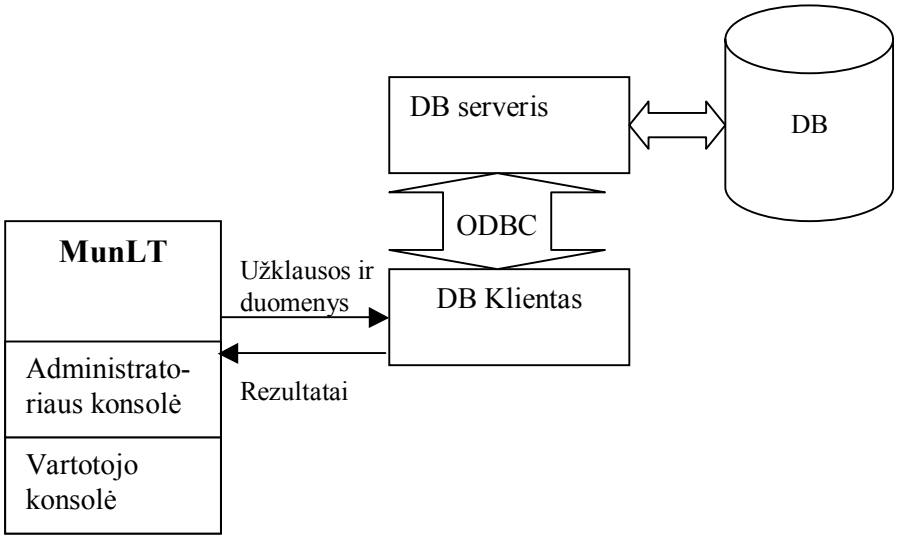
3.5. Sistemos funkcinis aprašymas

Sistema suskaidyta į kelis funkcinius lygmenis:

- ✓ Administratoriaus
- ✓ Vartotojo

Administratoriaus sąsajos lygmens funkcijos yra sistemos parametrų ir vartotojų valdymas. Vartotojo sąsajos lygmens funkcijos yra analizių atlikimas.

Bendras sistemos darbas yra apsaugomas prisijungimo teisėmis ir vartotojais. Kadangi duomenys yra konfidencialaus pobūdžio, todėl sistemos saugumo lygis užtikrinta ir sistemos gyvavimo ciklo ilgaamžiškumą. .

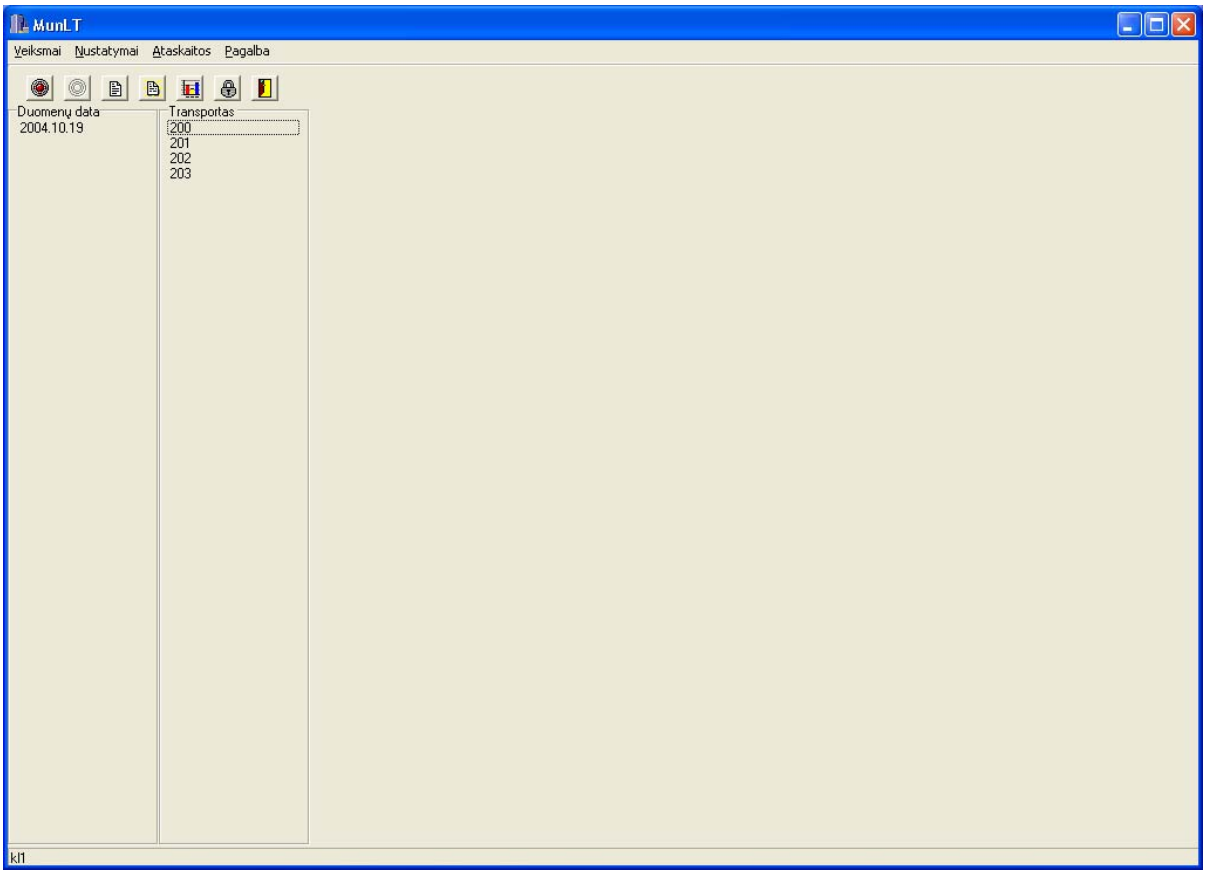


3.14 pav. Sistemos supaprastintas procesų vaizdas

3.5.1. Vartotojo meniu

Vartotojo meniu lauke „Duomenų data“ yra visos esančios duomenų bazėje važiavimo realizavimo datos. Lauke „Transportas“ matomi sistemoje užregistruotos transporto priemonės.

Visi veiksmai realizuojami naudojant meniu, esančiu viršutinėje programos lango dalyje, bei greitaisiais klavišais.

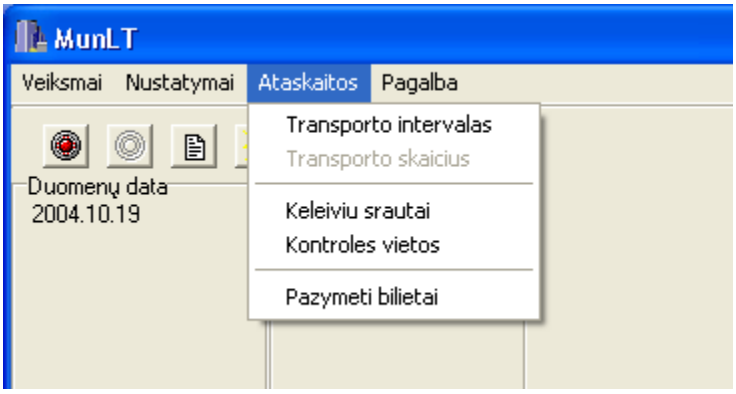


3.15 pav. Vartotojo meniu pagrindinis langas

Ataskaitų generavimas

Ataskaitų generavimas iškviečiamas iš vartotojo formos meniu punkto ataskaitos arba greitojo klavišo pagalba. Yra penki ataskaitų tipai:

- ✓ Transporto intervalas.
- ✓ Transporto skaičius.
- ✓ Keleivių srautai.
- ✓ Kontrolės vietos.
- ✓ Pažymėti bilietai.



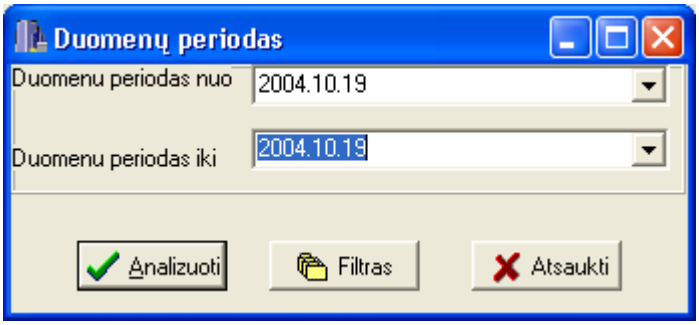
3.16 pav. Sistemos ataskaitų tipai

Visos vartotojui uždraustos daryti ataskaitos yra neaktyvios. Norint vartotojui suteikti teisę daryti ataskaitą, reikia iš administratoriaus meniu jo grupei priskirti tą ataskaitą.

Prieš sistemai sugeneruojant ataskaitą, vartotojas privalo nurodyti analizuojamų duomenų periodą. Pageidaujant galima nurodyti ir maršruto filtrą, t.y. kurie maršrutai turi būti įtraukti į ataskaitos formavimą.

Duomenų intervalo parinkimas

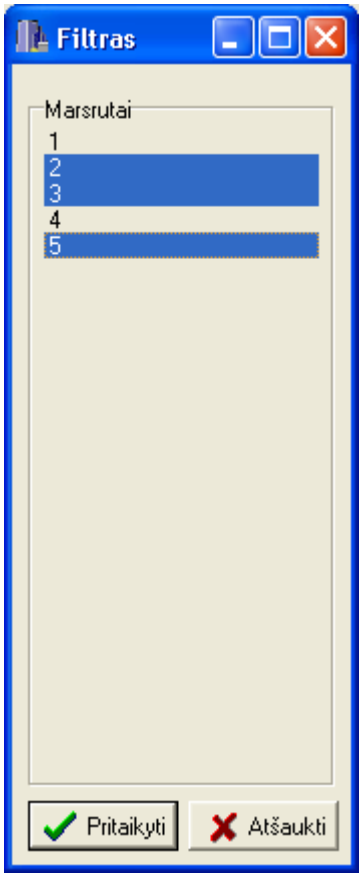
Duomenų intervalas nuo parenkamas spaudžiant ant rodyklės žemyn, esančios prie duomenų periodo nuo. Sistema išveda visas, sistemos duomenų bazėje esančių duomenų sudarymo datas. Analogiškai parenkamos ir „duomenų periodo iki“ Pasirinkus duomenų periodo datas spausti klavišą „Analizuoti“ arba norint grįžti į ankstesnį meniu spausti klavišą „Atšaukti“.



3.17 pav. Duomenų periodo pasirinkimo dialogas

Maršruto filtro parinkimas

Duomenų filtras pasirenkamas duomenų intervalo dialogo meniu paspaudus klavišą „Filtras“. Į ekraną išvedama maršruto parinkimo filtras. Galima pasirinkti kelis maršrutus, nuspaudus klavišą „Ctrl“ ir pelyte žymint maršrutus. Pasirinkus maršrutus spausti klavišą „Pritaikyti“. Filtras bus įsimintas ir sugrįžtama į ankstesnį dialogą. Paspaudus klavišą „Atšaukti“, filtras nebus pritaikytas ir sugrįžtama į ankstesnį dialogą.



3.18 pav. Maršruto filtro pasirinkimo dialogas

Ataskaita - transporto skaičius

Transporto skaičiaus ataskaita skirta gauti rekomenduojamą transporto skaičių maršrutui. Norint vartotojui atspausdinti rekomenduojamo transporto skaičiaus ataskaitą reikia iš vartotojo meniu pasirinkti „ataskaitos“ ir „transporto skaičius“.

Ataskaita - keleivių srautai

Transporto skaičiaus ataskaita skirta gauti keleivių srautus maršrutų sustojimuose. Norint vartotojui atspausdinti keleivių srautų ataskaitą reikia iš vartotojo meniu pasirinkti „ataskaitos“ ir „keleivių srautai“.

Ataskaita - kontrolės vietos

Transporto skaičiaus ataskaita skirta gauti optimaliausias keleivių kontrolės vietas. Analizė suskirstyta į laiko zonas kas dvi valandas. Pateikiamas stotelių atsižymėjimų procentinis santykis tarp įlipusių ir atsižymėjusių keleivių. Kuo procentas mažesnis, tuo didesnė tikimybė, kad tame sustojime bus rezultatyvus keleivių be bilietų tikrinimas. Norint vartotojui atspausdinti kontrolės vietų ataskaitą reikia iš vartotojo meniu pasirinkti „ataskaitos“ ir „kontrolės vietos“.

Ataskaita - pažymėti bilietai

Pažymėtų bilietų ataskaita skirta tirti bilietavimo sistemą. Kartu ataskaitoje pateikiama pinigų suma, gauta atžymėjus bilietus. Norint vartotojui atspausdinti keleivių srautų ataskaitą reikia iš vartotojo meniu pasirinkti „ataskaitos“ ir „pažymėti bilietai“.

Darbo pabaiga

Vartotojas darbą baigia pasirinkęs menių punktą „Atsijungti“ arba „Baigti“. Pasirinkus „Atsijungti“ vartotojas patenką į prisijungimo langą. Sistema pereina į laukimo rėžimą ir laukia kol prisiregistruos kitas vartotojas. Pasirinkus punktą „Baigti“ programa išjungiama.

4. PROGRAMINĖS ĮRANGOS TYRIMAS

4.1. Kokybės analizė

Sukurto produkto kokybės įvertinimui buvo sudarytas testavimo planas. Pagrindinis testavimo plano tikslas, atskleisti programinės įrangos trūkumus ir sumažinti klaidų kiekį programinėje įrangoje iki vartotojui priimtino kiekio.

Pagal testavimo planą, sukurto produkto kokybės įvertinimui buvo naudojamos sekančios testavimo priemonės [19]:

- ✓ Funkcinis testavimas: testuojama, kaip sistema atlieka įvairias operacijas, tikrinamas rezultatų teisingumas, remiantis vartotojo reikalavimų specifikacija. Funkcinio testavimo esmė:
    - o Ar sistema atlieka visas funkcijas numatytas vartotojo reikalavimų specifikacijoje?
    - o Ar atliktų funkcijų rezultatai sutampa su rezultatais apibrėžtais testavimo plane, reikalavimų specifikacijoje ?
    - o Kaip sistema reaguoja į nekorektiškus funkcijų duomenis ?
- ✓ Nefunkcinių reikalavimų testavimas. Testuojama, kaip sistemoje realizuotas funkcionalumas, ar atitinka vartotojo reikalavimų specifikaciją (nefunkciniai reikalavimai).
- ✓ Vartotojo sąsajos testavimas. Tikrinamas meniu, duomenų pateikimo teisingumas. Šis testavimo metodas skirtas aptikti sąsajos netikslumus. Taip pat testuojant sąsają buvo galima aptikti ir funkcionalumo klaidų, kurias detalizavome atlikdami papildomą funkcinį testavimą ir vėliau - atitinkamų komponentų testavimą. Sąsajos testavimo esmė:
    - o Ar veikia visi meniu punktai?
    - o Ar sąsaja tinkamai atvaizduojama?
    - o Ar sąsaja yra atnaujinama?
    - o Ar teisingai apdorojamas operacijos patvirtinimo nutraukimas?

4.2. Kokybės įvertinimas

Užsakovui buvo pateiktas programinio modulio prototipas, pirminė užduotis, vartotojo reikalavimų specifikacija, architektūros specifikacija, programos kodas ir kokybės vertinimo kriterijų lentelė, kurioje kiekvienas vertinimo kriterijus įvertinamas remiantis ISO įvertinimų sistema:

4.1 lentelė.

Programinės įrangos vertinimo sistema

*****	Labai gerai	Išlaikyta
****	Gerai	
***	Patenkinamai	
**	Nepatenkinai	Neišlaikyta
*	Labai silpnai	

Vartotojai pateiktą vertinimų lentelę užpildė sekančiai:

4.2 lentelė

Produkto kokybės įvertinimai

Prioritetas / Kriterijus	*****	****	***	**	*
Naudojamumas		+			
Saugumas	+				
Efektyvumas		+			
Teisingumas	+				
Patikimumas		+			
Palaikomumas	+				
Testuojamumas		+			
Lankstumas		+			
Suprantamumas	+				
Pakartotinis panaudojamumas			+		

Pagal įvertinimus matyti, jog pagerintas dauguma sistemos charakteristikų saugumas, efektyvumas, įsiliejamumas, lankstumas ir efektyvumas.

Buvo nustatyta sistemos tolesnė vizija:

1. Patvirtinti programos priežiūros planą pagal ISO/IEC 14764, Tarptautinį Programinės Įrangos Priežiūros standartą.[20].
2. Suderinti su papildomais DB tipais.

5. PROGRAMINĖS ĮRANGOS EKSPERIMENTINIS TYRIMAS

Norėdami palyginti naujai sukurtos ir pateiktos sistemos privalumus bei teikiamą naudą, privalome pirmiausia apskaičiuoti atliekamų tyrimų apimtis ir reikalaujamus žmogiškus resursus nenaudojant sistemos.

Tam, kad transporto tyrimų rezultatai būtų patikimi, reikia nustatyti būtinų tyrimų apimtį. Ji nustatoma tikimybių teorijos ir matematinės statistikos metodais. Tiriant keleivių srautus tyrimų apimtis galima apskaičiuoti pagal formulę [3]:

$$n = \frac{t^2 - \delta^2}{\Delta^2}. \tag{5.1}$$

t – tikimybės koeficientas priklausantis nuo būtino tikslumo; δ^2 - vidutinė kvadratinė rezultatų dispersija; Δ - leistinas nuokrypis (priklauso nuo to kokį patikimumą norine gauti).

Tiriant gyventojų transportinius ryšius galime laikyti, kad mieste yra *N* gyventojų. Iš jų yra a gyventojų su charakteristika VT, o b – likusi gyventojų dalis. Kadangi nėra galimybės apklausit visus gyventojus todėl negalime tiksliai nustatyti a ir b reikšmes.

Tarkime, kad apklausėme *n* gyventojų ir nustatėme, kad $a_0 = m / n$, tai sudaro gyventojų, kurių charakteristika *A*, dalį. Kadangi a_0 - tam tikro atsitiktinio dydžio realizavimas, tai *c* patikimumo ribos gali būti nustatytos naudojant tikimybių teoriją.

Šiuo atveju tyrėjus patenkintų 90% rezultatų patikimumas. Tada dydį *a* galime nustatyti taip:

$$a_0 - 1.64\sqrt{\frac{(N-n)a \cdot b}{(N-1)n}} \le a \le a_0 + 1.64\sqrt{\frac{(N-n)a \cdot b}{(N-1)n}}. \tag{5.2}$$

Kadangi nežinomos tikslios a ir b reikšmės, imamos max reikšmės. Žinom, kad b = 1 – a, tai max $a \cdot b = 0{,}25$. Be to a + b = 1. Tada:

$$a_0 - 1.64\sqrt{\frac{0{,}25(N-n)}{(N-1)n}} \le a \le a_0 + 1.64\sqrt{\frac{0{,}25(N-n)}{(N-1)n}}. \tag{5.3}$$

Tarkime, kad:

$$\delta = 0.82\sqrt{\frac{N-n}{(N-1)n}}, \tag{5.4}$$

gauname mus tenkinantį a nuokrypį nuo a_0.

Tada tyrimų apimtį esant 0,9 tikimybei nustatome taip:

$$n = \frac{N}{1 + 1{,}4872\delta^2 \cdot (N-1)}. \tag{5.5}$$

(5.5) lygtimi galime nustatyti skaičių gyventojų, kuriuos reikia apklausti tiriant jų transportinius ryšius ir 90% užtikrinant tyrimų patikimumo lygį. Apskaičiuojame Kauno mieste atliekamų apklausų apimtis norint užtikrinti 90 % tyrimų patikimumo lygį. Gyventojų skaičių *N* žinome iš statistikos duomenų [21], skaičiuojame tyrimų apimtis:

$$n = \frac{368900}{1 + 1.4872 * 0.000025754 * 368900} = 24383.$$

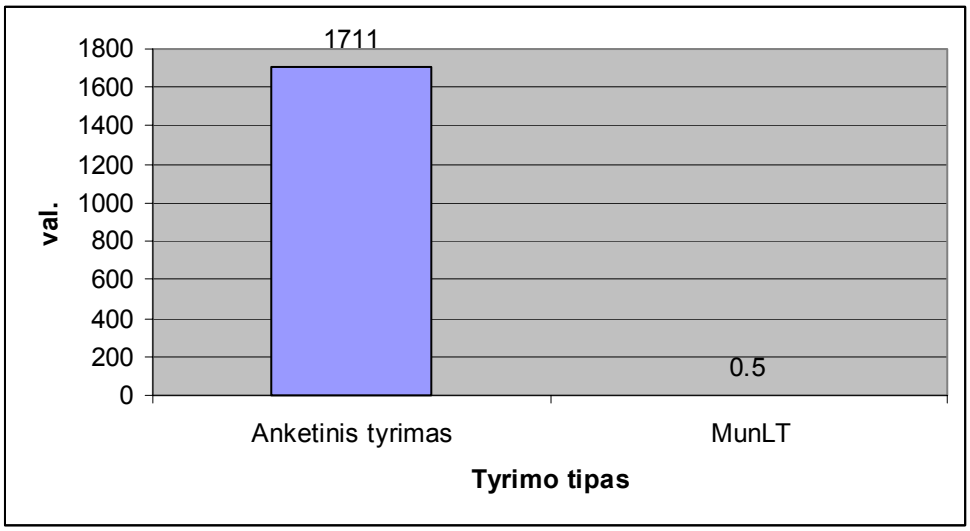
Gaunam tyrimų apimties skaičių:

n = 24383.

Norint atlikti tokios apimties tyrimą, reikia apklausti 24383 žmones. Jei skaičiuosim, kad vienos anketos užpildymui sugaištama 3 min, tai tyrimui atlikti prireiks 1283 val.

Duomenų analizavimui surinktus duomenis reikia suvesti. Vienos anketos įvedimas trunka 30 s. Tyrimo duomenims suvesti sugaištama 428 val. Viso duomenų analizei sugaišta 1711 val.

Tuo tarpu naudojant sistemą MunLt, duomenų analizė užima 0,5 val., neskaičiuojant laiko, kuris reikalingas surinkti duomenis iš transporto.



5.1 pav. Darbo sąnaudų palyginimas pagal tyrimo tipą

6. IŠVADOS

1. Sistemos prototipas sukurtas naudojant objektiškai orientuotą programavimą. Sistemos kūrimo procesas griežtai dokumentuotas. Atliktas sistemos funkcinis ir struktūrinis testavimas.
2. Programos kūrimo procesas atliktas laikantis sistemos kūrimo taisyklių, ko dėka produktas užsakovui pateiktas laiku, biudžetas neviršijo numatyto, sistema pilnai dokumentuota ir ištestuota.
3. Atliktas sistemos įtakos vartotojo darbui eksperimentinis tyrimas parodė sistemos keleivių srautų ir bilietavimo tyrimo greičio pranašumą lyginant su tradiciniais tyrimo būdais.
4. Išanalizavus sistemos kokybės įvertinimą pastebėta, kad norint sėkmingai užtikrinti sistemos MunLt gyvavimą, būtina sistemą suderinti su papildomais DB tipais ir praplėsti sistemos funkcijų kiekį.

7. LITERATŪRA

1. Clements P., Bochmann F., Bass L. Documenting Software Architecture. 2002. P. 7–83.
2. Jorgensen P. Software Testing. A Craftsman‘s Approach. Second Editon. 2002. P. 29–51.
3. Butkevičius J. Keleivių vežimai. Monografija. Technika. Vilnius, 2002. P. 146–157.
4. Kitas požiūris į programinės įrangos atkartojimą [Žiūrėta 2005.02.23]. Prieiga per internatą: < http://www.soften.ktu.lt/~ziber/t120m013/Jerzabek.pdf>.
5. Coombs P. IT Project Estimation. A Practical Guide to the Costing of Software. Cambridge university press. 2003. P. 42–56.
6. IEEE 1219-19998 IEEE standarto programos palaikymui aprašymas. [Žiūrėta 2005.04.23]. Prieiga per internetą: <http://standards.ieee.org/reading/ieee/std_public/description/se/1219-1998_desc.html>.
7. Programos „Municom“ vartotojo vadovas. PZI Taran. 2000. P. 23–47.
8. Programos „Pikas“ reklaminė medžiaga [Žiūrėta 2005.02.10]. Prieiga per internetą <http://www.merakas.lt>.
9. Eismo logistikos sistema „PTV internal“ [Žiūrėta 2005.03.16]. Prieiga per internetą <http://www.ptv.de/cgi-bin/traffic/traf_ip.pl>.
10. Kliento/serverio architektūra [Žiūrėta 2005.02.19]. Prieiga per internetą <http://www.sei.cmu.edu/str/descriptions/clientserver_body.html>.
11. Kruchten P. The Rational Unified Process. An introduction. Second Editon. 2000. P. 23–52.
12. Boggs W., Boggs M. UML with Rational Rose 2002. 2002. P 15–76.
13. Standartinis krioklio modelis [Žiūrėta 2005.02.16]. Prieiga per internetą http://visualbasic.ittoolbox.com/browse.asp?c=VBPeerPublishing&r=%2Fpub%2FSPK112603%2Epdf.
14. Evoliucnis modelis [Žiūrėta 2005.04.15]. Prieiga per internetą <http://csweb.cs.bgsu.edu/maner/domains/Proto.htm>.
15. Formalus metodas [Žiūrėta 2005.03.12]. Prieiga per internetą <http://www.cs.odu.edu/~zeil/cs451/Lectures/02reqts/specmodel/>.
16. Pakartotinio panaudojimo modeli [Žiūrėta 2005.03.15]. Prieiga per internetą <http://www.sei.cmu.edu/publications/documents/92.reports/92.sr.004.htm>.
17. Kas yra ODBC [Žiūrėta 2005.03.21]. Prieiga per internetą: <http://www.techcourt.com/technologies/odbc-001.htm>.
18. Сигнор Р., Стеман М. Использование для доступа к базам данных. Москва, 1995. С. 14–50.

19. *ISO/IEC 14764:2001* Informacinės technologijos – programų priežiūra [Žiūrėta 2005.04.24]. Prieiga per internetą: < http://www.standards.com.au/select/Script/ Details.asp?DocN= AS267916628415&TDetailsSelect:tvRefs:node0_Expand=true#TDetailsSelect:tvRefs:node0 >.
20. Aksomaitis A. Tikimybių teorija ir statistika. Technologija. Kaunas, 2000. 340 p.
21. Kauno miesto gyventojų skaičius. [Žiūrėta 2005.05.03]. Prieiga per internetą: < http://www.kaunas.lt/miestas_/statistika/Teritorija/teritorija.shtml>.

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

PĮ – programinė įranga

3T - trijų pakopų architektūra

UML – Programinės įrangos projektavimo kalba (Unified Modeling Language)

DB – duomenų bazė

RUP – apibendrintas projektavimo metodas

ODBC - atvira priėjimo prie duomenų bazių sąsaja (Open Database Connectivity)

API - Taikomųjų programų sąsajos su nurodyta sistema programavimo priemonė (Application Programming Interface).