KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Donatas Mickevičius

APTARNAVIMO SISTEMŲ AGREGATINIŲ IMITACINIŲ MODELIŲ STATISTINĖS ANALIZĖS POSISTEMĖ

Magistro darbas

Vadovas

doc. dr. V. Janilionis

KAUNAS, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N. Listopadskis
2009 06 06

APTARNAVIMO SISTEMŲ AGREGATINIŲ IMITACINIŲ MODELIŲ STATISTINĖS ANALIZĖS POSISTEMĖ

Matematikos magistro baigiamasis darbas

Vadovas
doc. dr. V. Janilionis
2009 06 04

Recenzentas
doc. dr. V. Pilkauskas
2009 06 02

Atliko
FMMM 7 gr. stud.
D. Mickevičius
2009 06 01

KAUNAS, 2009

KVALIFIKACINĖ KOMISIJA

Pirmininkas – Leonas Saulis, profesorius (VGTU).

Sekretorius – Eimutis Valakevičius, docentas (KTU).

Nariai: Algimantas Jonas Aksomaitis, profesorius (KTU),

Arūnas Barauskas, Vice-prezidentas projektams (UAB „Baltic Amadeus“),

Vytautas Janilionis, docentas (KTU),

Zenonas Navickas, profesorius (KTU),

Vidmantas Povilas Pekarskas, profesorius (KTU),

Rimantas Rudzkis, valdybos pirmininko patarėjas („DnB NORD” bankas).

SUMMARY

Mickevičius D. STATISTICAL ANALYSIS SUBSYSTEM FOR AGGREGATE SIMULATION MODELS OF SERVICE SYSTEMS: Master's work in applied mathematics / supervisor assoc. prof. dr. V. Janilionis; Department of applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2009. – 114 p.

There are a lot of simulation systems, which allow making complicated systems models. The statistical analyses of these models are unfitted with new software packages of statistical analysis (SAS, SPSS and etc.). The relevant task is to develop tools for statistical analysis of service system simulation models to use modern statistical packages.

The goal of master's work is to improve statistical analysis abilities of service system models based on mathematical modelling scheme.

To reach this goal the following tasks were solved:

- Create aggregate simulation system SIMAS++/ASAIM;
- Create statistical analysis model for MathCad system;
- Create SAS macros, which perform statistical analysis;
- Create a statistical analysis subsystem for service system simulation models, which consists of simulation system SIMAS++, statistical analysis package SAS and universal mathematical software MathCad;
- Apply created software for models of service systems.

Created software allows to make, edit and to save service system models faster and easier and perform statistical analyses of model with MathCad and SAS. It signally extends the use of the results of service simulation models for statistical analysis.

Turinys

Paveikslų sąrašas

Lentelių sąrašas

ĮVADAS

Šiuo metu yra sukurta daug aptarnavimo sistemų imitacinio modeliavimo priemonių, kurios leidžia kurti sudėtingų sistemų modelius. Jų analizei reikia taikyti įvairius statistikos metodus, tačiau daugumos egzistuojančių modeliavimo sistemų statistinės analizės priemonės nepritaikytos darbui su šiuolaikinėmis statistinės analizės sistemomis (SPSS, SAS ir t.t.), nes jos yra uždaros ir turi tik aprašomosios statistikos pateikimo priemones. Todėl aktualus uždavinys - sukurti priemones, kurios leistų imitacinio modeliavimo duomenų analizei panaudoti šiuolaikinius statistikos paketus.

Viena iš matematinių schemų naudojamų aptarnavimo sistemų modeliavimui yra agregatinė matematinė schema.

Darbo tikslas – išplėsti aptarnavimo sistemų agregatinių imitacinių modelių statistinės analizės galimybes.

Siekiant įgyvendinti tikslą buvo sukurta aptarnavimo sistemų agregatinių imitacinių modelių statistinės analizės posistemė, kuri apjungė modeliavimo sistemą SIMAS++/ASAIM, universalųjį matematikos paketą MathCad ir statistinės analizės paketą SAS.

Agregatinei modeliavimo sistemai SIMAS++/ASAIM pasiūlytas universalus duomenų rinkimo ir modeliavimo duomenų matricos formavimo modelis. Modeliavimo duomenų matricos transformavimui į statistinės analizės matricas pasiūlyta metodika, kuri remiasi releacinės algebros operacijomis. Modeliavimo duomenų statistinė analizė atliekama universaliu matematikos paketu MathCad ir statistinės analizės paketu SAS.

Analitinėje dalyje patiekta aptarnavimo sistemų imitacinio modeliavimo sistemų apžvalga, jų veikimo principai, bei trūkumai. Pateiktas universalios aptarnavimo sistemos agregatinis modelis.

Tiriamojoje dalyje pateiktas valdomas matavimo operatorius, modeliavimo duomenų matricos formavimo principai, modeliavimo duomenų matricos transformavimo į statistinės analizės matricas metodika. Aprašyta sukurta aptarnavimo sistemų agregatinė imitacinio modeliavimo programa SIMAS++/ASAIM, statistinės analizės modeliai, kurie realizuoti universaliu matematikos paketu MathCad ir statistinės analizės paketu SAS. Taip pat pateikta statistinės analizės posistemės veikimo schema ir sudarymo principai. Sukurtos statistinės analizės positemės testavimui sudaryti du aptarnavimo sistemų agregatiniai modeliai. Jų testavimo rezultatai pateikti tiriamosios dalies pabaigoje.

Sukurta posistemė leidžia paprasčiau kurti, redaguoti ir išsaugoti aptarnavimo sistemos modelius, atlikti modeliavimo rezultatų statistinę analizę panaudojant universalų matematikos paketą MathCad ir statistinės analizės paketą SAS. Tai ženkliai išplečia aptarnavimo sistemų imitacinių modelių rezultatų statistinės analizės galimybes.

Darbo rezultatai pristatyti konferencijose „Taikomoji matematika 2008“ ir „Matematika ir matematinis modeliavimas“ (2009).

1. ANALITINĖ DALIS

1.1. IMITACINIO MODELIAVIMO SISTEMŲ IR JŲ STATISTINĖS ANALIZĖS PRIEMONIŲ ANALIZĖ

Sistemos imitacinį modelį galima parašyti bet kurioje universalioje algoritminėje kalboje (PASCAL, C, JAVA ir t.t.), tačiau tokių modelių sudarymas reikalauja daug aukštos kvalifikacijos specialistų darbo sąnaudų. Šiuo metu pasaulyje yra sukurta daug įvairių modeliavimo sistemų ir kalbų. Jos automatizuoja modelių sudarymo, programavimo ir derinimo procesus, sumažina reikalavimus modeliuotojų kvalifikacijai. Dalis jų pateikiama 1.1 lentelėje. Yra paskelbta daug darbų, kuriuose modeliavimo sistemos lyginamos pagal įvairius požymius: sprendžiamų uždavinių tipai, formalaus aprašymo metodai, modeliavimo greitis, reikalaujamas operatyvinės atminties dydis ir t.t. Pagal naudojamus sistemų formalizavimo metodus modeliavimo sistemas galima skirtyti į tris grupes:

1. Tolydaus modeliavimo kalbos, sistemų formalizavimui naudojamas diferencialinių lygčių aparatas (CSMP, DAS, SPAS, ANALOG); [3,7]
2. Diskretaus modeliavimo kalbos, sistemų formalizavimui naudojančios skaitinius metodus (DDL, LOGAL, ATLAS, SIMBOL, LOGSIM); [3,7]
3. Sistemos, naudojančios tranzaktus, daugiausia naudojamos masinio aptarnavimo sistemoms imituoti (GPSS, CELLSIM, CAPS, SIMDIS, GSIM) [1,3].

Imitavimo sistemų programavimo kalbų tikslas yra kuo patogiau aprašyti modeliavimo sistemą ir imitacinius eksperimentus. Sistemos imitavimas vykdomus pagal bendrus sistemos veikimo principus, kurie reikalauja specifinio įvykių deklaravimo. Į imitacinį modeliavimą orientuotos programavimo kalbos buvo pradėtos plėtoti kai tik atsirado aukštesnio lygio programavimo kalbos. Pavyzdžiui, SIMULA67 buvo plėtojama standartinės programavimo kalbos ALGOL60 bazėje.

DYNAMO, SIMULATE kalbos orientuotos imituoti dideles ekonomines sistemas su atsakomųjų ryšių mechanizmu, baigtinėmis lygčių sistemomis. Jos pritaikytos tiesinių sistemų imitaciniam modeliavimui. Šios kalbos yra STELLA kalbos, sukurtos Macintosh kompiuteriams, atitikmenys.

GSP (General Simulation Program) yra naudojama bendriems imitaciniams modeliams. Ji specializuota gamybinės produkcijos imitaciniam modeliavimui. Taip pat yra keletas šios kalbos atšakų: ESP ir CSL (Control & Simulation Language). Pastaroji yra panaši į SIMSCRPT. [3]

Norint imituoti tolydinių sistemų modelį, šie sprendimai netinka. Tam tinka diferencialinių lygčių modeliais paremtas CSMP paketas. Panašios programos yra TUTSIM arba ISIM.

Tiesiniam sistemų modeliavimui yra sukurta daug skirtingų kalbų, pvz. SPICE, MathLab, COL, Micro.Cap. [4]

1.1 lentelė.

Programinė įranga ir modeliai

Pavadinimas	OS sistemos tipas, kompiliatorius	Pastaba
Diskretiniai modeliai		
GPSS/PC	MS Windows	Interaktyvi grafinė sąsaja
GPSS/H	MS Windows	Supaprastinta GPSS versija IBM x486 PC
SIMSCRIPT II.5	MS Windows, MF	Parametriniui modeliavimui
SIMULA	MS Windows, MF	Parametriniui modeliavimui
SIM	MS Windows, PS/2	Grafinė sąsaja
PC-SOL 4.0	MS Windows	
Tolydiniai modeliai		
DYNAMO III /F+/ 370	MF, FORTRAN	Netiesiniai modeliai
MicroDYNAMO	MS Windows	
ProDYNAMO+	MS Windows	Yra modelių optimizavimo galimybės
CSSL-IV	MS Windows	
ISIM	MS Windows, MF	
ACSL	MS Windows, FORTRAN	
Tolydiniai-diskretiniai modeliai		
SLAM II/PC	MS Windows	
MicroPASSIM	MS Windows, PASCAL	Išplėstos statistinės analizės galimybės; orientuota į pramonines sistemas; grafinė vartotojo sąsaja
PASION	MS Windows, PASCAL	
SIMAN + Cinema	MS Windows	
Dinaminės sistemos		
LSMP	MS Windows	Netiesinėms dinaminėms sistemoms
DSL/VS	MF, FORTRAN	Tiesinėms dinaminėms sistemoms
Realaus laiko sistemos		
NET Real Time	MS Windows	Tolydieji modeliai, grafinė vartotojo sąsają.
Sistemos orientuotos į produkcijos patikimumą ir kokybės vertinimą		
GEM-II	MS Windows, FORTRAN-77	Tinklų modeliai
MAST	MS Windows, FORTRAN-77	Produkcijos kokybės analizė
SPAR	MS Windows	Įmonės augimo tikimybių skaičiavimas
XCELL+	MS Windows	Industrinių sistemų prototipų konstravimas
SIMIS III	MS Windows, MF	Pramoninių žaliavų srautų modeliavimas
Sistemos, modeliuojančios ekonominius procesus		
LIBRA	MS Windows	Ekonominių sistemų modeliavimas
MULTISIM	DEC-20, Simula	Rinkos modeliavimas
Statistinis sistemų modeliavimas ir duomenų apdorojimas		
СТАТМОД	MS Windows	Statistinis modeliavimas atsitiktinių dydžių, vektorių, procesų
СТАН	MS Windows	Statistinė eksperimento rezultatų analizė

Uždari programų paketai, tokie kaip GPSS ar SLAM II nėra tinkami, kai reikia specifinio imitavimo proceso aprašymo ar detalios analizės. [8]

Dažniausiai programų paketai yra orientuoti spręsti specifinių sričių problemas. Tik bendros paskirties kalba gali pasiūlyti lankstumą ir galimybę išspręsti daugelį suformuluotų uždavinių. Jie yra kompromisas tarp lankstumo ir paprastumo.[7]

Pastaruoju metu labai didelis dėmesys skiriamas efektyviai modeliavimo rezultatų analizei. Yra kuriamos statistinės analizės procedūrų bibliotekos, orientuotos specialiai imitacinio modeliavimo sistemoms, tokios kaip „Statistical Expert System for Simulation Analysis (SESSA)“ , kurios gali būti dažnai atnaujinamos, bet neturi jokio tiesioginio ryšio su pačiu imitaciniu modeliu, todėl apsunkina jų panaudojimą. [8]

Integruotos imitacinio modeliavimo rezultatų analizės priemonės, tokios kaip „Warwick Expert Simulation(WES)“ ir „Expert Post-processor for Simulation Output Analysis (EPSONA)“ susieja statistinės analizės procesą su imitacinio modelio kūrimo ir veikimo procesu, tačiau yra orientuotos į neturintį patirties statistinėje analizėje vartotoją ir yra imlios laiko atžvilgiu. Taipogi bandoma kurti apibendrintas imitacinio modelio ir proceso analizės schemas, išskaidant imitacinio modelio kūrimo ir modeliavimo procesus į atskiras dalis.

1.2. APTARNAVIMO SISTEMŲ MATEMATINIAI MODELIAI

1.2.1. ANALITINIAI MODELIAI

Terminas „Eilių teorija“ naudojamas nusakyti aptarnavimo sistemų analizinius metodus. Eilių teorija dažniausiai taikoma supaprastintiems realių procesų modeliams, kuriems galima užrašyti analizines matematines išraiškas [10].

Eilių teorijoje naudojamas universalus modelių žymėjimas *A/B/n/m*:

a) *A* nusako paraiškų atėjimo į sistemą procesą (laiko tarp gretimų paraiškų sraute pasiskirstymo dėsnį; M reiškia eksponentinį pasiskirstymo dėsnį; G – bendrąjį pasiskirstymo dėsnį, o D – determinuotą pasiskirstymo dėsnį);

b) *B* nurodo paraiškų aptarnavimo trukmės pasiskirstymo dėsnį, ji taip pat gali būti *M, D, G*;

c) *n* nusako paraiškas aptarnaujančių linijų skaičių, kuris gali kisti nuo 0 iki ∞;

d) *m* nusako maksimalų laukimo vietų eilėje skaičių. [12].

Yra daug aptarnavimo sistemų analizinių modelių, paremtų skirtingais eilės valdymo tikslais ir aptarnavimo sistemos sąlygomis.

Analiziniai matematiniai aptarnavimo sistemų modeliai taikomi, kai paraiškų srautas aptarnavimo sistemoje yra stacionarus Puasono procesas, paraiškų eilės prie aptarnavimo įrenginių yra neribotos ir aptarnavimas vyksta pagal taisyklę „pirma atėjo – pirma išėjo“.

Nors analizinėmis išraiškomis yra pagrindžiami stacionarių Markovo procesų pagrindinių klausimų atsakymai, tačiau analitiniai metodai yra bejėgiai sudarant didesnius modelius, kurie sudaryti iš daug aptarnavimo įrenginių. Be to, sistemai sudarytai pagal Markovo grandines, norint surasti pagrindinius rezultatus, reikia išspręsti tiesinių lygčių sistemą, kuri didėja, didėjant būsenų skaičiui. Bendru atveju, kai paraiškų/klientų srautas nėra Pausoninis (pvz. G/G/1), analiziniai modeliai yra sudėtingi, o lygčių sistemos dažnai neišsprendžiamos[2].

Minėtos prielaidos leidžia sudaryti tik paprastus modelius, modeliuoti atskiras sistemos, o ne visą kompleksinę aptarnavimo sistemą. Analizuojant aptarnavimo sistemas dažniausiai norima atsakyti į šiuos klausimus:

1. Koks bus paraiškų vidutinis aptarnavimo laikas?
2. Kokia dalis paraiškų bus aptarnauta?
3. Kiek vidutiniškai kanalų bus užimta?
4. Kokią laiko dalį sistema neturės darbo?
5. Kiek laiko paraiška lauks eilėje ir t.t?

Atlikus analizinius skaičiavimus, gaunami aptarnavimo sistemos kiekybiniai rodikliai, kuriuos galime naudoti sprendžiant optimizavimo uždavinius, pvz. optimizuoti aptarnaujančių kanalų kiekį, pasiūlyti optimalią aptarnavimo strategiją, minimizuoti paraiškų aptarnavimo laiką ir kt.

Imitacinis modeliavimas leidžia detaliau pažvelgti į aptarnavimo procesus, kurie negali būti išnagrinėti analitiniais metodais.

1.2.2. IMITACINIAI MODELIAI

Imitaciniai modeliai papildo aptarnavimo sistemų modelių tyrimo galimybes, analizinių rezultatų tikrinimą, pateikia vizualią informaciją, tačiau dauguma iš jų turi ribotas statistinės analizės galimybes.

Kiekvienos imitacinio modeliavimo sistemos bazę sudaro matematinė schema, kuri naudojama modeliuojamų objektų formaliam aprašymui[3]. Dažniausiai naudojamos matametinės schemos pateiktos 1.2 lentelėje.

Pagal imitacinių modelių sudarymo automatizacijos lygį modeliavimo sistemas galima suskirstyti į tris grupes[3]:

1. Bendros paskirties imitacinio modeliavimo kalbos, tokios kaip Simula, GPSS, GASP;
2. Problemiškai orientuotos specialios paskirties imitacinio modeliavimo sistemos, besiremiančios algoritminio aprašymo koncepcija, tokios kaip MPL/I-VS, OSSL, OASIS;
3. Deklaratyvinio tipo modeliavimo sistemos, tokios kaip SIMAS, ASIM, MAST, TOMAS.

1.2 lentelė.

Imitacijos būdai

Modelio tipas	Matematinė schema	Imitavimo priemonės
Diskretinis	Įvykių	SIMSCRIPT
	Darbų	CSL, ESCL, SIMON
	Procesų	SIMPULA, MPL/1
	Tranzaktų	GPSS, SIMDIS, SLAM-II
Tolydinis	Aproksimacija diferencialinėmis lygtimis	DYNAMO, MIMIC
Tolydinis-Diskretinis	Kombinuotas (įvairūs modeliavimo būdai yra galimi)	SLAM, НЕДИС
	Agregatinis	САПАС, PRANAS, SIMAS

Bendros paskirties modeliavimo sistemos paprastai turi savo formalią modeliavimo kalbą, kurios pagalba galima aprašyti ir formalizuoti sistemą bei jos funkcionavimo procesą, be to, realizuojamos pagrindinės imitaciniam modeliavimui reikalingos operacijos: įvykių kalendoriaus tvarkymas, modeliavimo eigos valdymas, statistinės informacijos apie sistemos funkcionavimą kaupimas ir analizė.[1]

Didžiausią ir greičiausiai gausėjančią imitacinio modeliavimo sistemų dalį sudaro 2 ir 3 grupės sistemos, nes jos būna pritaikytos konkrečiai probleminei sričiai ir paprastai turi bazinių modeliavimo elementų biblioteką, kuri leidžia daug greičiau aprašyti ir formalizuoti tos probleminės srities objektus ir sistemas. Tokios sistemos reikalauja mažiau sąnaudų jų kūrime, bet jų panaudojimas apsiriboja tam tikra problematine sritimi. Modelių kūrimo procesas, naudojant tokias sistemas, smarkiai pagreitėja.

Antro grupės modeliavimo sistemos modeliai rašomi sistemos programavimo kalba naudojant bazines bibliotekas ir procedūras, todėl iš vartotojo reikalauja programavimo įgūdžių. Trečios grupės modeliavimo sistemas paprastai sudaro keletas skirtingų modeliavimo klasių tipų, o modelis kuriamas modelio struktūrą aprašant nesudėtinga deklaratyvinio tipo kalba. Tokia modeliavimo sistema gali naudotis vartotojai neturintys programavimo įgūdžių.

Bendros paskirties imitacinio modeliavimo sistemose imitacinio modelio aprašymas ir statistinės analizės aprašymas yra kuriami vienu metu ir yra neatsiejami, todėl norint pakeisti statistinės analizės eigą ar duomenų rinkimą, reikia keisti visą modelį.

Įvairiose modeliavimo sistemose statistinės analizės proceso aprašymo sakinių sintaksė yra skirtinga, bet semantika panaši. Jų pagalba aprašomi objektai, kurie naudojami duomenų apie stebimus dydžius kaupimui ir nurodoma, kaip sukauptus duomenis apdoroti. Kiekvienam statistikos tipui yra savi matavimo sakiniai, kurie įterpiami tose programos vietose, kur keičiasi stebimojo kintamojo reikšmė. Modelio kūrimo etape reikia apibrėžti, kokius kintamuosius stebėsime, kaip apdorosime stebėjimo duomenis ir kokioje formoje išvesime rezultatus. Toks statistinės analizės proceso ir

imitacijos proceso suliejimas vienoje programoje gali būti pateisintas tik kuriant nesudėtingus, vienkartinio panaudojimo modelius. Jis nepriimtinas daugkartinio panaudojimo modeliuose, kur gali keistis ne tik stebimų dydžių aibė, bet ir tyrimo schema. Taip pat ji nepriimtina ir tuo atveju, kai viso modelio atskiros dalys naudojamos sudarant kitus modelius, kas yra būdinga agregatinei matematinei schemai. Pasikeitus stebimų dydžių aibei arba tyrimo schemai, reikia perprogramuoti modelį, o tai dažniausiai gali padaryti tik pats modelio kūrėjas (programuotojas). Tai apsunkina modelio arba jo dalių panaudojimą vartotojui, kuris nori tirti kitas modelio savybes, t.y. pakeisti analizės schemą, nekeisdamas imitacijos proceso aprašymo[6].

Dabar egzistuojančių modeliavimo sistemų statistinės analizės priemonės nėra pritaikytos darbui su šiuolaikinėmis statistinės analizės sistemomis (SAS, SPSS ir t.t.)

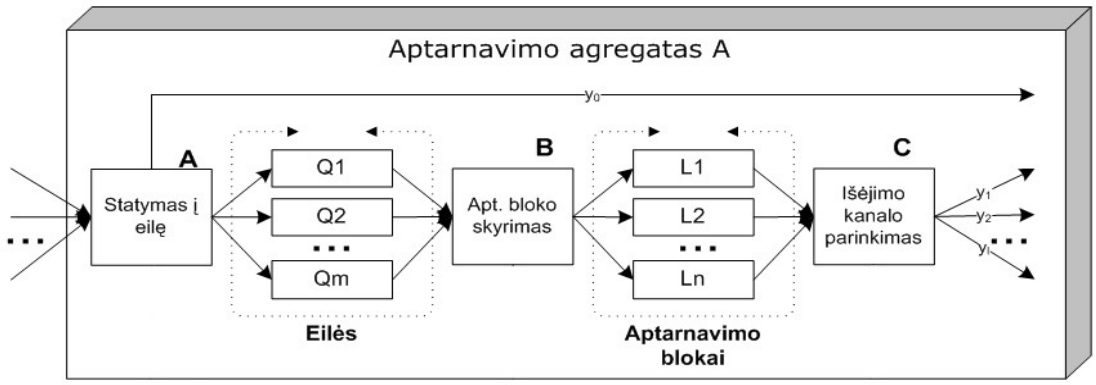
Imitacinio modeliavimo sistemose rezultatų statistinės analizės problema sprendžiama panaudojant standartines, integruotas statistinės analizės priemones. Tokios priemonės negali realizuoti visų šiuolaikinių statistinės analizės programų galimybių, todėl statistinės analizės galimybes nulemia pasirinktos imitacinio modeliavimo sistemos galimybės. Dažniausiai imitacinio modeliavimo sistemos pateikia tik skaitines charakteristikas, neatlieka duomenų kaupimo, kuris reikalingas taikant sudėtingesnius statistikos metodus. Kadangi dėl komercinių tikslų modeliavimo sistemos dažniausiai yra uždaros ir nepritaikytos vartotojui atlikti atliekamų modeliavimo matavimų kaupimo, todėl nėra pritaikytos darbui su išoriniais statistikos paketais.

Duomenų rinkimo ir pritaikymo statistinės analizės paketams, bei statistinės analizės modelių sudarymas yra problemos, kurias galima spręsti naudojant arba atviro kodo imitacinio modeliavimo sistemas, kurių yra mažai dėl komercinių tikslų, arba sukurti pačiam remiantis bendriausia agregatine matematine schema ir pritaikyti duomenų rinkimą konkrečiam statistinės analizės modeliui.

1.3. UNIVERSALUS APTARNAVIMO SISTEMŲ AGREGATINIS MATEMATINIS MODELIS

Panaudojus agregatinę matematinę schemą yra sukurtas universalus aptarnavimo sistemos agregatas, kuris gali būti naudojamas įvairios struktūros ir paskirties masinio aptarnavimo tinklų imitaciniam modeliavimui.

Universalus aptarnavimo agregatas sudarytas iš m eilių ir n aptarnavimo blokų, k įėjimų ir l išėjimų. Vienu iš agregato įėjimų atėjusios paraiškos yra statomos į vieną iš eilių, naudojant vieną iš siūlomų algoritmų. Jų pradinę padėtį eilėje nusako pasirinktas eilės sudarymo algoritmas. Atsiradus laisvai aptarnavimo linijai naudojant vieną iš siūlomų algoritmų, pasirenkama aptarnaujama eilė ir aptarnaujama paraiška iš eilės laisvajai aptarnavimo linijai. Aptarnavimo trukmė nusakoma skirstiniu iš siūlomų skirstinių aibės. Analogiškai, pagal vieną iš siūlomų schemų aptarnauta paraiška yra perduodama vienu iš galimų agregato išėjimų tolyn. Universalaus agregato schema pateikta 1.1 pav.



1.1 pav. Universalaus agregato schema

Matematinis modelis sudarytas panaudojant valdančiųjų sekų metodą [7]. Agregatą apibūdina tokios aibės: laiko momentų aibė T, įėjimo signalų aibė X, išėjimo signalų aibė Y, būsenų aibė Z, parametrų aibė B, įvykių aibė E, kontrolinių sumų aibė W, perėjimo operatorių aibe H ir išėjimų operatorių aibė G. Agregatinėje sistemoje perduodamos paraiškos struktūra: *KL={ID, P, N, TK, TP},* kur *ID* - identifikatorius, *P*- prioritetas, *N* - vardas, *TK* – sukūrimo laikas, *TP* - laikinų kintamųjų laukas.

1. *Laiko momentai:* $T = \{t_1, t_2, \ldots, t_m, \ldots\}$.
2. *Įėjimo signalai:* $X = \{X_1\}, \quad X_1 = (KL)$.
3. *Išėjimo signalai:* $Y = \{Y_0, Y_1, Y_2, \ldots, Y_l\}, \quad Y_i = (KL), i = \overline{0,l}$, kur

   l – išėjimo signalų (kanalų) kiekis.

   $Y_i = (KL), i = \overline{1,l}$ - paraiškos išėjimas iš agregato.

   Y_0 - neaptarnautų paraiškų išėjimas.
4. *Įvykių aibė:* $E = E' \cup E''$.

   $E' = \{e'_1, e'_2, \ldots, e'_k\}$ - išorinių įvykių aibė,

   $e'_i, i = \overline{1,k}$ - paraiškos atėjimas i-uoju kanalu.

   $E'' = \{e''_{1,j}\}_{j=1}^{N}$ - vidinių įvykių aibė, e''_j - paraiškos aptarnavimo pabaiga agregato linijoje.
5. *Parametrų aibė:* $B = \{L, M, N, S^A, S^L, S^Q, S_i^P, S^O, l_i^Q, F_j\}$, kur

   L – išėjimo kanalų kiekis,

   M – eilių kiekis agregate,

   N – aptarnavimo linijų kiekis,

   S^A - aptarnaujamos eilės pasirinkimo strategija,

   S^L - aptarnavimo linijos pasirinkimo strategija,

S^Q - eilės pasirinkimo atvykusiai paraiškai strategija,

S_i^P - paraiškos pasirinkimo aptarnavimui eilėje Q_i strategija,

S^O - išėjimo kanalo parinkimo strategija,

l_i^Q - maksimalus galimas eilės Q_i ilgis,

F_j - aptarnavimo trukmės linijoje L_j skirstinys.

$F_j = (F_v, f_1, f_2, \ldots, f_{k(v)})$, kur F_v - skirstinio tipas,

f_j, $j = \overline{1..k(v)}$ - skirstinio parametrai, *k(v)* – skirstinio parametrų skaičius.

6. *Agregato būsenų aibė:*

$Z = \{Q_A(t_m), Q_X(t_m), L_A(t_m), [Q_i(t_m)]_{i=\overline{1,M}}, [L_j(t_m), U_j(t_m), V_j(t_m)]_{j=\overline{1,N}}\}$, kur

$Q_A(t_m)$ - paskutinė pasirinkta aptarnavimui eilė,

$Q_X(t_m)$ - eilė, į kurią buvo paskirta paskutinė atėjusi paraiška,

$L_A(t_m)$ - aptarnavimo linija, į kurią buvo paskirta paskutinė atėjusi aptarnavimui paraiška,

$Q_i(t_m)$ - paraiškų kiekis eilėje Q_i, $i = \overline{1,M}$,

$L_j(t_m)$ - užimta (1) ar ne (0) nurodytu laiko momentu aptarnavimo linija $j = \overline{1,N}$,

$U_j(t_m)$ - linijos apkrautumas, $j = \overline{1,N}$,

$V_j(t_m)$ - paskutinio aptarnavimo pabaigos momentas, $j = \overline{1,N}$.

7. *Kontrolinės sumos:* $W = \{\{W(e_j'', t_m)\}_{j=1}^N, e_Q''\}$.

8. *Pradinės būsenos:* $Z = \{Q_A(t_0), Q_X(t_0), L_A(t_0), [Q_i(t_0)]_{i=\overline{1,M}}, [L_j(t_0), U_j(t_0), V_j(t_0)]_{j=\overline{1,N}}\}$,

$Q_A(t_0) = 0$, $Q_X(t_0) = 0$, $L_A(t_0) = 0$, $Q_i(t_0) = 0, i = \overline{1,M}$,

$L_j(t_0) = 0, U_j(t_0) = 0, V_j(t_0) = t_0, W(e_j'', t_0) = \infty, j = \overline{1,N}$.

9. *Perėjimo ir išėjimo operatoriai* H, G*:*

$H(e_1'), G(e_1')$: $KL_{TP} = t_m$ - laikiname kintamajame išsaugomas atėjimo į sistemą laikas.

a) $\exists j : L_j(t_m) = 0, j = \overline{1,N} \Rightarrow$ nėra eilių ir yra laisva aptarnavimo linija., todėl paraiška nukreipiama aptarnavimui.

$H(e_1')$:

$$L_A(t_{m+1}) = \begin{cases} \min\limits_j ind\left\{L_j(t_m) : L_j(t_m) = 0, j = \overline{L_A(t_m), M}, \overline{1, L_A(t_m)}\right\}, jei\ S^L = 1, \\ \min\limits_j ind\left\{L_j(t_m) : L_j(t_m) = 0, j = \overline{1, M}\right\}, jei\ S^L = 2, \\ ind \min\limits_j \left\{U_j(t_m) : L_j(t_m) = 0, j = \overline{1, M}\right\}, jei\ S^L = 3, \\ ind \min\limits_j \left\{V_j(t_m) : L_j(t_m) = 0, j = \overline{1, M}\right\}, jei\ S^L = 4, \\ ind \max\limits_j \left\{V_j(t_m) : L_j(t_m) = 0, j = \overline{1, M}\right\}, jei\ S^L = 5. \end{cases}$$

$$W(e_j'', t_{m+1}) = \begin{cases} t_m + GEN(F_j), jei\ L_A(t_{m+1}) = j, \\ W(e_j'', t_m), kitais\ atvejais. \end{cases} \quad j = \overline{1, N}, \text{čia}$$

$GEN(F_j)$ - atsitiktinių dydžių generatorius.

$$V_j(t_{m+1}) = \begin{cases} W(e_j'', t_{m+1}), jei\ L_A(t_{m+1}) = j, \\ V_j(t_m), kitais\ atvejais. \end{cases} \quad j = \overline{1, N}$$

$$U_j(t_{m+1}) = \begin{cases} U_j(t_m) + (W(e_j'', t_{m+1}) - t_m), jei\ L_A(t_{m+1}) = j, \\ U_j(t_{m+1}), kitais\ atvejais. \end{cases} \quad j = \overline{1, N}$$

b) $\left(\neg\exists j : L_j(t_m) = 0, j = \overline{1, N}\right) \cap \left(\exists i : Q_i(t_m) < l_i^Q, i = \overline{1, M}\right) \Rightarrow$ nėra laisvų aptarnavimo linijų, yra neužpildytų eilių, paraiška statoma į eilę.

$$H(e_1') : Q_X(t_{m+1}) = \begin{cases} ind \min\limits_i \left\{Q_i(t_m) : Q_i(t_m) < l_i^Q, i = \overline{1, .M}\right\}, jei\ S^Q = 1, \\ \min\limits_i ind\left\{Q_i(t_m) : Q_i(t_m) < l_i^Q, i = \overline{Q_X(t_m) + 1, M}, \overline{1, Q_X(t_m)}\right\}, jei\ S^Q = 2, \text{čia} \\ rand(\{(\tilde{p}_i, Q_i(t_m)) : Q_i(t_m) < l_i^Q, i = \overline{1, M}\}), jei\ S^Q = 3. \end{cases}$$

$rand(\{(\tilde{p}_i, Q_i(t_m)) : Q_i(t_m) < l_i^Q, i = \overline{1, M}\})$ - atsitiktinių indeksų generatorius, iš pagal sąlyga gautų porų parenkantis vieną, kurios indeksas ir yra rezultatas. Generatoriuje naudojamos modifikuotos tikimybės $\tilde{p}_i = p_i \left(\sum\limits_{k : Q_k(t_m) < l_k^Q, k = \overline{1, M}} p_k \right)^{-1}$

$$Q_i(t_{m+1}) = \begin{cases} Q_i(t_m) + 1, jei\ Q_X(t_{m+1}) = i \\ Q_i(t_m), kitais\ atvejais. \end{cases} \quad i = \overline{1, M}.$$

c) $\left(\neg\exists j : L_j(t_m) = 0, j = \overline{1, N}\right) \cap \left(\neg\exists i : Q_i(t_m) < l_i^Q, i = \overline{1, M}\right) \Rightarrow$ nėra laisvų aptarnavimo linijų, nėra neužpildytų eilių, todėl paraiška išeina iš agregato neaptarnauta per išėjimą Y_0.

$G(e_1') : y_0 = x_m = (KL).$

$H(e_n''), G(e_n''), n = \overline{1..N}:$

$H(e_n''):$

$$W(e_j'', t_{m+1}) = \begin{cases} \infty, jei\ n = j, \\ W(e_j'', t_m), kitais\ atvejais. \end{cases} \quad j = \overline{1, N}$$

$$L_j(t_{m+1}) = \begin{cases} 0, \ jei \ n = j, \\ L_j(t_{m+1}), \ kitais \ atvejais. \end{cases} \quad j = \overline{1, N}$$

$G(e_n'')$:

$$ind_y = \begin{cases} S_{on}^O, jei \ S_S^O = 2, \\ rand(\{(p_l) : l = \overline{1, L}\}), jei \ S_S^O = 1. \end{cases}$$

$$y_{ind_Y} = (KL).$$

$\exists i : Q_i(t_m) \neq 0, i = \overline{1, M} \Rightarrow$ yra netuščių eilių, pradėti aptarnavimą.

a) $jei \ (S_{S1}^A = 2) \cap (Q_{Q_A(t_m)} \neq 0) \Rightarrow Q_A(t_{m+1}) = Q_A(t_m).$

b) $jei \ (S_{S1}^A \neq 2) \cup (Q_{Q_A(t_m)} = 0) \Rightarrow$

$$Q_A(t_{m+1}) = \begin{cases} ind \max\limits_i \{Q_i(t_m) : Q_i(t_m) \neq 0, i = \overline{1, M}, jei \ S_{S2}^A = 1, \\ \min\limits_i ind\{Q_i(t_m) : Q_i(t_m) \neq 0, i = \overline{Q_A(t_{m+1}) + 1, M}, \overline{1, Q_A(t_{m+1})}\}, jei \ S_{S2}^A = 2, \\ \min\limits_i ind\{Q_i(t_m) : Q_i(t_m) \neq 0, i = \overline{1, M}\}, jei \ S_{S2}^A = 3, \\ rand(\{(\widetilde{p}_i, Q_i(t_m)) : Q_i(t_m) \neq 0, i = \overline{1, M}\}), jei \ S_{S2}^A = 4. \end{cases}$$ čia

$$\widetilde{p}_i = p_i \left(\sum_{k:Q_{ki}(t_m) \neq 0, k = \overline{1, M}} p_k \right)^{-1}.$$

$$Q_i(t_{m+1}) = \begin{cases} Q_i(t_m) - 1, jei \ Q_A(t_{m+1}) = i \\ Q_i(t_m), kitais \ atvejais. \end{cases} \quad i = \overline{1, M}$$

$$W(e_j'', t_{m+1}) = \begin{cases} t_m + GEN(F_j), \ jei \ L_A(t_{m+1}) = j, \\ W(e_j'', t_m), \ kitais \ atvejais. \end{cases} \quad j = \overline{1, N}$$

$$V_j(t_{m+1}) = \begin{cases} W(e_j'', t_{m+1}), \ jei \ n = j, \\ V_j(t_m), \ kitais \ atvejais. \end{cases} \quad j = \overline{1, N}$$

$$U_j(t_{m+1}) = \begin{cases} U_j(t_m) + (W(e_j'', t_{m+1}) - t_m), \ jei \ n = j, \\ U_j(t_{m+1}), \ kitais \ atvejais. \end{cases} \quad j = \overline{1, N}$$

1.4. IMITACINIŲ MODELIŲ STATISTINĖS ANALIZĖS PRIEMONIŲ TRŪKUMAI IR DARBO UŽDUOTIS

Imitacinio modeliavimo sistemos yra uždaro tipo programos, todėl norint tyrinėti specifinės problematikos uždavinį reikia išmanyti atitinkamos imitacinės sistemos veikimo schemą, algoritmus, taikomus matematinius modelius. Renkantis imitacinę modeliavimo sistemą reikia žinoti kokius statistinės analizės rezultatus apie stebimus dydžius gali pateikti pasirinktas modeliavimo programų paketas.

Daugelio imitacinio modeliavimo paketų modeliavimo rezultatų statistinės analizės apdorojimo priemonės yra primityvios. Dažnai nėra galimybės atlikti pasirinktų dydžių stebimų reikšmių kaupimo, kuris leistų panaudoti šiuolaikinius statistinės analizės paketus ir atlikti sudėtingesnę modelio statistinę analizę.

Pagrindiniai trūkumai yra šie:

- imitacinio modeliavimo sistemos dažniausiai yra uždaros, vartotojas negali išplėsti sistemos galimybių;
- labai stiprus ryšys tarp imitavimo ir statistinės analizės procesų.
- modeliavimo sistemų statistinės analizės posistemių programinės priemonės dažnai yra primityvios (jos dažniausiai realizuoja tik aprašomosios statistikos metodus);
- nėra galimybės kaupti pasrinktų dydžių stebėjimo reikšmes , kas leidžia panaudoti šiuolaikinius statistinės analizės paketus;

Darbo tikslas – išplėsti aptarnavimo sistemų agregatinių imitacinių modelių statistinės analizės galimybes.

Darbo užduotys:

1. Susipažinti su imitacinio modeliavimo sistemomis ir atlikti jų statistinės analizės priemonių analizę.
2. Susipažinti su agregatine matematine schema ir imitacinių modelių kūrimu.
3. Pasiūlyti universalų duomenų rinkimo modelį agregatinei modeliavimo sistemai ir modeliavimo duomenų matricos struktūrą.
4. Pasiūlyti modeliavimo duomenų matricos transformavimo į statistinės analizės matricas modelį įgalinantį atlikti statistinę duomenų analizę.
5. Sukurti aptarnavimo sistemų imitacinę agregatinę modeliavimo sistemą.
6. Sukurti statistinės analizės posistemę apjungiančią aptarnavimo sistemų imitacinio modeliavmo programą, statistinės analizės paketą ir universalų matematinį paketą.
7. Panaudojant sukurtą metodiką, modelius ir programines priemones, atlikti konkretaus aptarnavimo sistemos agregatinio imitacinio modelio statistinę analizę.

2. TIRIAMOJI DALIS

2.1. APTARNAVIMO SISTEMOS AGREGATAI

2.1.1. PARAIŠKŲ SRAUTO GENERATORIUS

Paraiškos – dinaminiai sistemos objektai. Jos juda sistemos kanalais iš vieno agregato į kitą ir jų struktūra turi aprėpti visus būtinus paraiškos identifikavimo elementus, kurie priklauso nuo imituojamos sistemos. Paraiškos struktūra nėra griežtai apibrėžta. Imitavimo sistemoje naudojamas toks paraiškos struktūros variantas:

KL = {ID, P, TK, D, GID},

čia ID – paraiškos unikalus numeris, P - paraiškos prioritetas, TK – paraiškos sukūrimo generatoriuje laikas, TP – darbinis kintamasis laikiniems duomenims apie jos buvimą sistemoje saugoti, GID – paraišką sugeneravusio generatoriaus kopijos numeris.

Paraiškų generavimo agregatas *G* naudojamas paraiškų srautui sistemoje generuoti. Parametrai:

1. *L* – išėjimo kanalų kiekis. *L = 1.*
2. *M* – eilių kiekis agregate. *M = 0.*
3. *N* – aptarnavimo linijų skaičius. *N = 0.*
4. *P* – kuriamai paraiškai suteikiamas prioritetas. *P = {1, 2, ..., 20}.*
5. F_j - laiko intervalų tarp paraiškų kūrimo atėjimo skirstinys.

$F_j = (F_v, f_1, f_2)$, kur F_v - skirstinio tipas:

a) $v = 1,\ F_1 \sim E(\lambda)$.

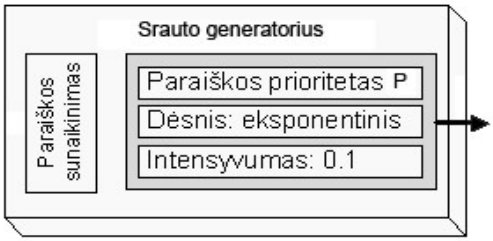
b) $v = 2,\ F_2 \sim N(\mu, \sigma^2)$.

c) $v = 3,\ F_3 \sim T(a, b)$.

d) $v = 4,\ F_4 \sim P(\lambda, k)$.

G srauto generatoriuje privalomai turi būti adresas į aptarnavimo agregatą, kitu atveju G generatorius nedirbs.

Prioritetas ir laiko intervalo skirstinys, bei jo parametrai, nustatomi vartotojo aprašant modelį. Srauto generatoriaus schema parodyta 2.1 paveiksle.



2.1 pav. Srauto generatoriaus schema

2.1.2. PARAIŠKŲ APTARNAVIMO AGREGATAS

Aptarnavimo agregatas A, kurio schema parodyta 1.7 paveiksle, yra modeliavimo sistemos dalis atliekanti imituojamą aptarnavimo sistemos darbą. Paraiškų aptarnavimo agregatas sukurtas remiantis universaliojo paraiškų aptarnavimo agregato schema.

Sistemoje SIMAS++ naudojamas agregatas turi tokius parametrus ir strategijas:

1. L – išėjimo kanalų kiekis. *L = {1, 2,...,20}*.
2. M – eilių kiekis agregate. *M = {1, 2,...,20}*.
3. N – aptarnavimo linijų kiekis, N = *{1, 2,...,20}*.
4. S^A - aptarnaujamos eilės pasirinkimo strategija.

   $S^A = (s_1, s_2, \{p_1, p_2, \ldots, p_M\}), s_1 \in \{1,2\}, s_2 \in \{1,2,3,4\}$, kur

   a) $s_1 = 1$ - aptarnaujama po vieną iš pasirinktos eilės,

   b) $s_1 = 2$ - pasirinkta eilė aptarnaujama tol, kol baigias,

   c) $s_2 = 1$ - pasirenkama ilgiausia eilė,

   d) $s_2 = 2$ - eilės aptarnaujamos ciklišk ai,

   e) $s_2 = 3$ - pasirenkama pirma netuščia eilė nuo pradžių,

   f) $s_2 = 4$ - atsitiktinai, naudojant lygias arba nurodytas $\{p_1, p_2, \ldots, p_M\}$ tikimybes.

5. S^L - aptarnavimo linijos pasirinkimo strategija.

   $S^L = (s), s \in \{1,2,3,4,5\}$, kur

   a) $s = 1$ - sekanti neužimta linija po paskutinės pasirinktos,

   b) $s = 2$ - pirmoji neužimta linija,

   c) $s = 3$ - pagal minimalų vidutinį apkrautumą,

   d) $s = 4$ - pagal ilgiausią paskutinės prastovos trukmę,

   e) $s = 5$ - pagal trumpiausios paskutinės prastovos trukmę.

6. S^Q - eilės pasirinkimo atvykusiai paraiškai strategija.

   $S^Q = \{s, \{p_1, p_2, \ldots, p_M\}), \ s \in \{1,2,3\}$, kur

   a) $s = 1$ - trumpiausia,

   b) $s = 2$ - ciklišk ai,

   c) $s = 3$ - atsitiktinai, naudojant lygias arba nurodytas $\{p_1, p_2, \ldots, p_M\}$ tikimybes.

7. S_i^P - paraiškos pasirinkimo aptarnavimui eilėje Q_i strategija.

   $S_i^P = \{s\}, s \in \{FIFO, LIFO, PRMAX\}$, kur

   a) $s = FIFO$ - aptarnavimui pasirenkama pirmiausia atėjusi į eilę paraiška,

b) $s = LIFO$ - aptarnavimui pasirenkama vėliausiai atėjusi į eilę paraiška,

c) $s = PRMAX$ - aptarnavimui pasirenkama pirma su didžiausiu prioritetu paraiška.

8. S^O - išėjimo kanalo parinkimo strategija.

$S^O = (s, \{p_1, p_2, \ldots p_l\}, \{o_1, o_2, \ldots o_M\})$, $s \in \{1,2\}$, kur

a) $s = 1$ - atsitiktinai, naudojant lygias arba nurodytas: $\{p_1, p_2, \ldots, p_N\}$ tikimybes,

b) $s = 2$ - pasirenkamas aptarnavimo liniją atitinkantis kanalas $\{o_1, o_2, \ldots o_N\}$.

9. l_i^Q - maksimalus galimas eilės Q_i ilgis.

$l_i^Q = \{0, 1, ..., \infty\}$

10. F_j - aptarnavimo trukmės linijoje L_j skirstinys.

$F_j = (F_v, f_1, f_2)$, kur F_v - skirstinio tipas:

e) $v = 1$, $F_1 \sim E(\lambda)$.

f) $v = 2$, $F_2 \sim N(\mu, \sigma^2)$.

g) $v = 3$, $F_3 \sim T(a, b)$.

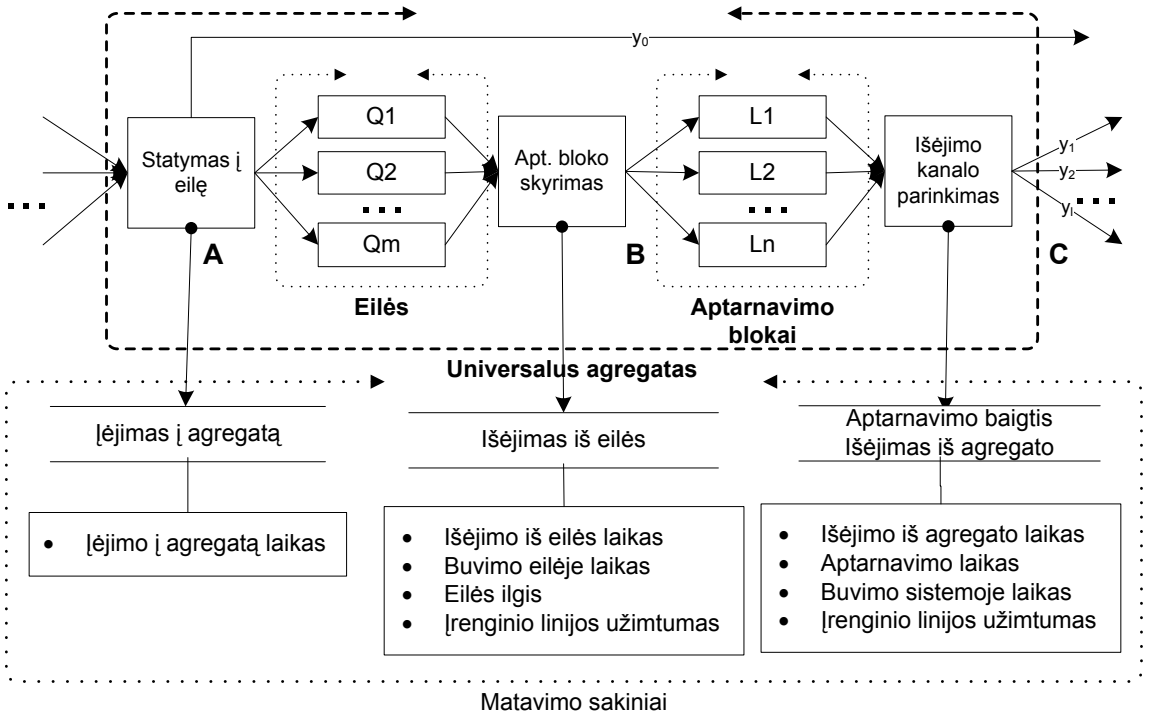
h) $v = 4$, $F_4 \sim P(\lambda, k)$.

2.2. APTARNAVIMO SISTEMOS MODELIAVIMO DUOMENŲ MATRICOS SUDARYMO MODELIS

Duomenų surinkimas ir kaupimas yra pirmasis žingsnis imitacinio modelio analizės procese. Imitaciniame modelyje būtina numatyti galimybę rinkti informaciją apie imituojamos sistemos arba jos atskirų dalių funkcionavimą. Dažnai duomenų saugojimo forma, kuri yra patogi duomenų rinkimui, būna nepriimtina informacijos statistiniam apdorojimui, todėl po eksperimento atlikimo reikia transformuoti gautą duomenų matricą į kitą, duomenų analizės etapui patogesnę formą, ir tik po to atlikti statistinę analizę.

2.2.1. UNIVARSALUS MATAVIMO OPERATORIUS

Suteikiant modelio kūrėjui galimybę apibrėžti stebimų kintamųjų aibę, o modelio vartotojui galimybę išskirti stebimų kintamųjų poaibį, surinkti apie juos duomenis ir atlikti statistinę analizę, nekeičiant imitacinio modelio programos, galima sumažinti ryšį tarp imitacijos ir statistinės analizės procesų aprašymų. Tai galima agregatiniuose modeliuose atlikti įterpiant valdomus matavimo operatorius į agregatų imitacijos proceso aprašymus.



2.2 pav. Paraiškų aptarnavimo agregato schema

Siūloma stebimų dydžių matavimo operatoriaus $\Gamma(e_m)$ struktūra:

$$\Gamma(e_m): Matavimas(U, K, A, N, V, Z, P), e_m \in E,$$

čia, U – valdantis kintamasis, K – agregato, kuriame vykdomas matavimas, vardas, A – agregato indeksas (kopijos numeris), N – eilės/aptarnavimo linijos numeris, V – stebimas kintamojo vardas (bendru atveju gali būti su indeksais), Z – stebimo kintamojo reikšmė, P – stebimo kintamojo žymė (label).

Matavimo objektą pilnai identifikuoja vektorius *(U, K, A, N, V)*. Jeigu vektorių pažymėsime Φ, tuomet matavimo sakinį galime užrašyti:

$$\Gamma(e_m): Matavimas(\Phi, Z, P), e_m \in E$$

Agregatinio matematinio modelio papildymas modeliavimo duomenų rinkimui ir kaupimui. Duomenų objektai statistinei informacijai kaupti: $\Phi = \left\{\Phi_r = (U, K, A, N, V), r = \overline{1,7}\right\}$, čia U- valdantis kintamasis; K - agregato, kuriame vykdomas matavimas, vardas; A – agregato indeksas (kopijos numeris); N – eilės arba aptarnavimo linijos numeris, V - stebimo dydžio vardas (bendru atveju gali būti su indeksais). Objektai:

Φ_1 - eilės ilgis,

Φ_2 - išėjimo iš eilės laikas,

Φ_3 - buvimo eilėje laikas,

Φ_4 - aptarnavimo linijos užėmimas,

Φ_5 - buvimo agregate laikas,

Φ_6 - buvimo sistemoje laikas,

Φ_7 - neaptarnautų paraiškų kiekis.

Statistikos rinkimo operatorių aibė: $\Gamma = \{\Gamma_r, r = 1,2\}$.

$\Gamma(e_1')$:

a) $\exists j : L_j(t_m) = 0, j = \overline{1,N} \Rightarrow$ į agregatą atėjusi nauja paraiška iškart nukreipiama aptarnavimui:

$$\Gamma(e_1') : \begin{cases} Matavimas(\Phi_1, 0, KL_P), \\ Matavimas(\Phi_2, t_m, KL_P), \\ Matavimas(\Phi_3, 0, KL_P), \\ Matavimas(\Phi_4, 0, KL_P). \end{cases}$$

b) $\left(\neg\exists j : L_j(t_m) = 0, j = \overline{1,N}\right) \cap \left(\exists i : Q_i(t_m) < l_i^Q, i = \overline{1,M}\right) \Rightarrow$ į agregatą atėjusi nauja paraiška statoma į eilę:

$\Gamma(e_1') : Matavimas(\Phi_1, Q_i(t_m), KL_P)$.

c) $\left(\neg\exists j : L_j(t_m) = 0, j = \overline{1,N}\right) \cap \left(\neg\exists i : Q_i(t_m) < l_i^Q, i = \overline{1,M}\right) \Rightarrow$ į agregatą atėjusi nauja paraiška neaptarnauta išeina per išėjimą Y_0:

$\Gamma(e_1') : Matavimas(\Phi_7, l_n, KL_P)$.

$\Gamma(e_n''), n = \overline{1,N}$:

a) e_j'' - paraiškos aptarnavimo pabaiga agregato *j*-oje linijoje

$$\Gamma(e_n'') : \begin{cases} Matavimas(\Phi_4, 0, KL_P), \\ Matavimas(\Phi_5, t_m, KL_P), \\ Matavimas(\Phi_5, t_m - KL_{TP}, KL_P), \\ Matavimas(\Phi_6, t_m - KL_{TK}, KL_P). \end{cases}$$

$\exists i : Q_i(t_m) \neq 0, i = \overline{1,M} \Rightarrow$ yra netuščių eilių, paimta iš eilės paraiška pradedama aptarnauti.

$$\Gamma(e_n'') : \begin{cases} Matavimas(\Phi_2, t_m, KL_P), \\ Matavimas(\Phi_3, t_m - KL_{TP}, KL_P), \\ Matavimas(\Phi_4, Q_i(t_m), KL_P), \\ Matavimas(\Phi_4, 1, KL_P), \\ Matavimas(\Phi_5, L_A(t_{m+1}), KL_P). \end{cases}$$

Matavimo rezultatai eksperimento metu gaunami tokia seka, kokia vyksta įvykiai agregatuose ir yra nerūšiuoti pagal objektus, kurių parametrai ar charakteristikos yra matuojamos. Tai nesukelia problemų jei reikia gauti skaitines modelio charakteristikas, tikrinti paprasčiausias hipotezes ar matavimo kintamojo reikšmę yra konkretaus statinio objekto (tarkime agregato) charakteristiką. Tačiau statistinei analizei dažnai yra svarbios ne atskiros objektų charakteristikos, bet tokių charakteristikų aibė, nes tada galima atlikti statistinę analizę ir gauti išvadas apie vieno charakteristikų poaibio priklausomybę nuo kito charakteristikų poaibio. Gauti matavimo rezultatai turi būti grupuojami pagal objektus, kuriems jie buvo atlikti.

2.2.2. MODELIAVIMO DUOMENŲ MATRICA

Statistiniai paketai dažniau reikalauja duomenis pateikti tokia forma: kintamieji, pagal kuriuos renkami duomenys turi sudaryti atskirus stulpelius, o eilutė yra sudaroma iš vieno objekto stebėjimo rezultatų. Tokia forma pateikti duomenys leidžia daug paprasčiau realizuoti sudėtingesnius statistinės analizės būdus, todėl iš anksto būtina numatyti modeliavimo duomenų matricos formą, pagal kurią bus formuojami ir pateikiami modeliavimo duomenys išoriniams statistinės analizės paketams. Galimas įvairus objekto savybių išdėstymas matricoje, kurios eilutė yra stebėjimo objektas, o stulpeliai – stebimo objekto kintamieji. Siūloma matricos forma pateikta 2.1 lentelėje.

2.1 lentelė.

Kintamieji ir jų žymėjimas

Kintamieji	**Žymėjimas**
Paraiškos ID	ID_P
Generatoriaus ID	ID_G
Sisteminis laikas	T
Stebimas dydis	V
Agregato indeksas	A
Įrenginys/eilė	N
Prioritetas	P
Reikšmė	Z

- Paraiškos ID – unikalus paraiškos numeris, kuris priskiriamas sukuriant paraišką.
- Generatoriaus ID – paraišką sukūrusio srauto generatoriaus numeris.
- Sisteminis laikas – laikas, kada daromas matavimas.
- Stebimas dydis – stebimo dydžio vardas.
- Agregato indeksas – agregato numeris, kuriame įvyksta įvykis.
- Įrenginys/eilė – aptarnavimo linija arba eilė (priklausomai nuo parinktos stebimos statistikos), kurioje įvyksta įvykis.

- Prioritetas – paraiškos prioritetas, kuris suteikiamas kuriant paraišką.
- Reikšmė – matuojama reikšmė (laikai, eilės ilgiai, aptarnavimo įrenginio užėmimas).

Eksperimento metu matavimo operatoriaus $\Gamma(e_m)$ renkami duomenys formuoja modeliavimo duomenų matricą R, kuri yra būtina norint realizuoti detalesnę statistinę analizę su išoriniais statistiniais paketais. Sistema imituodama aptarnavimo sistemos modelio darbą gauna užsakytų duomenų dydžius. Atitinkamų stebimų dydžių reikšmes sistema surašo į modeliavimo duomenų matricą.

Modeliavimo duomenų matricos R pavyzdys pateiktas 2.2 lentelėje.

2.2 lentelė.

Modeliavimo duomenų matricos pavyzdys.

Paraiškos ID	Generatoriaus ID	Sisteminis laikas	Stebimas dydis	Agregato indeksas	Įrenginys /eilė	Prioritetas	Reikšmė
1580	1	528.2971734	T_QU	1	0	2	0
1580	1	528.2971734	A_BU	1	0	2	1
1580	1	528.2971734	Q_Q	1	0	2	0
1577	1	528.3829185	A_BU	4	0	2	0
1577	1	528.3829185	EX_A	4	0	2	0.522696793
1577	1	528.3829185	EX_SA	4	0	2	1.015065193
1577	1	528.3829185	EX_S	4	0	2	1.015065193
1578	2	528.7492808	A_BU	2	0	3	0
1578	2	528.7492808	EX_A	2	0	3	0.491463482
1578	2	528.7492808	EX_SA	2	0	3	0.491463482
1578	2	528.7492808	Q_Q	5	0	3	1
1578	2	528.7492808	T_QU	5	0	3	0
1578	2	528.7492808	A_BU	5	0	3	1
1578	2	528.7492808	Q_Q	5	0	3	0
1579	0	528.8016582	A_BU	0	0	1	0
1579	0	528.8016582	EX_A	0	0	1	0.509989917
1579	0	528.8016582	EX_SA	0	0	1	0.509989917

2.2.3. DUOMENŲ MATRICOS TRANSFORMAVIMAS Į STATISTINĖS ANALIZĖS DUOMENŲ MATRICĄ

Atliekant statistinę analizę, dalis duomenų, sukauptų modeliavimo duomenų matricoje R yra pertekliniai. Tuomet naudojantis reliacinės algebros operacijomis (išrinkimo ir projekcijos pagal atributus) galime suformuoti duomenų matricą, kuri reikalinga konkrečiai statistinei analizei atlikti.

Pateiksime pasiūlytus metodus, kurie panaudojant realiacinės algebros operacijas transformuoja modeliavimo duomenų matricą R į statistinės analizės duomenų matricas D_i, $i=\overline{1,5}$.

1. Metodas transformuoja modeliavimo duomenų matricą R į D_1. Atrenka duomenis pagal nurodytą stebimą dydį (*SD*), aptarnavimo įrenginį (*A*), aptarnavimo liniją arba eilę (*I*):

$$\boldsymbol{D_1 = \pi_{\{T,R\}}\left(\sigma_{\{SD=z,A=x,I=y\}}(R)\right)}.$$

2. Metodas transformuoja modeliavimo duomenų matricą R į D_2. Atrenka duomenis pagal nurodytą stebimą dydį (*SD*):

$$\boldsymbol{D_2 = \pi_{\{T,P,R\}}\left(\sigma_{\{SD=x\}}(R)\right)}.$$

3. Metodas transformuoja modeliavimo duomenų matricą R į D_3. Atrenka duomenis pagal pasirinktus aptarnavimo įrenginius (A) ir aptarnavimo linijas arba eiles (I):

$$\boldsymbol{D_3 = \pi_{\{T,R\}}\left(\sigma_{\{A=x,I=y\}}(R)\right)}.$$

4. Metodas transformuoja modeliavimo duomenų matricą R į D_4. Atrenka modeliavimo duomenų matricos (R) eilutes pagal nurodytą stebimą dydį (SD), aptarnavimo įrenginį (A) ir paraiškos prioritetą (P):

$$\boldsymbol{D_4 = \pi_{\{T,R\}}\left(\sigma_{\{SD=x,A=y,P=z\}}(R)\right)}.$$

5. Metodas transformuoja modeliavimo duomenų matricą R į D_5. Atrenka modeliavimo duomenų matricos (R) eilutes pagal nurodytą stebimą dydį (SD), aptarnavimo įrenginį (A), aptarnavimo liniją arba eilę (I) ir paraiškos prioritetą (P):

$$\boldsymbol{D_5 = \pi_{\{T,R\}}\left(\sigma_{\{SD=x,A=y,I=z,P=w\}}(R)\right)}.$$

Sukurtos duomenų matricos transformacijos D_1, D_2, D_3, D_4, D_5 taikomos priklausomai nuo pasirinkto stebimo dydžio ir užsakytos stebimo dydžio statistinės analizės. Dirbant ne su visa modeliavimo duomenų matrica R, o suskaidant į statistinei analizei tinkamas matricas, leidžia neatlikinėti perteklinių duomenų atrinkimo ir skaičiavimo veiksmų.

Keletas pavyzdžių kaip panaudojamas duomenų matricų transformacijos matricos:

1. Paraiškų su *z* prioritetu laikas praleistas *x* agregate:

$$\boldsymbol{D_5 = \pi_{\{T,R\}}\left(\sigma_{\{SD=T_{QU},A=x,P=z\}}(R)\right)}.$$

2. Paraiškų laikas praleistas agregate *x, y* aptarnavimo linijoje:

$$\boldsymbol{D_1 = \pi_{\{T,R\}}\left(\sigma_{\{SD=EX_A,A=x,I=y\}}(R)\right)}.$$

3. Paraiškų buvimo sistemoje laikas.

$$\boldsymbol{D_2 = \pi_{\{ID_P,R\}}\left(\sigma_{SD=EX_S}(R)\right).}$$

4. Eilės *y* ilgis, *x* agregate.

$$\boldsymbol{D_1 = \pi_{\{T,R\}}\left(\sigma_{\{SD=Q_Q,A=x,I=y\}}(R)\right)}$$

Šios modeliavimo duomenų matricos transformacijos gali būti suprogramuotos ir pritaikytos daugumoje matematikos ir statistinės analizės paketų. Šiame darbe transformacijos buvo realizuotos universaliame matematikos pakete MathCad (žr. 2.3.3.1. skyrelyje) ir statistinės analizės pakete SAS.

2.3. APTARNAVIMO SISTEMŲ AGREGATINIŲ IMITACINIŲ MODELIŲ STATISTINĖS ANALIZĖS POSISTEMĖ

2.3.1. POSISTEMĖS STRUKTŪRA

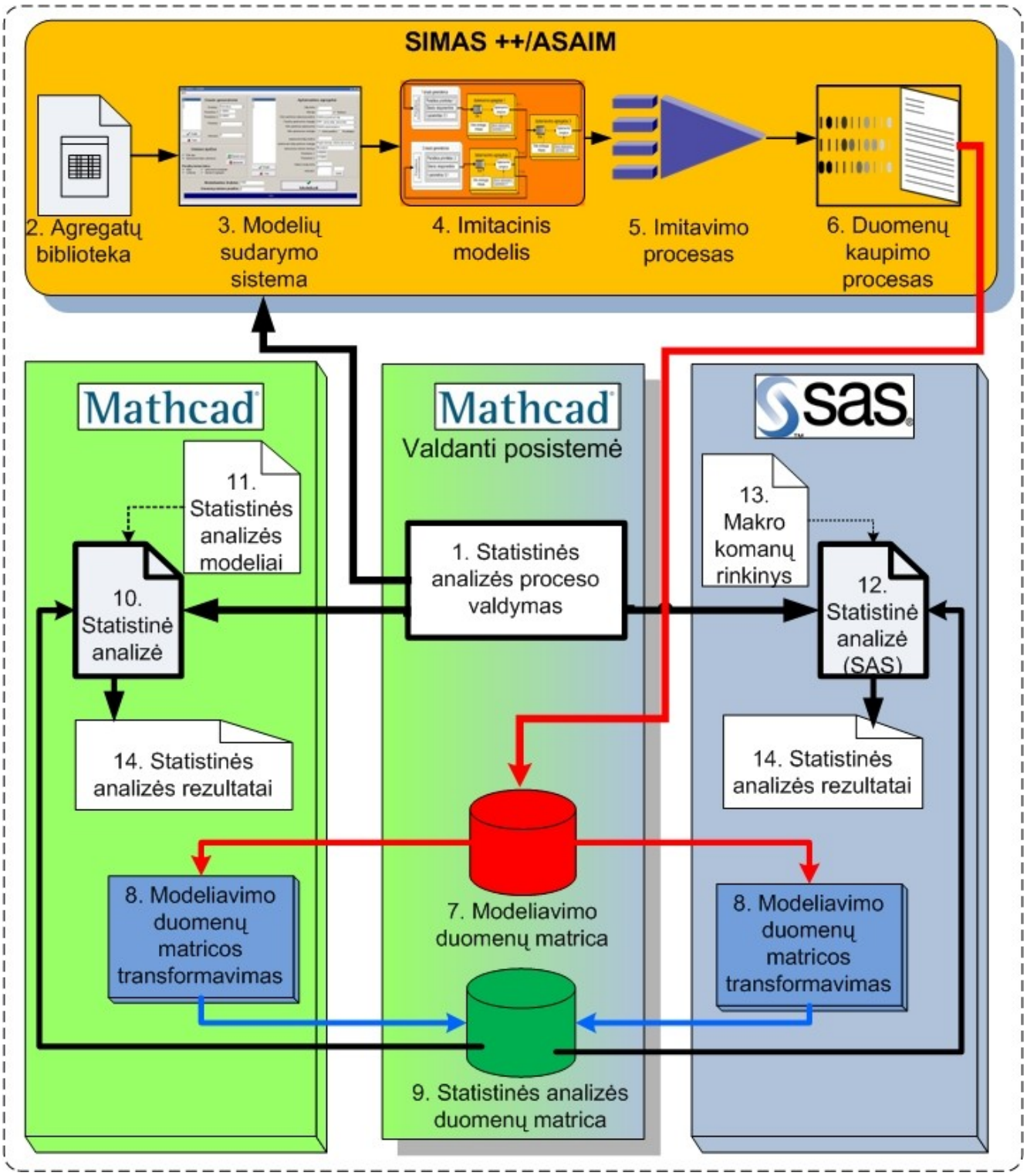
Norint taikyti modeliavimo duomenų analizei sudėtingesnius matematikos metodus reikia sukurti modeliavimo sistemą (žr. 2.3.2 skyrelį) ir ją apjungti su šiuolaikiniais matematikos paketais. Integruojant modeliavimo sistemą su matematikos ir statistinės analizės paketais reikia užtikrinti:

1) aptarnavimo sistemos modelių kūrimą ir ir redagavimą;
2) modeliavimo duomenų rinkimą;
3) vartotojo sukurto statistinio analizės modelio realizavimą;
4) matematinių išraiškų užrašymo patogumą;
5) tarpinių rezultatų pateikimą ir grafinį vaizdavimą.

Kadangi, matematinis paketas MathCad plačiai naudojamas visame pasaulyje dėl matematinių išraiškų formuluočių užrašymo, pagrindinių matematinių uždavinių sprendimo, programavimo galimybių, aprašomosios statistinės analizės paketo, nuspręsta pasirinktį šį matematinį paketą. [17]

Kadangi MathCad yra universalus matematikos paketas ir jame realizuota tik dažniausiai naudojami klasikiniai statistikos metodai, todėl siekiant užtikrinti sudėtingesnių modeliavimo duomenų statistinę analizę papildomai buvo pasirinktas statistinės analizės paketas SAS.

Naudojant MathCad teikiamas galimybės, sukurtas valdantis modulis, kuris apjungia modeliavimo sistemą SIMAS++/ASAIM ir paketus MathCAD bei SAS. Posistemės struktūra pateikta 2.3 paveiksle.



2.3 pav. Statistinės analizės posistemės struktūra

Paaiškinsime 2.3 paveiksle pateiktos statistinės analizės posistemės struktūrą:

1. Statistinės analizės proceso modelis. Valdantysis modulis inicijuoja modeliavimo sistemos SIMAS++/ASAIM iškvietimą, statistinės analizės atlikimą sistemose MathCad ir SAS.
2. Agregatų biblioteka. Joje saugomi vartotojo sukurti agregatai, kurie naudojami sudaryti ASAIM modelius.

3. Aptarnavimo sistemų modelių sudarymo sistema. Vartotojo valdoma dialoginė vartotojo sąsaja naudojama modelių kūrimui.
4. Imitacinis modelis. Pagal vartotojo užsakymą sukurtas ASAIM modelis.
5. Imitavimo procesas. Sukurto ASAIM modelio imitacinis modeliavimas.
6. Duomenų kaupimo procesas. Imitavimo proces metu atliekamas užsakytų modeliavimo duomenų kaupimas.
7. Modeliavimo duomenų matrica. SIMAS++/ASAIM pateikta modeliavimo duomenų matricą.
8. Modeliavimo duomenų matricos transformavimas. Modeliavimo duomenų matricos transformavimas į statistinės analizės duomenų matricas.
9. Statistinės analizės duomenų matrica. Matrica kuriai atliekama statistinė analizė.
10. MathCad atliekama statistinė analizė pagal realizuotus statistinės analizės modelius.
11. Statistinės analizės modeliai realizuoti MathCad sistemoje.
12. SAS atliekama statistinė analizė pagal aprašytus statistinės analizės modelius.
13. Makro komandų rinkinys.SAS makro komandos, kurios atlieka statistinę analizę.
14. Statistinės analizės rezultatai. MathCad sistemos ir/arba paketo SAS pateikiami statistinės analizės rezultatai.

Nors galimas platus SIMAS++/ASAIM modeliavimo duomenų pritaikymas statistinei analizei, šiame darbe rekomenduojama posistemė, kuri apjungia matematinį paketą MathCad, modeliavimo sistemą SIMAS++/ASAIM ir statistinės analizės paketą SAS.

2.3.2. AGREGATINIŲ IMITACINIŲ MODELIŲ SUDARYMO PRIEMONĖ SIMAS++/ASAIM

Sukurta aptarnavimo sistemų agregatinių imitacinių modelių modeliavimo programa SIMAS++/ASAIM (realizuota C++), kurios modeliavimas pagrįstas agregatine matematine schema. Sukurti du agregatai – paraiškų srauto generatorius (žr. 2.1.1 skyrelyje) ir paraiškų aptarnavimo agregatas (žr. 2.1.2 skyrelyje), kurie naudojami sudarytant įvairius aptarnavimo sistemų imitacinius modelius.

Imitacinio eksperimento planavimo etape nurodomi: imitacinio modelio parametrai, eksperimento vykdymo scenarijaus aprašymas ir modelio statistinei analizei reikalingų stebimų dydžių matavimų užsakymas.

Sistema gali atlikti tokius veiksmus:

- sukurti naują modelį;
- redaguoti sukurtą modelį (užduoties failas (*.tsk));

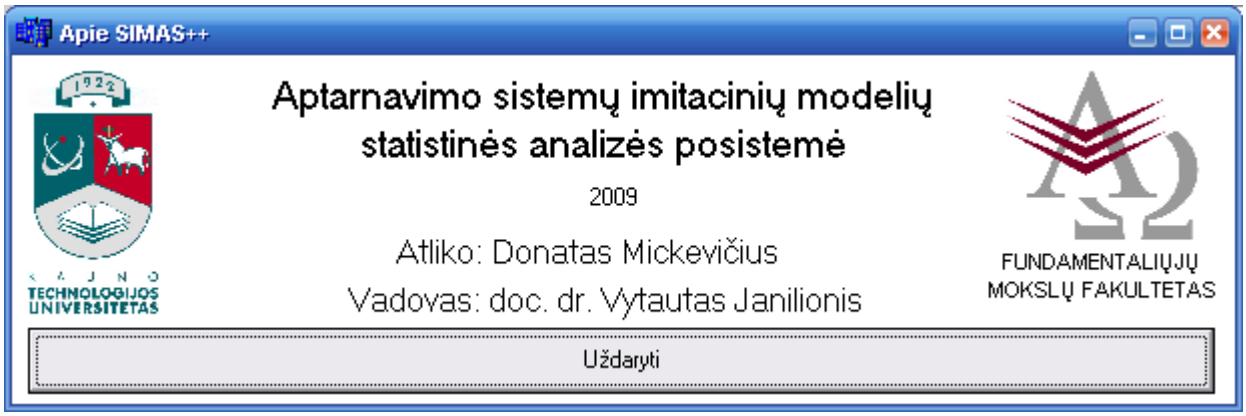
- nustatyti modelio paraiškų srauto generatoriaus parametrus:
    - intervalo tarp paraiškų skirstinį ir jo parametrą(-us);
    - paraiškos prioritetą;
    - srauto generatoriaus ryšį su kitu agregatu;
- nustatyti modelio aptarnavimo agregatų parametrus:
    - eilių kiekį;
    - eilių ilgį;
    - eilės parinkimo strategiją atėjusiai paraiškai;
    - paraiškų aptarnavimo strategiją;
    - eilės parinkimo aptarnavimui strategiją;
    - eilės aptarnavimo strategiją;
    - aptarnavimo kanalų kiekį;
    - paraiškos aptarnavimo laiko skirstinį, bei jo parametrus;
    - agregato ryšius su kitais agregatais;
- užsakyti modeliavimo duomenų rinkimą, kurie bus kaupiami modeliavimo duomenų matricoje. Duomenis galima kaupti apie
    - eilės ilgį;
    - įrenginio aptarnavimo linijų užimtumą;
    - paraiškų buvimo laiką eilėje;
    - paraiškų buvimo laiką aptarnavimo įrenginyje;
    - paraiškų buvimo laiką sistemoje;
    - paraiškų buvimo laiką sistemoje išeinant iš atitinkamo įrenginio;
- nurodyti duomenų rinkimo pradžios laiką;
- nurodyti modeliavimo trukmę.

Siekiant suteikti galimybė atlikti pakartotiną modelio tyrimą, modeliavimo sistemoje SIMAS++/ASAIM sukurti imitaciniai modeliai gali būti redaguojami ir saugojami pasirinktame kataloge. Modelio sudarymo ir imitacinio eksperimento vykdymo užduotis saugoma tekstinėje byloje su plėtiniu „.tsk“ Saugomos užduoties modelio sudarymui aprašo struktūra:

1. Aptarnavimo agregatų skaičius.
2. Srauto generatorių skaičius.
3. Modeliavimo trukmė.
4. Srauto generatorių sąrašas su parametrais.
5. Aptarnavimo agregatų sąrašas su parametrais.

Konkretaus imitacinio modelio aprašyto 2.4.1.1. skyrelyje saugojamo failo tekstas pateikiamas 1 priede.

Vykdant programą, pasirodo titulinis langas, kurioje nurodoma magistrinio darbo tema, autorius, vadovas, metai (2.4 pav.). Modelio sudarymui naudojama dialoginė vartotojo sąsaja (2.5 pav.).



2.4 pav. SIMAS++/ASAIM titulinis langas

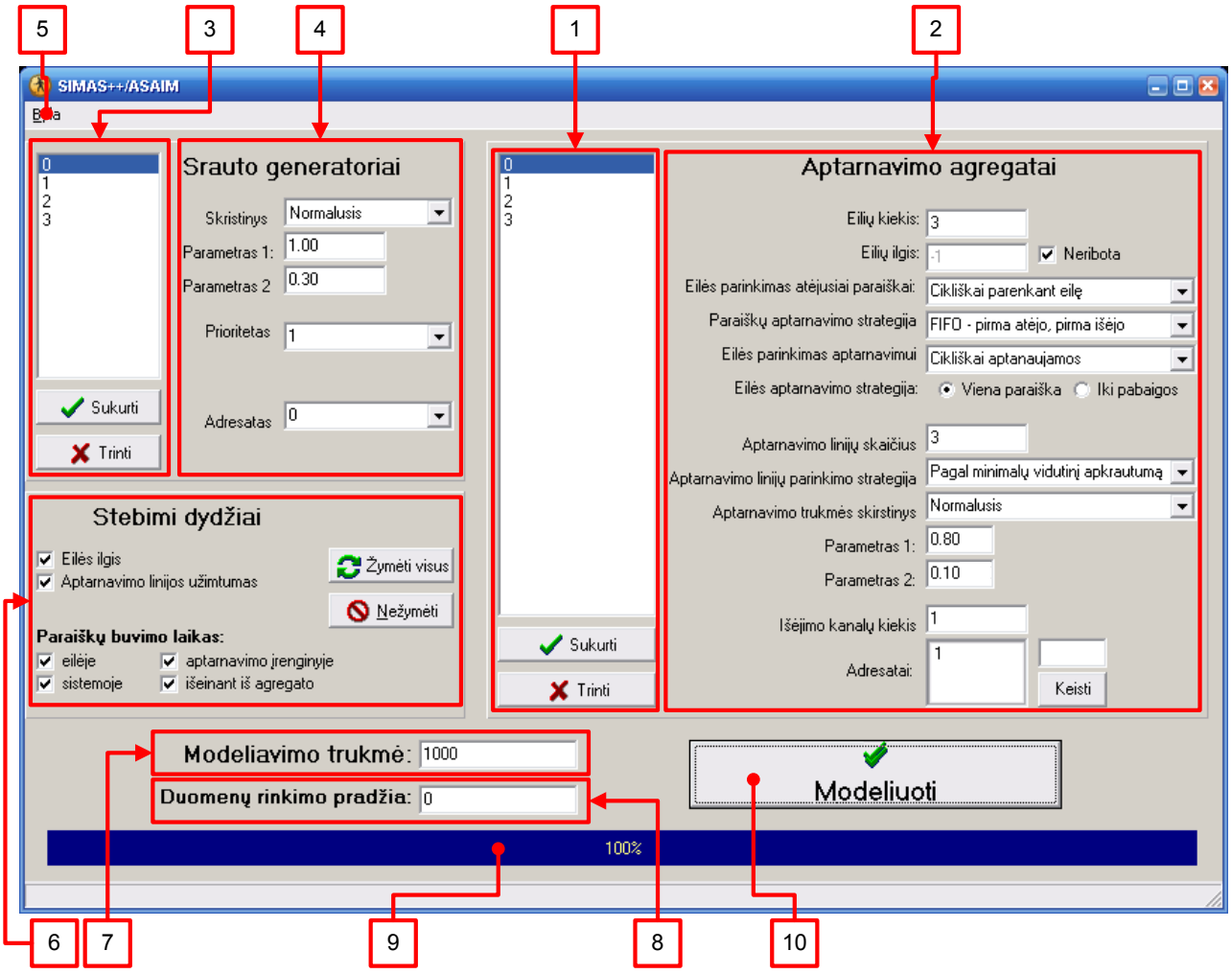
Atsivėrus modelių kūrimo langui, jame galima pasirinkti: kurti naują modelį ar vykdyti jau sukurtą. Sukurto modelio atidarymas vyksta 5 numeriu (2.5 pav) nurodyto meniu punkte „Atverti“ arba naudojant klavišų kombinaciją „Ctrl + A”. Atsidarius modelio pasirinkimo dialogo langui išsirenkamas modelio failas ir vykdomas modelio informacijos skaitymas ir atvaizdavimas modeliavimo sistemoje SIMAS++/ASAIM.

Kuriant naują imitacinį modelį, pirmiausia rekomenduojama sukurti aptarnavimo įrenginius imituojančius agregatus. 1 numeriu (2.5 pav) pažymėtoje srityje galima pridėti (mygtukas „Pridėti“) ir ištrinti (mygtukas „Trinti“) aptarnavimo įrenginiai. Pasirinkus aptarnavimo įrenginį iš 1 numeriu (2.5 pav) pažymėtos srities sąrašo, jo parametrai atvaizduojami 2 numeriu pažymėtoje srityje (2.5 pav). Toje srityje galimas pasirinkto įrenginio parametrų keitimas, adresatų nurodymas.

Sudarius aptarnavimo agregatų sąrašą su atitinkamais ryšiais, pridedami paraiškų srauto generatoriai srityje „Srauto generatoriai“ (3 ir 4 numeriu pažymėta sritis 2.5 paveiksle). Spaudžiant mygtuką „Pridėti“ (3 numeriu pažymėta sritis) pridedamas paraiškų srauto generatorius ir jo parametrai atvaizduojami 4 numeriu pažymėtoje srityje. Parenkami konkretūs pasirinkto generatoriaus parametrai, išrenkami adresatai. Norint ištrinti nereikalingą generatorių, pasirenkamas generatorius iš generatorių sąrašo ir spaudžiamas mygtukas „Trinti“ (3 numeriu pažymėta sritis).

Pasirenkami stebimi dydžiai (6 numeriu pažymėta sritis). Užsakytų stebimų dydžių informaciją sistema kaups modeliavimo duomenų matricoje. Nepažymėjus nei vieno stebimo dydžio, sistema nekaups jokios informacijos.

Nurodoma modeliavimo trukmė (7 numeriu pažymėta sritis), laikas nuo kurio pradėti stebimų dydžių duomenų kaupimą (8 numeriu pažymėta sritis) ir spaudžiamas mygtukas „Modeliuoti“ (10 numeriu pažymėta sritis).



2.5 pav. Dialoginė SIMAS++/ASAIM vartotojo sąsaja

Modeliavimo proceso metu matoma sisteminio laiko ir modeliavimo trukmės santykis išreikštas procentas (9 numeriu pažymėta sritis).

Sistemai baigus modeliavimą (kai sisteminis laikas pasiekia modeliavimo trukmės ribą) rodomas 100% vykdymas ir pranešimas apie išleistų ir aptarnautų paraiškų skaičių (paraiškų išėjusių iš sistemos).

Užsakytų stebimų dydžių modeliavimo duomenų matrica suformuojama tome pačiame kataloge kaip ir užsaugotas modelis. Jeigu modelis yra naujas ir dar niekur neišsaugotas, tai modeliavimo duomenų matrica bus suformuota darbiniame kataloge. Užsakytų stebimų dydžių modeliavimo duomenų matrica saugoma faile „out.txt“. Modeliavimo duomenų matricos struktūra aprašyta 2.2.2 skyrelyje.

Reikėtų atkreipti dėmesį į modeliavimo trukmės parinkimą. Ilgas modeliavimo laikas, stebimų dydžių kiekis ir santykinai intensyvių paraiškų generavimo ir aptarnavimo skirstinių parametrų nustatymas gali ženkliai įtakoti modeliavimo duomenų matricos dydį. Tai nesukelia didesnių problemų statistinės analizės paketui SAS, tačiau gali ženkliai įtakoti skaičiavimo laiką sistemoje MathCad.

2.3.3. SISTEMŲ SIMAS++/ASAIM, MATHCAD IR SAS INTEGRAVIMAS

Kuriant aptarnavimo sistemų agregatinių imitacinių modelių statistinės analizės posistemę, buvo apjungta aptarnavimo sistemų imitacinio modeliavimo programa su statistinės analizės ir matematiniu paketais.

Kadangi aptarnavimo sistemų imitacinio modeliavimo programoje SIMAS++/ASAIM, nėra realizuotas statistinės analizės modelis, tai eksperimento duomenų statistinė analizė vykdoma panaudojant universalų matematinį paketą MathCad ir/arba statistinės analizės paketą SAS.

Statistinės analizės modeliai yra realizuoti MathCad ir SAS paketuose, kurie pateikia modeliavimo duomenų statistinės analizės rezultatus. Statistinės analizės modeliai taip pat gali būti sukurti vartotojo, atsižvelgiant į sudaryto imitacinio modelio analizės tikslus.

Aprašomos statistikos ir grafinio vaizdavimo metodai yra realizuoti MathCad sistemoje, todėl informaciją, apie sukurto imitacinio modelio funkcionavimą, galima gauti pasinaudojus aprašomosios statistikos skaičiavimo funkcijomis (vidutinis eilės ilgis, vidutinis buvimo laikas eilėje, įrenginių užimtumas, vidutinis laikas sistemoje ir t.t)

Aprašomoji statistika pateikia stebimo dydžio imties vidurkį, imties dispersiją, standartinį nuokrypį, minimalią ir maksimalią imties reikšmę. Duomenų grafinio vaizdavimo funkcijos pateikia stebimų dydžių histogramas, pasiskirstymo funkcijas, stebimo dydžio reikšmių kitimo laike grafikus ir t.t. Tam pakanka MathCad matematinio paketo. Tačiau norint atlikti sudėtingesnę statistinę analizę (dispersinę analizę, regresinę analizę ir t.t.) rekomenduojama naudoti statistikos paketą SAS.

Sprendžiant konkrečią problemą dažnai reikia atlikti gilesnę imitacinio modelio statistinę analizę. Tuo atveju rekomenduojama pasinaudoti specializuotais statistinės analizės paketais. Šiame darbe pasirinktas SAS paketas, nes paketas SAS turi turtingą statistinės analizės procedūrų rinkinį ir galimybę kurti statistinės analizės modelius taikant makro komandas.

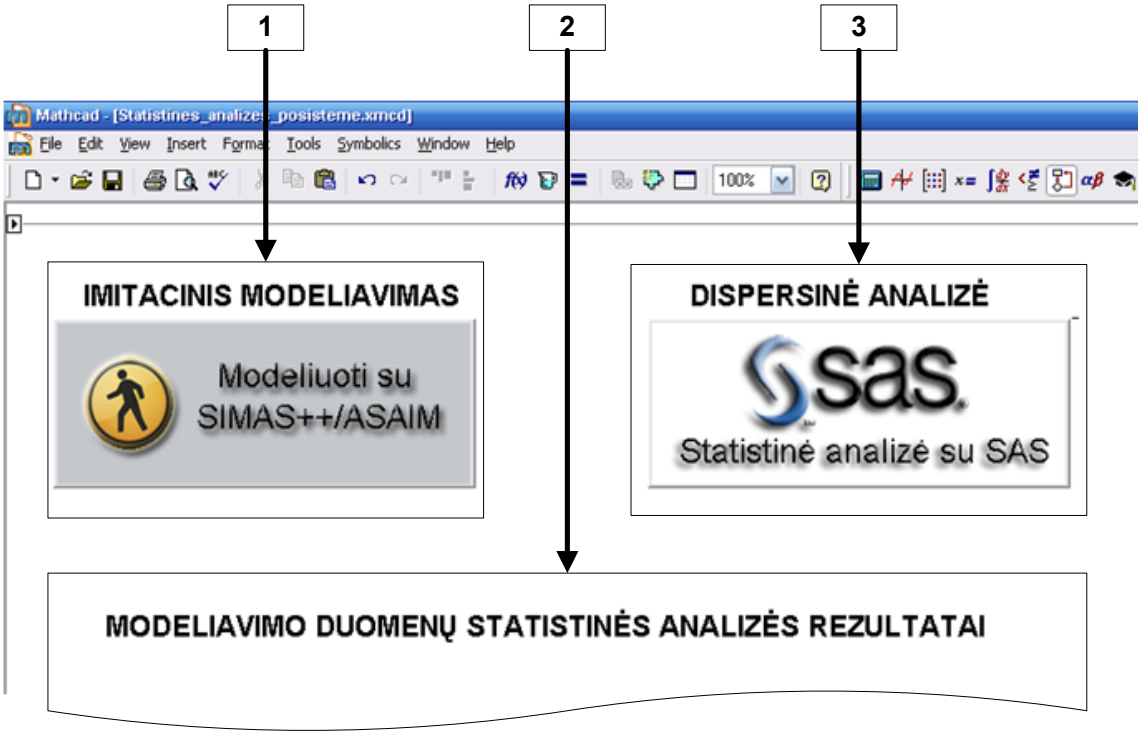
2.3.3.1. Valdantysis statistinės analizės proceso modulis

Sukurtas valdantysis statistinės analizės proceso modulis MathCad pakete, kuris iškviečia modeliavimo sistemą SIMAS++/ASAIM, skaitymą modeliavimo duomenų matricos, statistinės analizės matricų formavimą ir statistinės analizės rezultatų pateikimą.

Vykdant valdomąjį modulį *„Statistinės_analizės_posistemė.xmcd“* inicijuojamas realizuotų vykdymo funkcijų ir statistinės analizės modelių įkėlimas iš failo *„Funkcijos.xmcd“*. Jų realizacija pateikta 2 priede.

Norint gauti modeliavimo rezultatus būtina atlikti modeliavimą su modeliavimo programa SIMAS++/ASAIM, kurios iškvietimas spaudžiant 1 numeriu pažymėtą mygtuką (2.6 pav.)

Modeliavimo sistemoje sukuriamas imitacinis modelis, užsakomas modeliavimo duomenų rinkimas, vykdomas imitacinis aptaranavimo sistemos modeliavimas ir gaunama modeliavimo duomenų matrica (žr. 2.3.2 skyrelį).



2.6 pav. Statistinės analizės posistemės valdymo modulis

Kol vykdomas imitacinis modeliavimas sistemoje SIMAS++/ASAIM, valdantysis statistinės analizės modulis neatlieka jokių veiksmų.

Modeliavimo sistemai baigus modeliavimą ir uždarius SIMAS++/ASAIM programą, valdantysis modulis atlieka modeliavimo duomenų matricos transformavimą į statistinės analizės matricas ir pateikia statistinės analizės rezultatus paketu MathCad (2.6 pav. numeriu 2 pažymėta sritis „Modeliavimo duomenų statistinės analizės rezultatai“).

Norint atlikti gilesnę analizę, galima gauti statistinės analizės rezultatus ir su paketu SAS (paspaudus 3 numeriu pažymėta mygtuką (2.6 pav.)). Tada vykdomas modeliavimo duomenų transformavimas į statistinės analizės matricas pagal 2.2.3 skyrelyje aprašytą metodiką, atliekamas aprašytų statistinės analizės modelių taikymo procesas ir statistinės analizės rezultatų pateikimas internetinėje naršyklėje.

2.3.3.2. Modeliavimo duomenų statistinė analizė panaudojant MathCad

Pagal pasiūlytą duomenų matricos transformacijų atlikimo metodiką, sudaromos statistinės analizės matricos, kurios yra tinkamos realizuotiems statistinės analizės modeliams.

Transformacijas atlieka šie metodai:

1. *Pjuvis(Duomenys,StebDydis,Agregatas,Eile)* – metodas, atrenkantis modeliavimo duomenų matricos *(Duomenys)* eilutes pagal nurodytą stebimą dydį (*StebDydis*), aptarnavimo įrenginį (*Agregatas*), aptarnavimo liniją arba eilę (*Eile*);

$$\boldsymbol{Pjuvis}(\boldsymbol{Duomenys}, \boldsymbol{StebDydis}, \boldsymbol{Agregatas}, \boldsymbol{Eile}) = \boldsymbol{\pi}_{\{T,R\}}\left(\boldsymbol{\sigma}_{\{SD=StebDydis, A=Agregatas, I=Eile\}}(\boldsymbol{Duomenys})\right)$$

2. *PjuvisPagalStebDydi(Duomenys,StebDydis)* – metodas, atrenkantis modeliavimo duomenų matricos (Duomenys) eilutes atitinkančias pasirinką stebimą dydį StebDydis.

$$\boldsymbol{PjuvisPagalStebDydis}(\boldsymbol{Duomenys}, \boldsymbol{StebDydis}) = \boldsymbol{\pi}_{\{T,R\}}\left(\boldsymbol{\sigma}_{\{SD=StebDydis\}}(\boldsymbol{Duomenys})\right)$$

```
PjuvisPagalStebDydis ( Duomenys , StebDydis) :=
    m ← −1
    Rezultatas ← 0
    for i ∈ 0.. rows( Duomenys) − 1
      if str2vec(Duomenys_{i,3}) = str2vec( StebDydis)
        m ← m + 1
        Rezultatas_{m,0} ← Duomenys_{i,1}
        Rezultatas_{m,1} ← Duomenys_{i,7}
    return Rezultatas
```

2.7 pav. Metodo PjuvisPagalStebDydis realizacija (MathCad)

3. *PjuvisPagalAgregataIrEile(Duomenys,Agregatas,Eile)* – metodas, atrenkantis modeliavimo duomenų matricos (Duomenys) eilutes atitinkančias pasirinktus aptarnavimo įrenginius (Agergatas) ir aptarnavimo linijas arba eiles (Eile).

$$\boldsymbol{Pjuvis}(\boldsymbol{Duomenys}, \boldsymbol{Agregatas}, \boldsymbol{Eile}) = \boldsymbol{\pi}_{\{T,R\}}\left(\boldsymbol{\sigma}_{\{A=Agregatas, I=Eile\}}(\boldsymbol{Duomenys})\right)$$

4. *PjuvisPrioritetas(Duomenys,StebDydis,Agregatas,Apt_eile,Prioritetas)* - metodas atrenkantis, modeliavimo duomenų matricos *(Duomenys)* eilutes pagal nurodytą stebimą dydį (*StebDydis*), aptarnavimo įrenginį (*Agregatas*), aptarnavimo liniją arba eilę (*Eile*) ir paraiškos prioritetą (*Prioritetas)*.

$$\boldsymbol{PjuvisPrioritetas}(\boldsymbol{Duomenys}, \boldsymbol{StebDydis}, \boldsymbol{Agregatas}, \boldsymbol{Apt_{eile}}, \boldsymbol{Prioritetas}) = \boldsymbol{\pi}_{\{T,R\}}\left(\boldsymbol{\sigma}_{\{SD=StebDydis, A=Agregatas, I=Apt_{eile}, P=Prioritetas\}}(\boldsymbol{Duomenys})\right)$$

5. *Atrinkimas(Duomenys,StebDydis)* – metodas iš modeliavimo duomenų matricos *(Duomenys)* surenka informaciją, kuriuose aptarnavimo įrenginiuose ir aptarnavimo linijose arba eilėse buvo atliekami stebimo dydžio stebėjimai. Grąžinama matrica $A = a_{i,j}, i = \overline{0,M}, j = \overline{0,1}$, kur M – aptarnavimo linijų/eilių skaičius, kuriuose buvo atliekami matavimai, $a_{i,0}$ – aptarnavimo įrenginio numeris, $a_{i,1}$- aptarnavimo linijos/eilės numeris.

$$\boldsymbol{Atrinkimas}(\boldsymbol{Duomenys}, \boldsymbol{StebDydis}) = \boldsymbol{\pi}_{\{A,I\}}\left(\boldsymbol{\sigma}_{\{SD=StebDydis\}}(\boldsymbol{Duomenys})\right)$$

```
Atrinkimas( Duomenys, StebDydis) :=
    n ← 1
    yra ← 1
    Rezultatas ← PjuvisPagalStebDydis ( Duomenys, StebDydis)
    RezultatasG[0,0] ← Rezultatas[0,4]
    RezultatasG[0,1] ← Rezultatas[0,5]
    for i ∈ 1.. rows( Rezultatas) − 1
        a ← Rezultatas[i,4]
        b ← Rezultatas[i,5]
        for j ∈ 0.. n − 1
            if RezultatasG[j,1] = b ∧ RezultatasG[j,0] = a
                yra ← 1
                j ← n − 1
        if yra ≠ 1
            RezultatasG[n,0] ← a
            RezultatasG[n,1] ← b
            n ← n + 1
        yra ← 0
    RezultatasG ← Rikiavimas(csort(RezultatasG, 0))
    return RezultatasG
```

2.8 pav. Metodo Atrinkimas realizacija (MathCad)

Funkcijos atliekančios statistinės analizę:

1. *Vidutinis_ilgis(Duomenys)* – metodas, pateikiantis vidutinį eilės ilgį.

2. *Eiliu_ilgiai(Duomenys)* – metodas, apjungia metodus susijusius su eilių ilgių charakteristikomis (didžiausias ilgis, vidutinis ilgis ir t.t.) ir grąžina matricą $B = b_{i,j}, i = \overline{0,M}, j = \overline{0,4}$, kur M – aptarnavimo eilių skaičius, kuriuose buvo atliekami matavimai, $b_{i,0}$ – aptarnavimo įrenginio numeris, $b_{i,1}$ – eilės numeris aptarnavimo įrenginyje, $b_{i,2}$ – imties dydis, $b_{i,3}$ – didžiausias eilės ilgis, $b_{i,4}$- vidutinis eilės ilgis. Rezultatų pavyzdys pateiktas 2.9 paveiksle.

Eilių_ilgiai (Data) =

"Įrenginys"	"Eilė"	"Imties dydis"	"MAX"	"Vidurkis"
0	0	2301	4	0.38307
0	1	2300	4	0.34967
0	2	2300	4	0.36751
1	0	2298	3	0.19024
1	1	2298	3	0.17831
1	2	2298	3	0.1652

2.9 pav. Eilių ilgių skaitinės charakteristikos

3. *Uzimtumas(Duomenys,Agregatas,Linija)* – metodas skaičiuojantis aptarnavimo įrenginio (Agregatas) aptarnavimo linijos (Linija) užimtumą.

4. *Užimtumo_Lentelė(Duomenys)* - metodas grąžina matricą $C = c_{i,j}, i = \overline{0,M}, j = \overline{0,2}$, čia M – aptarnavimo linijų skaičius, $c_{i,0}$ – aptarnavimo įrenginio numeris, $c_{i,1}$ – aptarnavimo linijos numeris aptarnavimo įrenginyje, $c_{i,2}$ – aptarnavimo linijos užimtumas (%). Pavyzdys pateiktas 2.10 paveiksle.

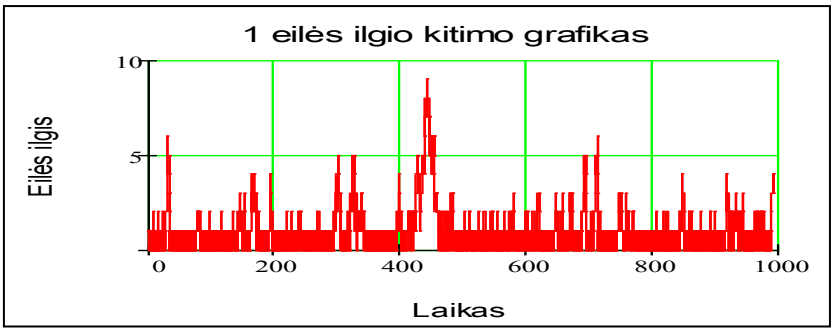
Užimtumo_Lentelė (Data) =

"Įrenginys"	"Linija"	"Užimtumas (%)"
0	0	92.932514
0	1	93.123228
0	2	91.015729
1	0	91.78234
1	1	92.010893

2.10 pav. Įrenginių linijų užimtumas

5. *Paraiškų_buvimo_laikas_eilėje(Duomenys)* – metodas grąžina matricą $E = e_{i,j}, i = \overline{0,M}, j = \overline{0,5}$, čia M – aptarnavimo eilių skaičius, kuriuose buvo atliekami matavimai, $e_{i,0}$ – aptarnavimo įrenginio numeris, $e_{i,1}$ – eilės numeris aptarnavimo įrenginyje, $e_{i,2}$ – imties dydis, $e_{i,3}$ – vidurkis. $e_{i,4}$ – dispersija, $e_{i,5}$ – mediana.

6. *Eilių ilgių kitimo grafikai*. Sukurtas aptarnavimo įrenginių su trimis eilėmis vaizdavimas. 2.11 pav. pateikiamas pavyzdys rodo aptarnavimo įrenginio pirmos eilės ilgio kitimą modeliavimo metu.



2.11 pav. Eilės ilgio kitimo grafikas

7. *Laikas_sistemoje_iki_išeinant_iš_įrenginio(Duomenys)* – metodas grąžina matricą: $K = k_{i,j}, i = \overline{0,M}, j = \overline{0,5}$, čia M – aptarnavimo įrenginių skaičius, kuriuose buvo atliekami matavimai, $k_{i,0}$ – aptarnavimo įrenginio numeris, $k_{i,1}$ – atliktų matavimų skaičius, $k_{i,2}$ – mažiausia stebėta reikšmė, $k_{i,3}$ – didžiausia stebėta reikšmė, $k_{i,4}$ – vidutinis paraiškų buvimo laikas sistemoje išeinant iš i-tojo aptarnavimo įrenginio, $k_{i,5}$ – paraiškų buvimo laiko sistemoje išeinant iš i-tojo aptarnavimo įrenginio dispersija. Rezultatų pavyzdys pateiktas 2.12 paveiksle.

Laikas_sistemoje_iki_išeinant_iš_įrenginio (Data) =

"Įrenginys"	"Imties dydis"	"MIN"	"MAX"	"Vidurkis"	"Dispersija"
0	3485	0.4673	9.83886	1.15103	0.40601
1	3485	0.61582	10.467	2.10921	1.61814

2.12 pav. Paraiškų buvimo laiko sistemoje iki išėjimo iš aptarnavimo įrenginio skaitinės charakteristikos

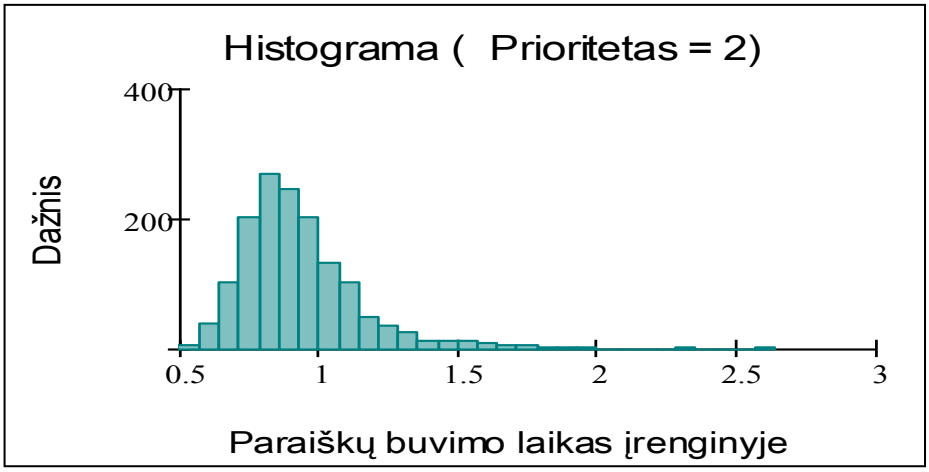
8. *Paraiškų_Buvimo_Laikas_sistemoje_iki_išeinant_iš_įrenginio_pagal_prioritetą(Duomenys)* – metodas grąžina matricą: $K = k_{i,j}, i = \overline{0, M}, j = \overline{0,5}$, čia M – aptarnavimo įrenginių skaičius, kuriuose buvo atliekami matavimai, $k_{i,0}$ – aptarnavimo įrenginio numeris, $k_{i,1}$ – paraiškos prioritetas, $k_{i,2}$ – atliktų matavimų skaičius, $k_{i,3}$ – vidurkis, $k_{i,4}$ – dispersija, $k_{i,5}$ – mediana.

9. *Paraiškų buvimo laiko sistemoje histograma iki išėjimo iš įrenginio. Pateikiamos atskiros skirtingo prioriteto paraiškų buvimo laiko sistemoje iki išėinant iš įrenginio histogramos ir modeliavimo fiksuotų reikšmių diagrama*. Posistemėje realizuota paraiškų buvimo laikų sistemoje iki išėjimo iš įrenginių histogramų ir atskirų histogramų pagal skirtingus paraiškų prioritetus vaizdavimas.

10. *Paraiškų_buvimo_laikas_įrenginyje(Duomenys)* – metodas grąžina matricą: $E = e_{i,j}, i = \overline{0, M}, j = \overline{0,5}$, kur M – aptarnavimo linijų skaičius, kuriuose buvo atliekami matavimai, $e_{i,0}$ – aptarnavimo įrenginio numeris, $e_{i,1}$ – aptarnavimo linijos numeris įrenginyje, $e_{i,2}$ – imties dydis, $e_{i,3}$ – vidutinis laikas praleistas įrenginyje , $e_{i,4}$ – dispersija, $e_{i,5}$ – mediana.

11. *Paraiškų_buvimo_laikas_įrenginyje_pagal_prioritetą(Duomenys)* – metodas grąžinana matricą $E = e_{i,j}, i = \overline{0, M}, j = \overline{0,5}$, kur M – prioritetų kiekio ir aptarnavimo linijų, kuriuose paraiškos buvo aptarnaujamos, sandauga, $e_{i,0}$ – aptarnavimo įrenginio numeris, $e_{i,1}$ – aptarnautų paraiškų prioritetas, $e_{i,2}$ – imties dydis, $e_{i,3}$ – vidutinis laikas praleistas įrenginyje , $e_{i,4}$ – dispersija, $e_{i,5}$ – mediana.

12. *Paraiškų buvimo laiko aptarnavimo įrenginyje histograma ir skirtingos histogramos pagal paraiškų prioritetus.* Posistemėje realizuota paraiškų buvimo laikų įrenginyje histogramų ir atskirų histogramų pagal skirtingus paraiškų prioritetus vaizdavimas. 2.13 pav. pateikta antro prioriteto paraiškų buvimo laiko pirmajame įrenginyje histograma.



2.13 pav. Antro prioriteto paraiškų buvimo laiko pirmajame aptarnavimo įrenginyje histograma

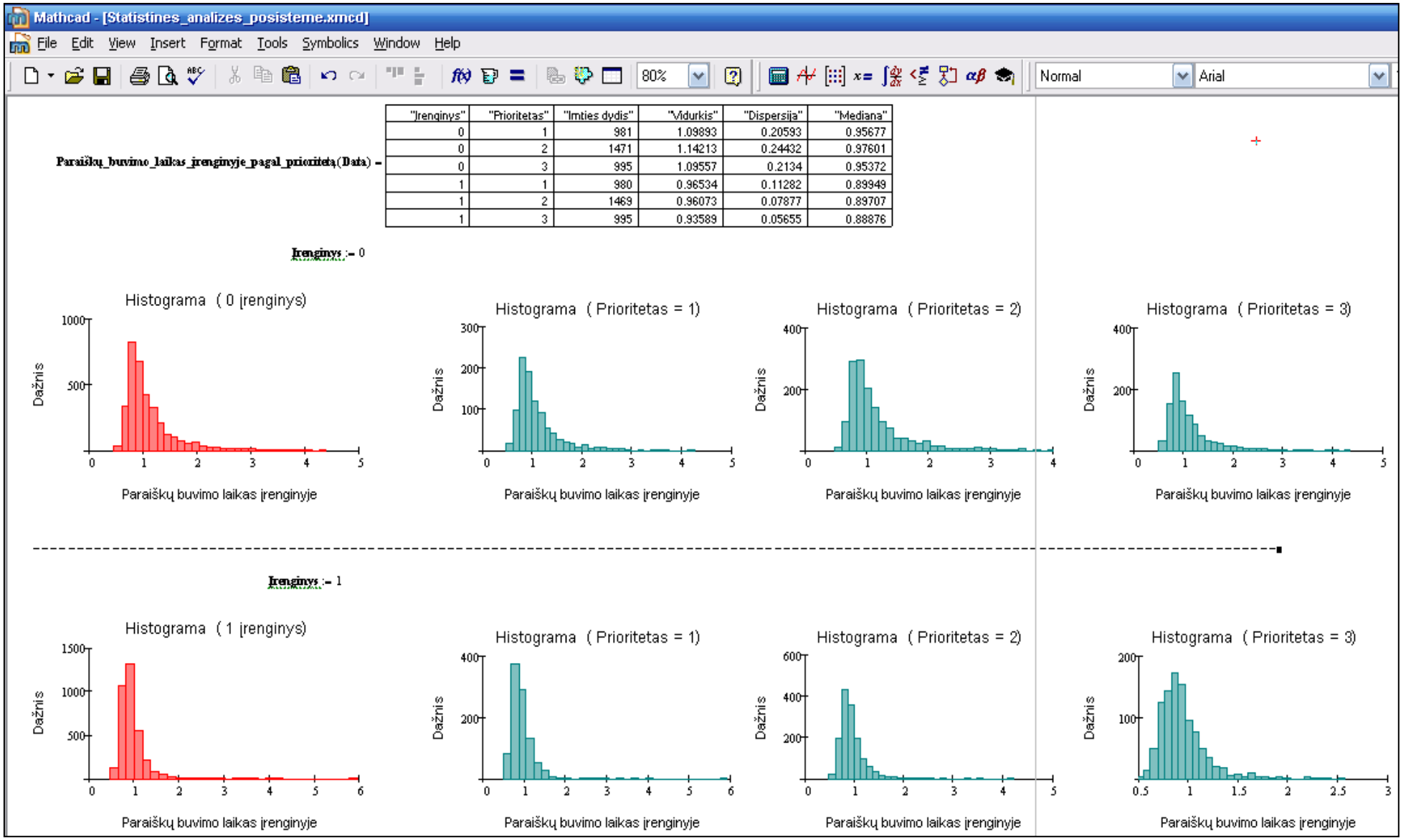
13. *Paraiškų buvimo laiko sistemoje histograma.* Taip pat pateikiama skirtingo prioriteto paraiškų buvimo laiko sistemoje histogramos ir modeliavimo metu stebėtų reikšmių diagrama. Posistemėje

realizuotas paraiškų buvimo laikų sistemoje histogramų ir atskirų histogramų pagal skirtingus paraiškų prioritetus vaizdavimas.

MathCad paketą patariama naudoti norint gauti aprašomosios statistikos dydžius, aprašančius kokybines modelio funkcionavimo reikšmes, užimtumo, eilių kitimo diagramas, aptarnavimo laiko histogramas.

Modeliavimo rezultatų, gautų matematiniu paketu MathCad, pavyzdys pateiktas 2.14 pav.

Norint gauti kitas modelio charakteristikas ar atlikti įvairią statistinę analizę, vartotojui palikta galimybė pačiam kurti statistinės analizės modelius. Žinant, kaip yra formuojama modeliavimo duomenų matrica ir kaip veikia matricos transformacijas atliekantys metodai, duomenų statistinei analizei nesunkiai galima taikyti ir kitus metodus.



2.14 pav. Statistinės analizės rezultatų pavyzdys (MathCad)

2.3.3.3. Modeliavimo duomenų statistinė analizė panaudojant SAS

Aptarnavimo sistemų modeliavimo sistema SIMAS++/ASAIM funkcionuoja nepriklausomai nuo pasirinktos statistinės analizės sistemos. Realizuojant statistinės analizės posistemę buvo pasirinktas MathCad paketas, kuris apjungia modeliavimo sistemą SIMAS++/ASAIM ir su statistinės analizės paketu SAS.

Statitistinės analizės paketas SAS pasirinktas dėl turtingo statistinės analizės funkcijų rinkinio bei makro komandų galimybių. [16]

Foniniu rėžimu aktyvuojamas SAS paketas, kuris atlieka statistinę analizę. Užduoties failo adresas perduodamas kaip parametras. Užduoties faile pateikiamas tik modeliavimo duomenų matricos adresas, statistinės analizės rezultatų failo pavadinimas, išsaugojimo adresas ir kreipiniai į makro komandas, kurios ir atlieka statistinę analizę.

Naudojantis turtinga SAS statistinės analizės funkcijų biblioteka, sukurtas vienfaktorės dispersinės analizės modelis, kuris apima tiek parametrinę dispersinę analizę su daugialypio palyginimo Scheffe kriterijumi, tiek neparametrinę dispersinę analizę su Kruskal – Wallis kriterijumi. Šiam modeliui realizuoti sukurta makro komanda, kuri ir naudojama užduoties faile.

Vienfaktorė dispersinė analizė. Faktorius ϕ - paraiškos prioritetas: $\phi \in \Phi = \{1,2,3\}$. Hipotezė: $H_0 : \mu_1 = \mu_2 = \mu_3$, H_1 : ne visi buvimo laikų vidurkiai tarpusavyje lygūs, α = 0,05. μ_i - i-tojo prioriteto paraiškų buvimo laikas sistemoje

Hipotezės tikrinimui naudojama statistika: $F = (MS_B / MS_R), F \sim F_{J-1,n-J}$, čia J-faktorių skaičius, n - matavimų skaičius.

Daugialypio palyginimo metodas: vidurkių lyginimas poromis. Hipotezė: $H_0 : \mu_i = \mu_j, H_1 : \mu_i \neq \mu_j, \alpha = 0{,}05\, i, j = \overline{1..3}, i \neq j$.

Naudojamas Scheffe kriterijus[18]: $(\bar{x}_k - \bar{x}_l) - S\hat{\sigma}^2 \leq (\mu_k - \mu_l) \leq (\bar{x}_k - \bar{x}_l) + S\hat{\sigma}^2$, kur $S\hat{\sigma}^2 = \sqrt{\overline{SS}_e (J-1)(1/n_k - 1/n_l) F_{\alpha;(J-i);(n-J)}}$.

Atlikus parametrinę dispersinę analizę ir nustačius, kad liekanų skirstinys nėra normalusis, taikau neparametrinę dispersinę analizę. Hipotezei tikrinti naudojama Kruskal-Wallis kriterijus:

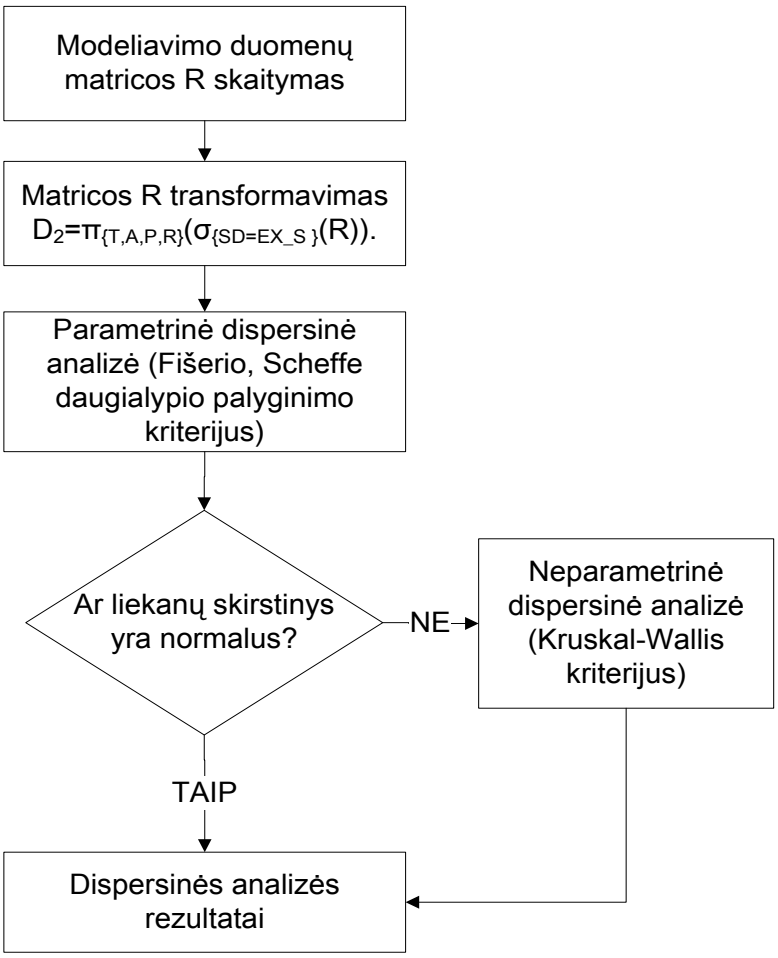
$$K = \frac{12}{n(n+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(n+1), K \sim \chi^2_{\alpha,k-1}$$. R_i – rangų suma i-oje grupėje.

Sukurta vienfaktorės dispersinės analizės makro komanda, kuri atlieka dispersinę analizę:

%macro dispersine(duomenys, stebimas_dydis, Prioritetas, Duomenu_rinkimo_pradžia) – makro komanda atlieka duomenų matricos *duomenys* kintamajam *stebimas_dydis* disperisinę analizę

pagal kintamąjį *Prioritetas*. Čia *Duomenu_rinkimo_pradžia* yra vartotojo nurodytas sistemos laikas nuo kurio sistema, manoma, tampa stacionari.

Dispersinės analizės makro komandos algoritmas pateikiamas 2.15 paveiksle.



2.15 pav. Algoritmo schema

Taikant dispersinės analizės modelį, pradžioje vykdoma taikoma parametrinė dispersinė analizė (Fišerio kriterijus) ir gaunami rezultatai. Iš pateikiamų 2.3, 2.4 lentelėse pavyzdžių matome, kad hipotezė apie skirtingo prioriteto paraiškų buvimo vidutinio laikų sistemoje lygybė neatmetama (p reikšmė 0.99). Daugialypio palyginimo pagal Scheffe kriterijų rezultatai rodo, kad visų prioritetų paraiškų vidutiniai buvimo laikai sistemoje patenka į vieną grupę (2.4 lentelė).

2.3 lentelė.

Hipotezės tikrinimo rezultatų pavyzdys (SAS)

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	0.08115	0.04057	0.01	0.9906
Error	3490	15039.50589	4.30931		
Corrected Total	3492	15039.58704			

2.4 lentelė.

Daugialypio palyginimo Scheffe kriterijaus taikymo rezultatai (SAS)

Scheffe Grouping	Mean	N	Prioritetas
A	4.67094	1512	2
A	4.66547	983	3
A	4.65936	998	1

Tačiau šie rezultatai yra nekorektiški, nes hipotezė apie liekanų skirstinio normalumą, yra atmetama, t.y. netenkina dispersinės analizės modelio taikymo prielaidų (2.5 lentelė).

2.5 lentelė.

Hipotezės apie liekanų skirstinio normalumą tikrinimo rezultatai (SAS)

Tests for Normality				
Test	**Statistic**		**p Value**	
Kolmogorov-Smirnov	**D**	0.14507	**Pr > D**	<0.0100
Cramer-von Mises	**W-Sq**	30.18663	**Pr > W-Sq**	<0.0050
Anderson-Darling	**A-Sq**	173.7067	**Pr > A-Sq**	<0.0050

Kai duomenys netenkina parametrinės dispersinės analizės prielaidų, tuomet taikoma neparametrinė dispersinė analizė. Kruskal – Voliso kriterijus. Tikrinama nulinė hipotezė H_0: „stebimo dydžio skirtstiniai k grupėse yra vienodi“, alternatyva, kad skirstiniai bent dvejose vietose skiriasi. 2.6 lentelėje pateiktas rezultatų pavyzdys, kuris nurodo, kad parametrinė analizė atliekama nekorektiškai, ir pateikiami tik neparametrinės dispersinės analizės rezultatai.

2.6 lentelė.

Neparametrinės dispersinės analizės pavyzdys(SAS)

Kruskal-Wallis Test	
Chi-Square	0.2070
DF	2
Pr > Chi-Square	0.9017

Rezultatų faile pateikiami tie dispersinės analizės rezultatai, kurių duomenys tenkino prielaidas. Jeigu išpildomos visos parametrinės dispersinės analizės taikymo prielaidos, tada pateikiamos tik parametrinės dispersinės analizės rezultatai. Priešingu atveju pateikiami tik neparametrinės dispersinės analizės rezultatai.

2.4. STATISTINĖS ANALIZĖS POSISTEMĖS TAIKYMAI

2.4.1. MODELIS AS1

2.4.1.1. Modelio AS1 aprašymas

Aptarnavimo sistemų imitacinių modelių statistinės analizės posistemės testavimui buvo sudarytas aptarnavimo sistemos imitacinis modelis, kurio struktūra pateikta 2.16 paveiksle.

Aptarnavimo sistemos tyrimo tikslas - sudaryti aptarnavimo sistemos (2.16 pav.) ASAIM modelį ir ištirti skirtingo prioriteto paraiškų buvimo laikus sistemoje panaudojant dispersinės analizės metodus.

2.7 lentelė.

Aptarnavimo įrenginių parametrai

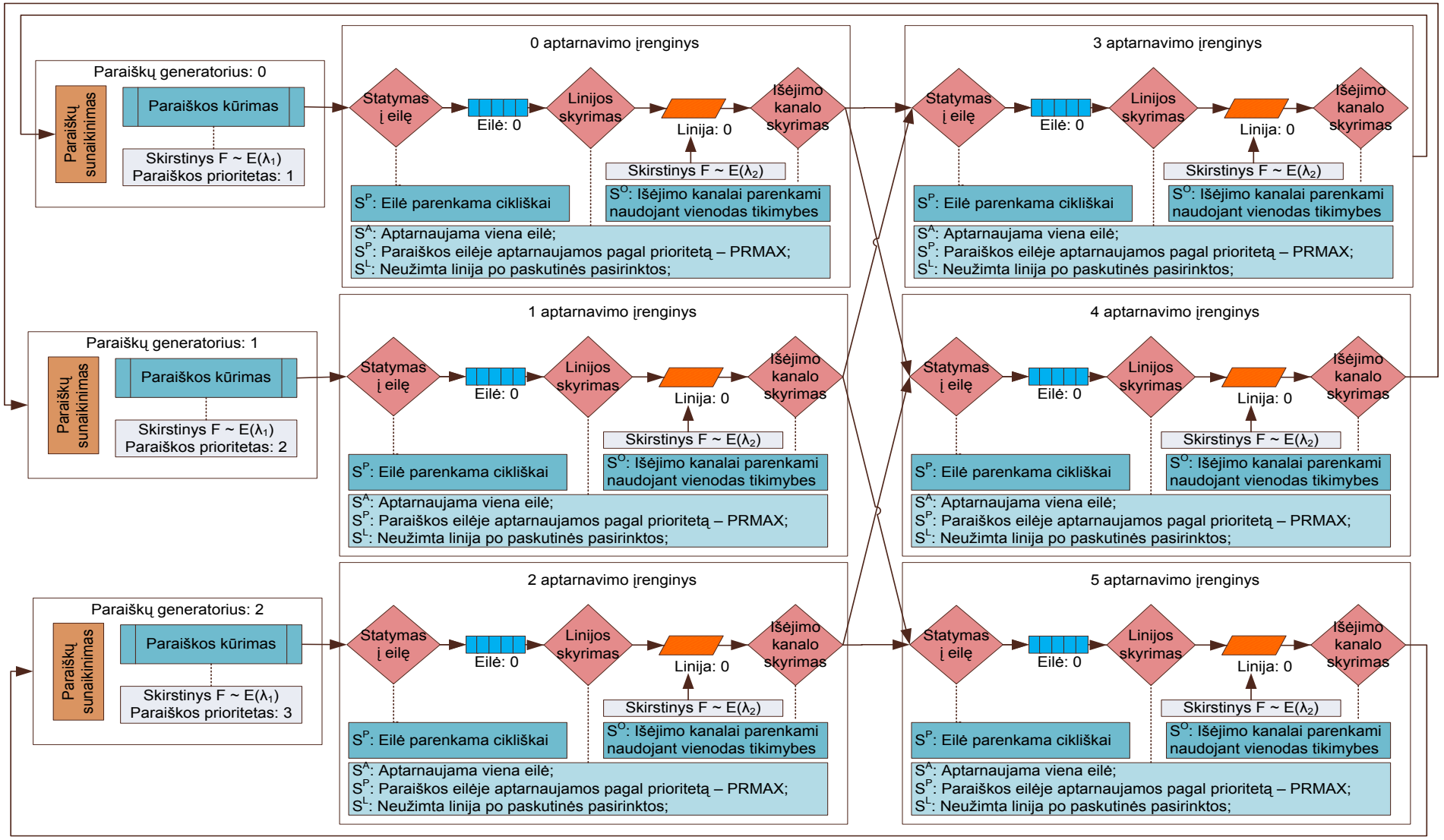
Parametrai	Įrenginių parametrai
Išėjimo kanalų kiekis (L)	L = 2
Išėjimo kanalo parinkimo strategija	Atsitiktinai, su vienodomis tikimybėmis
Eilių kiekis (M)	M = 1
Eilės parinkimas atėjusiai paraiškai (S^Q)	Ciklišai parenkant eilę
Maksimalus eilės ilgis (l^Q)	∞
Paraiškos parinkimo aptarnavimui strategija (S^P)	PRMAX – pagal aukščiausią prioritetą.
Aptarnaujamos eilės pasirinkimo strategija (S^A)	Ilgiausia eilė
Aptarnavimo linijos pasirinkimo strategija (S^L)	Neužimta linija po paskutinės pasirinktos
Aptarnavimo linijų kiekis	N = 1
Aptarnavimo trukmės skirstinys (F)	E(λ_2)

Modeliio kūrimui panaudotas aptarnavimo agregatas. Visų šešių aptarnavimo agregatų kopijų parametrai yra vienodi (2.7 lentelė). Agregatai tarpusavyje sujungti pagal schemą pateiktą 2.16 pav.

Modelyje naudojamos trys paraiškų srauto generatoriaus kopijos. Intervalo tarp paraiškų kūrimo skirstiniai yra eksponentiniai E(λ_1). Kiekvienas generatorius generuoja vieno pirioriteto paraiškas (1, 2, 3). Iš kiekvieno srauto generatoriaus išeinantis srautas nukreiptas į vieną iš aptarnavimo įrenginių. Generatorius „0“ į įrenginį „0“, generatorius „1“ į įrenginį „1“, generatorius „2“ į įrenginį „2“.

Buvo renkami modeliavimo duomenis apie įrenginių užimtumą ir paraiškų buvimo laiką sistemoje. Modeliuota 1000 sisteminių laiko vienetų.

Sukurtam modeliui atlikta dispersinė analizė, t.y. tikrinta hipotezė apie skirtingo prioriteto paraiškų buvimo vidutinių laikų sistemoje lygybę. Keičiant sistemos apkrovimą, atlikti keturi eksperimentai, kurių rezultatai pateikiami 2.4.1.2 skyrelyje.



2.16 pav. Aptarnavimo sistemos AS1 modelio schema

2.4.1.2. Modelio AS1statistinės analizės rezultatai

Panaudoję dispersinės analizės metodą, patikrinta hipotezė, ar skirtingų prioritetų paraiškų vidutiniai buvimo laikai sistemoje yra lygūs.

Atlikti keturi eksperimentai keičiant sistemos apkrovimo lygį. Pirmi trys eksperimentai buvo atliekami didinant srauto intensyvumus (λ_1 = 0,1; 0,2; 0,3). Ketvirtu atveju, srauto generatoriaus eksponentinio skirstinio intensyvumas buvo 0,3, o aptarnavimo trukmės skirstinio parametras 3, 4 ir 5 aptarnavimo įrenginiuose padidintas iki 1. Tai leido sumažinti 3, 4 ir 5 aptarnavimo įrenginių linijų užimtumą nuo 40% iki 27%.

2.8 lentelė.

Modelio AS1 statistinės analizės rezultatai.

	Srauto generatorius	Aptarnavimo įrenginiai		Kruskal - Volis kriterijus	Hipotezė
	Skirstinys	Skirstinys	Užimtumas	p-reikšmė	H_0
1	E(0.1)	Visi įrenginiai - E(0.7)	14%	0,7570	Neatmesta
2	E(0.2)	Visi įrenginiai - E(0.7)	28%	0,3125	Neatmesta
3	E(0.3)	Visi įrenginiai - E(0.7)	43%	0,0137	Atmesta
4	E(0.3)	0,1,2 įrenginiai - E(0.7)	42%	0.0932	Neatmesta
		3,4,5 įrenginiai - E(1)	29%		

Gauti rezultatai pateikti 2.8 lentelėje. Dispersinės analizės modelio taikymas sistemoje SAS pataikė neparametrinės dispersinės analizės rezultatus visais atvejais, nes buvo atmesta liekanų normalumo hipotezė (liekanų skirstinys turi būti normalusi). Neparametrinė dispersinė analizė (Kruskal – Voliso rezultatai): kai sistema nėra apkrauta (įrenginių užimtumas - 28%), tai nėra statistiškai reikšmingo skirtumo tarp skirtingo prioriteto paraiškų buvimo vidutinių laikų sistemoje (pirmas ir antras eksperimentas 2.8 lentelėje). Taip pat hipotezė neatmetama, kai dalies įrenginių užimtumas atitinkamai yra 29% ir 42% (4 eksperimentas). Tačiau, kai visų įrenginių užimtumas 42%, tai hipotezė apie vidurkių lygybes atmetama (trečias eksperimentas).

Visų eksperimentų rezultatai gauti atliekant dispersinę analizę pateikiami 2 priede.

2.4.2. MODELIS AS2

2.4.2.1. Modelio AS2 aprašymas

Testuojant aptarnavimo sistemų imitacinių modelių statistinės analizės posistemę, buvo sudarytas dar vienas aptarnavimo sistemos imitacinis modelis, kurio struktūra pateikta 2.17 paveiksle.

Tyrimo tikslas – prie skirtingų aptarnavimo įrenginių parametrų, pateikti statistinės analizės rezultatus gautus matematikos paketo MathCad.

2.9 lentelė.

Aptarnavimo įrenginių parametrai

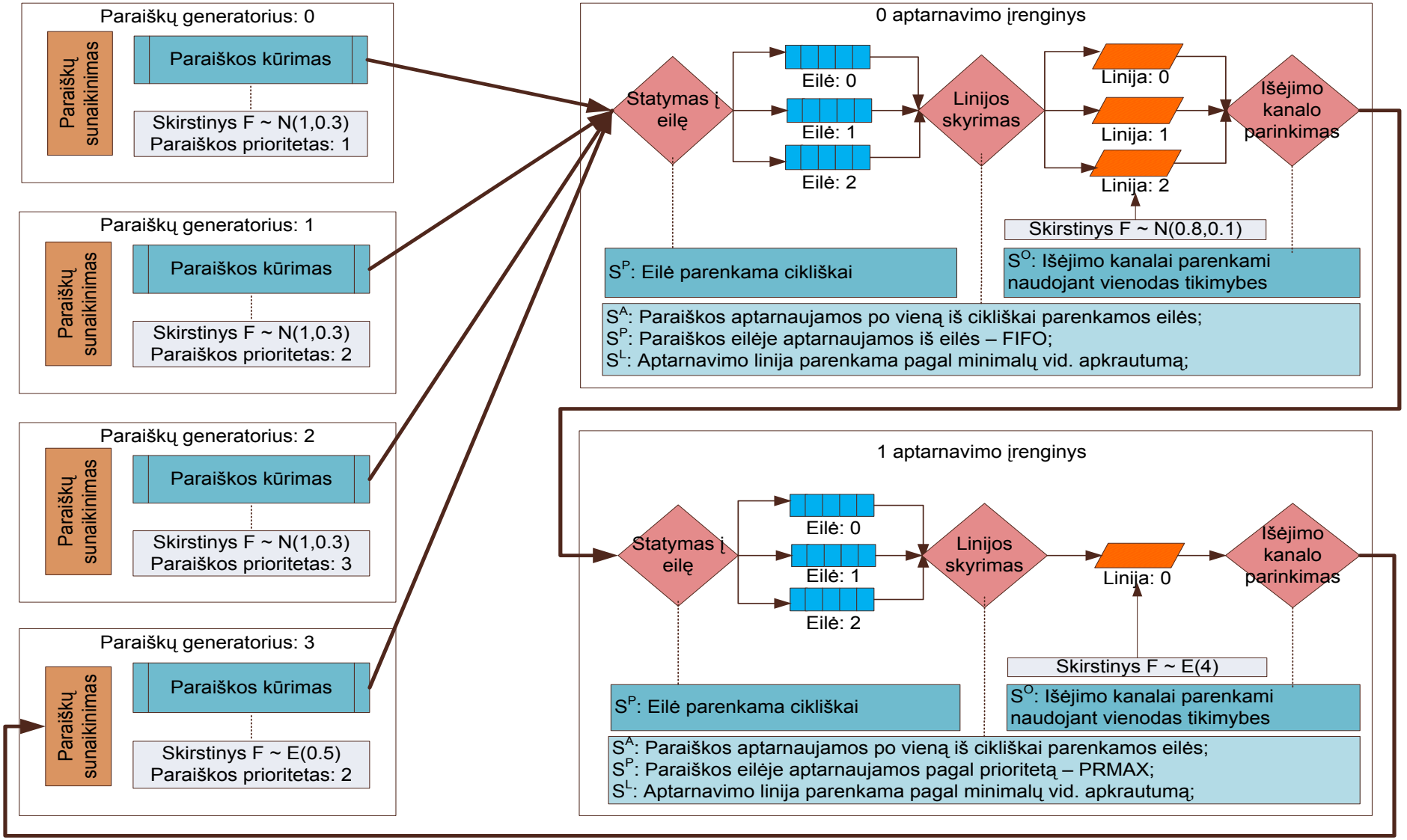
Parametrai	**Įrenginys „0“**	**Įrenginys „1“**
Išėjimo kanalų kiekis (L)	L = 1	
Išėjimo kanalo parinkimo strategija	Atsitiktinai, su vienodomis tikimybėmis	
Eilių kiekis (M)	M = 3	
Eilės parinkimas atėjusiai paraiškai (S^Q)	Cikliškai parenkant eilę	
Maksimalus eilės ilgis (l^Q)	∞	
Paraiškos parinkimo aptarnavimui strategija (S^P)	FIFO – pirma atėjo, pirma išėjo	PRMAX – pagal aukščiausią prioritetą.
Aptarnaujamos eilės pasirinkimo strategija (S^A)	Aptarnaujama viena paraiška iš cikliškai parenkamos eilės	
Aptarnavimo linijos pasirinkimo strategija (S^L)	Pagal minimalų vidutinį linijos apkrautumą	
Aptarnavimo linijų kiekis	N = 3	N = 1
Aptarnavimo trukmės skirstinys (F)	N(0,8;0,1)	E(4)

Modelyje buvo naudojamas aptarnavimo agregatas. Dvi jo kopijos buvo sukurtos su skirtingomis paraiškų parinkimo aptarnavimui ir aptarnavimo linijos parinkimo strategijomis, aptarnavimo linijų skaičiumi, aptarnavimo trukmės skirstiniu ir jo parametrais. Parametrai pateikti 2.9 lentelėje.

Pirmų trijų srauto generatorių laiko intervalo tarp paraiškų atėjimo momento skirstiniai yra N(1;0,3) ir atitinkamai generuojamos skirtingų prioritetų (1, 2, 3) paraiškos. Ketvirtos paraiškų srauto generatoriaus kopijos laiko intervalo tarp paraiškų atėjimo momento skirstinys E(0,5),o paraiškų prioritetas P = 2.

Visi generatoriai savo paraiškas siunčia į aptarnavimo įrenginį „0“. Po aptarnavimo paraiškos siunčiamos į įrenginį „0“, kuriam atlikus aptarnavimą paraiškos išeina iš sistemos ir sunaikinamos. Užsakyta stebėti visus modeliavimo duomenis. Modeliuota buvo 1000 sisteminių laiko vienetų.

Sukurtam modeliui atlikta modeliavimo duomenų statistinė analizė panaudojant sukurtus aprašomosios statistinės analizės modelius sistemoje MathCad ir dispersinės analizės modelį pakete SAS. Gauti rezultatai pateikiami 2.4.2.2. skyrelyje.



2.17 pav. Aptarnavimo sistemos AS2 modelio schema

2.4.2.2. Modelio AS2 statistinės analizės rezultatai

1. Aptarnavimo linijų užimtumas parodo, kiek procentų viso modeliavimo laiko aptarnavimo linija buvo užimta. Įrenginio „0“ visos linijos buvo užimtos daugiau nei 90%, o įrenginio „1“ – 86%.

Užimtumo_Lentelė (Data) =

"Įrenginys"	"Linija"	"Užimtumas (%)"
0	0	93.139
0	1	94.855
0	2	91.656
1	0	85.996

2.18 pav. Įrenginių aptarnavimo linijų užimtumas

2. Paraiškų buvimo laikas eilėje. Įrenginys „0“ turi 3 eiles, kuriose paraiškos vidutiniškai praleidžia 0,35 sisteminio laiko vieneto, o įrenginio „1“ eilėje „2“ vidutinis laikas yra didžiausias – 0.804.

Paraiškų_buvimo_laikas_eilėje (Data) =

"Įrenginys"	"Eilė"	"Imties dydis"	"Vidurkis"	"Dispersija"	"Mediana"
0	0	1164	0.3631	0.4302	0.16989
0	1	1163	0.34964	0.43647	0.14854
0	2	1161	0.33743	0.31518	0.16098
1	0	1162	0.6774	0.84482	0.31841
1	1	1162	0.65355	0.83051	0.30685
1	2	1161	0.80422	1.89631	0.29431

2.19 pav. Paraiškų buvimo laiko eilėje skaitinės charakteristikos

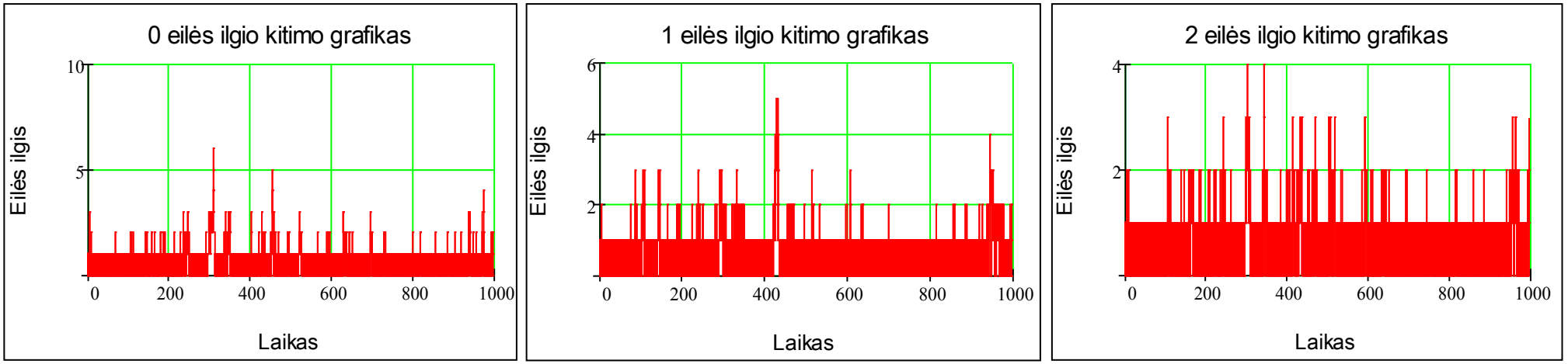
3. Eilių ilgiai. Įrenginio „0“ vidutinis eilės ilgis yra 0,41 paraiškos. Vidutiniškai ilgiausia eilė yra įrenginio „1“ eilė „2“, kurios vidutinis ilgis – 0,93 paraiškos. Maksimalus užfiksuotas eilių ilgis įrenginio „1“ eilėje „0“ – 6 paraiškos, įrenginio „1“ eilėje „2“ – 10 paraiškų.

Eilių_ilgiai (Data) =

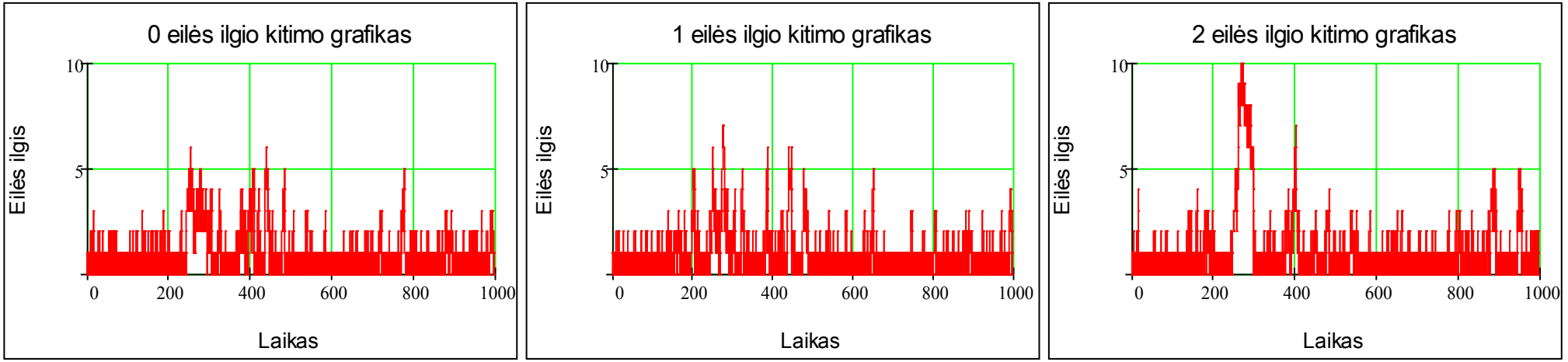
"Įrenginys"	"Eilė"	"Imties dydis"	"MAX"	"Vidurkis"
0	0	2329	6	0.42306
0	1	2327	5	0.40724
0	2	2325	4	0.39429
1	0	2324	6	0.7883
1	1	2324	7	0.76035
1	2	2322	10	0.93528

2.20 pav. Eilių ilgių skaitinės charakteristikos

4. *Eilių ilgių kitimo grafikai* vaizduoja kaip kito eilės ilgis modeliavimo metu. Matome, kad įrenginio „0“ eilės didžiausias ilgis buvo apie 300 sisteminį laiko vienetą. Įrenginio „1“ eilės „2“ didžiausias eilės ilgis buvo apie 280 sisteminį laiko vienetą.



2.21 pav. Eilių ilgių kitimų grafikai (0 įrenginys)



2.22 pav. Eilių ilgių kitimų grafikai (1 įrenginys)

5. Paraiškų buvimo laikas sistemoje iki išėjimo iš aptarnavimo įrenginių.

"Įrenginys"	"Imties dydis"	"MIN"	"MAX"	"Vidurkis"	"Dispersija"
0	3485	0.4673	9.83886	1.15103	0.40601
1	3485	0.61582	10.467	2.10921	1.61814

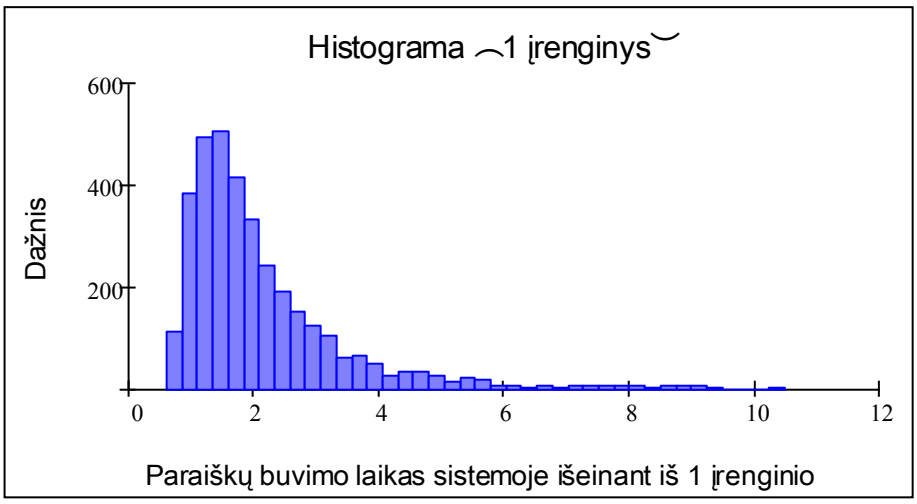
2.23 pav. Paraiškų buvimo laiko sistemoje iki išėjimo iš aptarnavimo įrenginio skaitinės charakteristikos

6. Skirtingo prioritetų paraiškų buvimo laiko sistemoje iki išėjimo iš aptarnavimo įrenginių skiriasi, nes įrenginio „0“ paraiškų pasirinkimo aptarnavimui strategija yra pagal aukštesnį paraiškos prioritetą. Todėl matome skirtingą vidutinį buvimo laiko sistemoje iki išėjimo iš įrenginio „0“. Pirmo prioriteto paraiškų vidutinis buvimo laikas – 1.28, trečio – 1.04.

"Įrenginys"	"Prioritetas"	"Imties dydis"	"Vidurkis"	"Dispersija"	"Mediana"
0	1	989	1.28121	0.93433	0.96717
0	2	1500	1.13627	0.23748	0.99149
0	3	996	1.04401	0.10661	0.95378
1	1	989	2.21318	2.07906	1.75533
1	2	1500	2.10585	1.49577	1.75863
1	3	996	2.01103	1.32437	1.70596

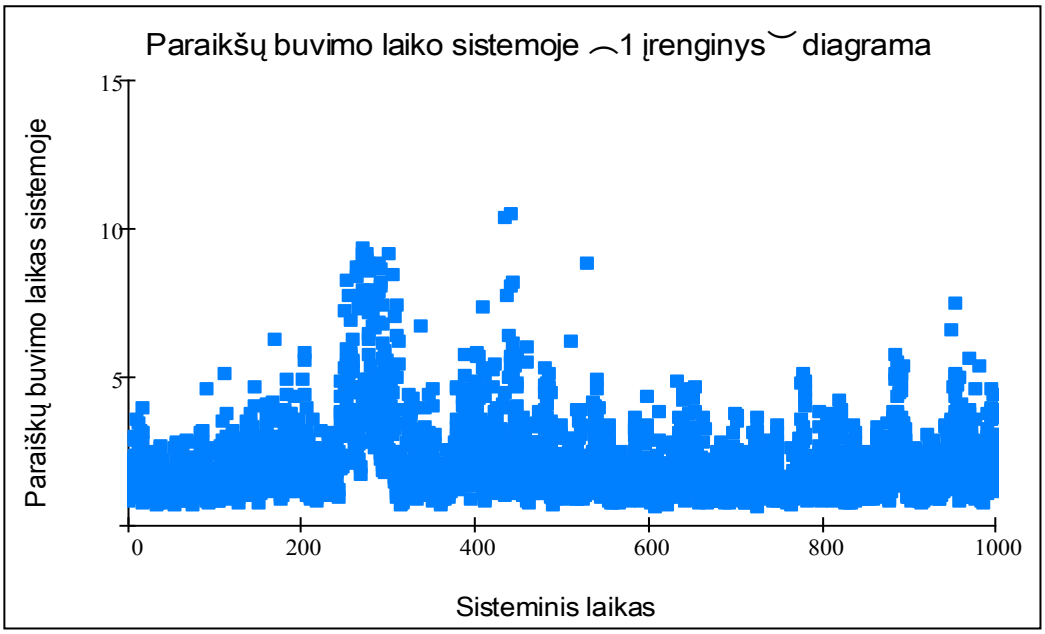
2.24 pav. Skirtingo prioriteto paraiškų buvimo laiko sistemoje iki išėjimo iš įrenginio skaitinės charakteristikos

7. Paraiškų buvimo laiko sistemoje histograma iki išėjimo iš įrenginio „1“ momento (2.25 pav).

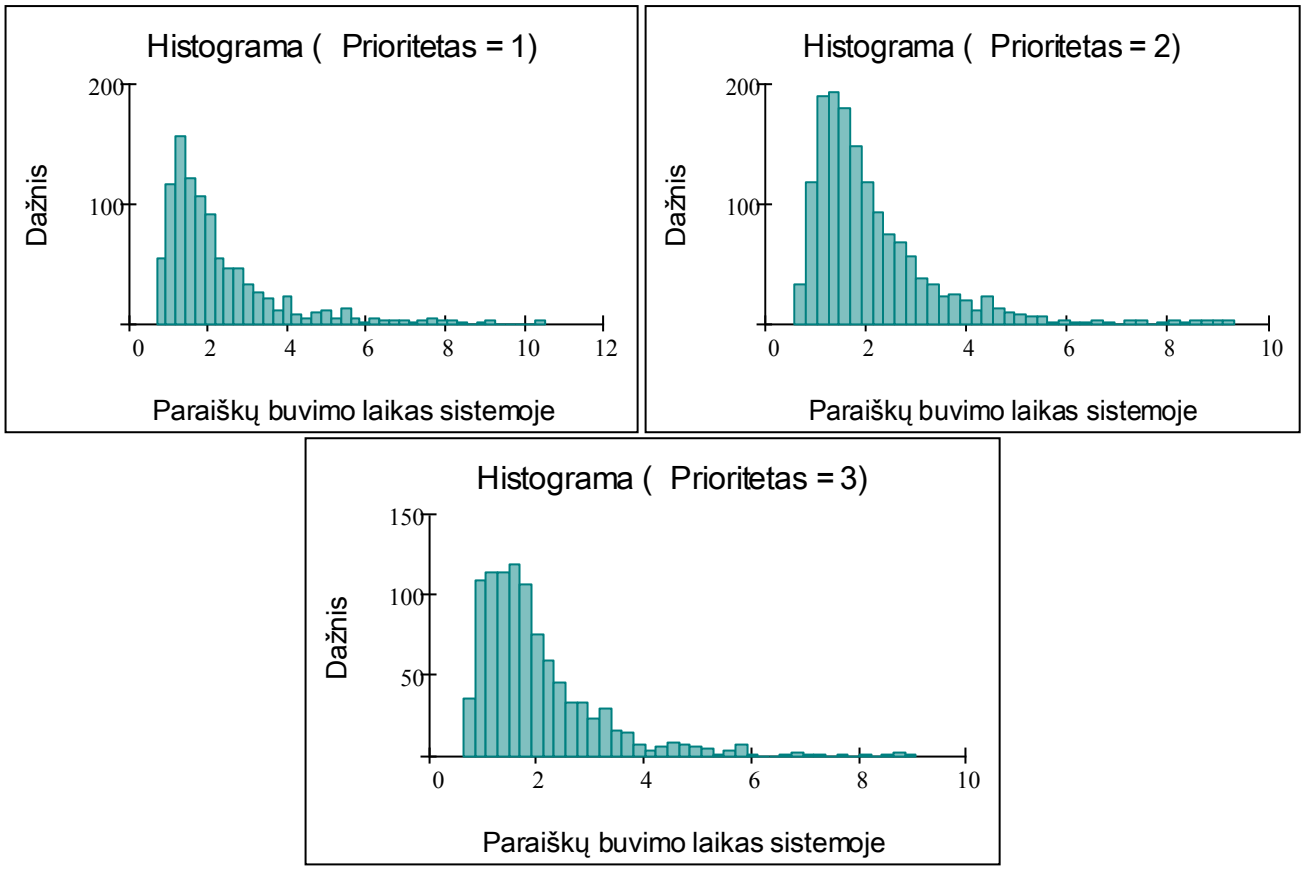


2.25 pav. Paraiškų buvimo laiko sistemoje iki išėjimo iš pirmo įrenginio histograma

2.26 paveiksle vaizduojama modeliavimo metu fiksuotų reikšmių diagrama. Matome, kad didžiausias buvimo laikas praleistas sistemoje yra tų paraiškų, kurių aptarnavimas buvo 300 sisteminį laiką. Tos paraiškos kurios buvo įrenginio „1“ eilėje „2“ ir sudarė ilgiausiai sistemoje buvusių paraiškų skaičių. 2.27 paveiksle pateikiamos skirtingo prioriteto paraiškų buvimo laiko sistemoje histogramos iki išėjimo iš pirmo įrenginio momento.



2.26 pav. Paraiškų buvimo laiko iki išėjimo iš pirmo aptarnavimo įrenginio diagrama



2.27 pav. Skirtingo prioriteto paraiškų buvimo laikų sistemoje iki išėjimo iš pirmo įrenginio histogramos

8. Paraiškų buvimo laiko įrenginyje skaitinės charakteristikos rodo, kad paraiškos trumpesnį laiką buvdavo „1“ įrenginyje, nei įrenginyje „0“ esančios paraiškos.

Paraiškų_buvimo_laikas_įrenginyje (Data) =

"Įrenginys"	"Linja"	"Imties dydis"	"Vidurkis"	"Dispersija"	"Mediana"
0	0	1160	1.15747	0.45386	0.97404
0	1	1185	1.15425	0.43509	0.97711
0	2	1140	1.14114	0.32695	0.96834
1	0	3485	0.95818	1.24948	0.59316

2.28 pav. Paraiškų buvimo laikas įrenginyje.

9. Analizuojant paraiškų buvimo laiką įrenginyje pagal prioritetus, matome, kad įrenginyje „0“ vidutiniškai ilgiausiai buvo paraiškos su žemesniu prioriteto (1 prioriteto – 1,28, 2 prioriteto – 1,31, 3 prioriteto – 1.04), kai tuo tarpu įrenginyje „0“ vidutinis laikas ženkliai nesiskyrė.

Paraiškų_buvimo_laikas_įrenginyje_pagal_prioritetą (Data) =

"Įrenginys"	"Prioritetas"	"Imties dydis"	"Vidurkis"	"Dispersija"	"Mediana"
0	1	989	1.28121	0.93433	0.96717
0	2	1500	1.13627	0.23748	0.99149
0	3	996	1.04401	0.10661	0.95378
1	1	989	0.93197	1.19921	0.56242
1	2	1500	0.96959	1.30032	0.58698
1	3	996	0.96701	1.22188	0.62671

2.29 pav. Skirtingo prioriteto paraiškų buvimo laikas įrenginyje.

Su paketu SAS atlikta modeliavimo duomenų dispersinė analizė, patikrinta hipotezė ar skirtingo prioriteto paraiškų vidutiniai buvimo sistemoje laikai yra vienodi.

2.10 lentelėje matome skirtingo prioriteto paraiškų buvimo laikų sistemoje charakteristikas, kurios atitinka matematinio paketo MathCad pateiktus skaičiavimus.

2.10 lentelė.

Statistinės analizės rezultatai (SAS)

Analizuojamas kintamasis: Paraiškų buvimo laikas sistemoje							
Prioritetas	**Imties dydis**	**Vidurkis**	**Dispersija**	**Standartinis nuokrypis**	**MIN**	**MAX**	**Std Error**
1	989	2.2132	2.0812	1.4426	0.6851	10.4670	0.0459
2	1500	2.1059	1.4968	1.2234	0.6158	9.3259	0.0316
3	996	2.0110	1.3257	1.1514	0.6512	9.0565	0.0365

2.11 lentelė.

Kurskal-Wallis kriterijaus taikymo rezultatai rezultatai (SAS)

Kruskal-Wallis kriterijus	
Chi-Square	6.0033
DF	2
p-reikšmė	0.0497

2.11 lentelėje pateikti tik neparametrinės dispersinės analizės (Kruskalo – Voliso kriterijus) rezultatai, nes duomenys netenkino parametrinės dispersinės analizės prielaidų. Iš 2.11 lentelės matome, kad hipotezė - „stebimo dydžio skirtstiniai k grupėse yra vienodi“, atmetama ($p = 0.0497$), Vadinasi yra statistiškai reikšmingas skirtumas tarp skirtingo prioriteto paraiškų buvimo laiko sistemoje. Yra statistiškai reikšmingas skirtumas tarp skirtingo prioriteto paraiškų vidutinių buvimo laiko sistemoje.

Atlikta dviejų aptarnavimo sistemų modelių statistinė analizė parodė, kad sukurta posistemė pilnai sprendžia darbe suformuluotus uždavinius. Ateityje rekomenduojama išplėsti statistinės analizės galimybės pildant posistemę naujais statistinės analizės modeliais.

2.5. IŠVADOS IR REKOMENDACIJOS

1. Atlikta aptarnavimo sistemų imitacinio modeliavimo sistemų analizė parodė, kad jos turi ribotas modeliavimo duomenų statistinės analizės galimybes. Pagrindinis trūkumas, kad imitacinio modeliavimo sistemos yra uždaros, imitavimo ir statistinės analizės procesai yra stipriai tarpusavyje susiję, kas apsunkina šiuolaikinių matematikos ir statistinės analizės paketų panaudojimą modeliavimo duomenų analizei.

2. Pasiūlytas valdomas matavimo operatorius, kuris įterpiamas į agregatinių modelių imitacijos proceso aprašymą ir leidžia ženkliai sumažinti ryšį tarp imitacijos ir statistinės analizės procesų aprašymo.

3. Panaudojus realiacinę algebrą pasiūlyta duomenų matricos struktūra, kuri leidžia taikyti aptarnavimo sistemų imitacinio modeliavimo duomenų analizei universalius matematikos ir statistinės analizės paketus.

4. Pasiūlyta metodika modeliavimo duomenų matricos transformavimui į statistinei analizei tinkamas matricas ir atlikta jų realizacija pakete MathCad ir SAS.

5. Sukurta aptarnavimo sistemų agregatinė imitacinė modeliavimo programa SIMAS++/ASAIM (apie 2300 C++ programos kodo eilučių), kuri leidžia automatizuoti imitacinių modelių sudarymą, užsakyti norimų stebėti dydžių rinkimą, atlikti imitacinį modeliavimą ir pagal pasiūlytą metodiką sudaro modeliavimo duomenų matricą.

6. Sukurtas valdantis modulis MathCad pakete, kuris apjungia modeliavimo sistemą SIMAS++/ASAIM, universalų matematikos paketą MathCad ir statistinės analizės paketą SAS, kas ženkliai išplėčia imitacinių modelių statistinės analizės galimybes.

7. Atliktas dviejų agregatinių aptarnavimo sistemų modelių tyrimas (aprašomosios statistikos ir dispersinės analizės metodus), parodė, kad sukurtos priemonės pilnai sprendžia darbe suformuluotus uždavinius.

8. Tobulinant aptarnavimo sistemų agregatinių imitacinių modelių statistinės analizės posistemę, siūlome ją papildyti naujais statistinės analizės modeliais bei sukurti juos realizuojančias makro kamandas.

LITERATŪRA

1. H. Lian, Zh. Wan. The Computer Simulation for Queuing System. Proceedings of World Academy of Science, Engineering and Technology volume 23 August 2007.
2. Kobayashi, Hisashi. System modeling and analysis: foundations of system performance evaluation, Prentice Hall, 2009.
3. Варжапетян, А. Г., Имитционное моделирование на GPPS/H, А. Г. Варжапетян; ГУАП. — СПб., 2007. — 384 с.: ил.
4. Nilsen. F. B., Queuing systems: Modeling, analysis and simulation. Department of Informatics, University of Oslo, 1998.
5. I. García, R. Mollá, E. Ramos and M. Fernández. D.E.S.K. Discrete Events Simulation Kernel. European Congress on Computational Methods in Applied Sciences and Engineering. Eccomas 2000.
6. V.Janilionis, I.Tamošaitytė. Imitacinių modelių statistinės analizės klausimu. Matematika ir matematinis modeliavimas. Konferencijos pranešimų medžiaga. 2 knyga. – Kaunas. Technologija, 1997 – 143p.
7. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer. GPSS - 40 Years of development. Proceedings of the 2001 Winter Simulation Conference.
8. R. Fourer, H.I. Gassmann,, J. Ma, R.K. Martin, An XML-Based Schema For Stochastic Programs. Industrial Engineering and Management Sciences, Northwestern University, 2006.
9. Hlupic, V., Simulation Software: A Survey of Academic and Industrial Users. International Journal of Simulation. Vol 1, No 1, 2000.
10. Zhang L. J., Ng Wen Wei J. L., Tay Seng C., Discrete – event simulation of queuing systems. Sixth Youth Science Conference, 2000, Ministry of Education, Singapore.
11. Abu-Taieh, E., El Sheikh, A R., 2007. Commercial Simulation Packages: A Comparative Study. International Journal of Simulation, vol 8, No 2, pp. 66-76
12. H. Pranevičius. Sudėtingų sistemų formalizavimas ir analizė. Kauno technologijos universitetas, 2008.
13. H. Pranevičius. Kompiuterių tinklų protokolų formalusis specifikavimas ir analizė: agregatinis metodas. Kaunas, Technologija, 2003. 177 p.
14. R. Rindzinevičius. Informacijos srautų pasiskirstymo teorija. Kauno technologijos universitetas, 2001. 199 p.
15. G. Der, Br. Everitt. A handbook of statistical analyses using SAS. CRC Press, 2002. 360 p.
16. Ronald W. Larsen, Introduction to Mathcad 13. Prentice Hall, 2006. 418 p.

1 PRIEDAS. MODELIAVIMO UŽDUOTIES PAVYZDYS

```
Aptarnavimo_Agregatu_Skaicius: 4
Srauto_generatoriu_skaicius: 4
LAIKAS: 1000

GENERATORIAUS_NUMERIS: 0
GEN_SKIRSTINYS: 2
GEN_SKIRSTINIO_PAR1: 1
GEN_SKIRSTINIO_PAR2: 0.3
GEN_PRIORITETAS: 1
GEN_NUORODA_I_APTARN_AGR: 0
-------------------------------------------------------------
GENERATORIAUS_NUMERIS: 1
GEN_SKIRSTINYS: 2
GEN_SKIRSTINIO_PAR1: 1
GEN_SKIRSTINIO_PAR2: 0.3
GEN_PRIORITETAS: 2
GEN_NUORODA_I_APTARN_AGR: 0
-------------------------------------------------------------
GENERATORIAUS_NUMERIS: 2
GEN_SKIRSTINYS: 2
GEN_SKIRSTINIO_PAR1: 1
GEN_SKIRSTINIO_PAR2: 0.3
GEN_PRIORITETAS: 3
GEN_NUORODA_I_APTARN_AGR: 0
-------------------------------------------------------------
GENERATORIAUS_NUMERIS: 3
GEN_SKIRSTINYS: 1
GEN_SKIRSTINIO_PAR1: 0.5
GEN_SKIRSTINIO_PAR2: 0.1
GEN_PRIORITETAS: 2
GEN_NUORODA_I_APTARN_AGR: 0
-------------------------------------------------------------
Agregato_numeris: 0
EILIU_KIEKIS: 3
EILIU_ILGIS: -1
Eiles_parinkimas_naujai_paraiskai: 2
Eiles_pasitinkimas_aptarnavimui: 12
Eiles_strategija: 1
Apt_liniju_skaicius: 3
Apt_liniju_strategija: 3
Apt_trukmes_skirstinys: 2
Apt_skirstinio_param1: 0.8
Apt_skirstinio_param2: 0.1
Isejimo_kanalu_kiekis: 1
Isejimo_kanalo_strategija: 1
Isejimo_kanalo_nuoroda: 1

Agregato_numeris: 1
EILIU_KIEKIS: 3
EILIU_ILGIS: -1
Eiles_parinkimas_naujai_paraiskai: 2
Eiles_pasitinkimas_aptarnavimui: 12
Eiles_strategija: 3
Apt_liniju_skaicius: 3
Apt_liniju_strategija: 3
Apt_trukmes_skirstinys: 2
Apt_skirstinio_param1: 0.8
```

```
Apt_skirstinio_param2: 0.1
Isejimo_kanalu_kiekis: 1
Isejimo_kanalo_strategija: 1
Isejimo_kanalo_nuoroda: 2

Agregato_numeris: 2
EILIU_KIEKIS: 3
EILIU_ILGIS: -1
Eiles_parinkimas_naujai_paraiskai: 2
Eiles_pasitinkimas_aptarnavimui: 12
Eiles_strategija: 1
Apt_liniju_skaicius: 3
Apt_liniju_strategija: 3
Apt_trukmes_skirstinys: 2
Apt_skirstinio_param1: 0.8
Apt_skirstinio_param2: 0.1
Isejimo_kanalu_kiekis: 1
Isejimo_kanalo_strategija: 1
Isejimo_kanalo_nuoroda: 3

Agregato_numeris: 3
EILIU_KIEKIS: 3
EILIU_ILGIS: -1
Eiles_parinkimas_naujai_paraiskai: 2
Eiles_pasitinkimas_aptarnavimui: 12
Eiles_strategija: 1
Apt_liniju_skaicius: 1
Apt_liniju_strategija: 1
Apt_trukmes_skirstinys: 1
Apt_skirstinio_param1: 4
Apt_skirstinio_param2: 0
Isejimo_kanalu_kiekis: 1
Isejimo_kanalo_strategija: 1
Isejimo_kanalo_nuoroda: -1
```

2 PRIEDAS. SAS MAKRO KOMANDA „DISPERSINE_ANALIZE“

```
ods html file = "rz.html";
ods graphics on;
options label papersize=A4;

data duomenys;
infile 'out.txt';
input ID Generatorius Laikas Statistika $ Agregato_numeris
Irenginys Prioritetas Reiksme;
run;

 %macro
dispersine(duomenys,stebimas_dydis,Prioritetas,Nuemimo_laikas);
title "Dispersine analize";
data duom1;
set &duomenys;
if Statistika = &stebimas_dydis;
if Laikas > &Nuemimo_laikas;
run;
proc sort data = duom1;
by Agregato_numeris &Prioritetas Laikas;
run;
```

```
    proc freq data = duom1;
     table &Prioritetas;
     title 'Dazniu lentele:';
    run;

    proc univariate data = duom1 normal;
      class &Prioritetas;
      *by &Prioritetas;
      var Reiksme;
      title 'Tikrinama ar normalus:';
    run;

    proc GLM data = duom1;
        title 'Dispersine analize su Scheffe daugialypio palyginimo
kriterijumi: ';
     class &Prioritetas;
     model Reiksme = &Prioritetas;
     means &Prioritetas/ scheffe lines;
     output out=outdat r=liekanos;
      run;

    title 'Liekanu pasiskirstymas';
    proc univariate data = outdat normal;
    var liekanos;
    *histogram liekanos;
    output out= tikrinama PROBN=normalumas;
    *if PROBN>0.05 then ods html exclude all;
    run;
    title 'Neparametrine dispersine analize';
    proc means data=duom1 ndec = 4 n mean var stddev min max stderr;
     class &Prioritetas;
     var Reiksme;
    run;
    proc npar1way wilcoxon data=duom1;
          class &Prioritetas;
          var Reiksme;
         * output out=tes wilcoxon;
       run;

    %mend dispersine;

    %dispersine(duomenys,"EX_S",Prioritetas,0);

    ods html close;
    ods graphics off;
```

3 PRIEDAS. EKSPERIMENTŲ REZULTATAI

1. Pirmo eksperimento rezultatai

Neparametrinė dispersinė analizė

The NPAR1WAY Procedure

Wilcoxon Scores (Rank Sums) for Variable Reiksme Classified by Variable Prioritetas					
Prioritetas	**N**	**Sum of Scores**	**Expected Under H0**	**Std Dev Under H0**	**Mean Score**
1	10109	153513016	153004770	717022.682	15185.7766
2	10154	153574637	153685867	717814.380	15124.5457
3	10007	151063933	151460949	715198.481	15095.8262
Average scores were used for ties.					

Kruskal-Wallis Test	
Chi-Square	0.5569
DF	2
Pr > Chi-Square	0.7570

Užimtumo_Lentelė (Data) =

"Įrenginys"	"Linija"	"Užimtumas (%)"
0	0	14.357
1	0	14.436
2	0	14.146
3	0	14.523
4	0	14.392
5	0	14.262

2. Antro ekperimento rezultatai

Neparametrinė dispersinė analizė

The NPAR1WAY Procedure

Wilcoxon Scores (Rank Sums) for Variable Reiksme Classified by Variable Prioritetas					
Prioritetas	**N**	**Sum of Scores**	**Expected Under H0**	**Std Dev Under H0**	**Mean Score**
1	2017	6185580.0	6088314.50	63860.4416	3066.72286
2	2019	6049973.0	6094351.50	63876.1953	2996.51956
3	2000	5984113.0	6037000.00	63725.1023	2992.05650

Kruskal-Wallis Test	
Chi-Square	2.3264
DF	2
Pr > Chi-Square	0.3125

Užimtumo_Lentelė (Data) =

"Įrenginys"	"Linija"	"Užimtumas (%)"
0	0	27.939
1	0	29.441
2	0	30.117
3	0	29.465
4	0	28.364
5	0	30.857

3. Trečio eksperimento rezultatai

Neparametrinė dispersinė analizė
The NPAR1WAY Procedure

Wilcoxon Scores (Rank Sums) for Variable Reiksme Classified by Variable Prioritetas					
Prioritetas	**N**	**Sum of Scores**	**Expected Under H0**	**Std Dev Under H0**	**Mean Score**
1	2986	13712140.0	13402661.0	115673.347	4592.14334
2	3004	13447580.0	13483454.0	115847.016	4476.55792
3	2986	13129056.0	13402661.0	115673.347	4396.87073

Kruskal-Wallis Test	
Chi-Square	8.5742
DF	2
Pr > Chi-Square	0.0137

Užimtumo_Lentelė (Data) =

"Įrenginys"	"Linija"	"Užimtumas (%)"
0	0	42.326
1	0	44.531
2	0	41.63
3	0	44.087
4	0	43.282
5	0	43.245

4. Ketvirto eksperimento rezultatai

Neparametrinė dispersinė analizė
The NPAR1WAY Procedure

Wilcoxon Scores (Rank Sums) for Variable Reiksme Classified by Variable Prioritetas					
Prioritetas	**N**	**Sum of Scores**	**Expected Under H0**	**Std Dev Under H0**	**Mean Score**
1	3055	13660965.0	13639047.5	115543.733	4471.67430
2	2877	12619315.0	12844366.5	113813.635	4386.27563
3	2996	13578776.0	13375642.0	114995.876	4532.30174

Kruskal-Wallis Test	
Chi-Square	4.7469
DF	2
Pr > Chi-Square	0.0932

Užimtumo_Lentelė (Data) =

"Įrenginys"	"Linija"	"Užimtumas (%)"
0	0	42.376
1	0	40.526
2	0	41.743
3	0	29.399
4	0	29.708
5	0	30.959

4 PRIEDAS. MATHCAD FUNKCIJŲ FAILAS „FUNKCIJOS.XMCD“

```
PjuvisStebDydisPrioritetas( Duomenys, B, Prioritetas) :=
    m ← −1
    Rezultatas ← 0
    for i ∈ 0.. rows( Duomenys) − 1
        if str2vec(Duomenys_{i,3}) = str2vec( B)  if Prioritetas = Duomenys_{i,6}
            m ← m + 1
            for j ∈ 0.. 7
                Rezultatas_{m,j} ← Duomenys_{i,j}
    return Rezultatas

PjuvisPagalAgregata( Duomenys, C, D) :=
    m ← −1
    Rezultatas ← 0
    for i ∈ 0.. rows( Duomenys) − 1
        if Duomenys_{i,3} = C ∧ Duomenys_{i,4} = D
            m ← m + 1
            for j ∈ 0.. 7
                Rezultatas_{m,j} ← Duomenys_{i,j}
    return Rezultatas

PjuvisPagalStebDydisDuomenys( Duomenys, B) :=
    m ← −1
    Rezultatas ← 0
    for i ∈ 0.. rows( Duomenys) − 1
        if str2vec(Duomenys_{i,3}) = str2vec( B)
            m ← m + 1
            for j ∈ 0.. 7
                Rezultatas_{m,j} ← Duomenys_{i,j}
    return Rezultatas

PjuvisPagalAgregataIrEile( Duomenys, C, D) :=
    m ← −1
    Rezultatas ← 0
    for i ∈ 0.. rows( Duomenys) − 1
        if Duomenys_{i,4} = C ∧ Duomenys_{i,5} = D
            m ← m + 1
            for j ∈ 0.. 7
                Rezultatas_{m,j} ← Duomenys_{i,j}
    return Rezultatas
```

```
Eiliu ilgiai( Q) := Rezultatas0,0 ← "Irenginys"
                   Rezultatas0,1 ← "Eilė"
                   Rezultatas0,2 ← "Imties dydis"
                   Rezultatas0,3 ← "MAX"
                   Rezultatas0,4 ← "Vidurkis"
                   F ← Atrinkimas(Q,"Q_ Q")
                   for i ∈ 0,1.. rows( F) − 1
                     Lentele ← Pjuvis(Q,"Q_ Q",Fi,0,Fi,1)
                     Rezultatasi+1,0 ← Fi,0
                     Rezultatasi+1,1 ← Fi,1
                     Rezultatasi+1,2 ← rows( Lentele)
                     Rezultatasi+1,3 ← max(Lentele⟨7⟩)
                     Rezultatasi+1,4 ← Vidutinis_ ilgis ( Lentele)
                   return Rezultatas
```

```
Uzimtumas( Q,B,C) := Data ← Pjuvis(Q,"A_ BU",B,C)
                     i ← 1
                     rez ← 0
                     pirmas ← Data0,2
                     while i < rows( Data)
                       rez ← rez + Datai,2 − Datai−1,2
                       i ← i + 2
                     return rez / (Qrows( Q) −1,2 − Q0,2) ·100
```

```
PjuvisStebDydisIrenginysPrioritetas ( Duomenys,B,Irenginys,Prioritetas) := m ← −1
                                        Rezultatas ← 0
                                        for i ∈ 0.. rows( Duomenys) − 1
                                          if str2vec(Duomenysi,3) = str2vec( B) if Prioritetas = Duomenysi,6 if Irenginys = Duomenysi,4
                                            m ← m + 1
                                            for j ∈ 0.. 7
                                              Rezultatasm,j ← Duomenysi,j
                                        return Rezultatas
```

```
AtkritusiuParaiskuProcentas ( Q) := | a ← rows(PjuvisPagalStebDydisDuomenys (Q, "EX_ Q" ))
                                    | b ← max(Q^⟨0⟩)
                                    | b ← a / max(Q^⟨0⟩)  if a > 0
                                    | b ← 0  if a ≤ 0
                                    | return b·100

Atrinkimaslrenginiu ( Duomenys , A) := | n ← 1
                                       | yra ← 1
                                       | Rezultatas ← PjuvisPagalStebDydisDuomenys ( Duomenys , A)
                                       | RezultatasG_{0,0} ← Rezultatas_{0,4}
                                       | for i ∈ 1.. rows( Rezultatas) − 1
                                       |   a ← Rezultatas_{i,4}
                                       |   for j ∈ 0.. n − 1
                                       |     if RezultatasG_{j,0} = a
                                       |       yra ← 1
                                       |       j ← n − 1
                                       |   if yra ≠ 1
                                       |     RezultatasG_{n,0} ← a
                                       |     n ← n + 1
                                       |   yra ← 0
                                       | RezultatasG ← csort(RezultatasG , 0)
                                       | RezultatasG ← Pildyti( RezultatasG)
                                       | return RezultatasG

Užimtumo| Lentelė ( Duomenys) := | Rezultatas_{0,0} ← "Įrenginys"
                                | Rezultatas_{0,1} ← "Linija"
                                | Rezultatas_{0,2} ← "Užimtumas ⌢%⌣"
                                | F ← Atrinkimas(Duomenys , "A_ BU" )
                                | for i ∈ 0, 1.. rows( F) − 1
                                |   Rezultatas_{i+1,0} ← F_{i,0}
                                |   Rezultatas_{i+1,1} ← F_{i,1}
                                |   Rezultatas_{i+1,2} ← Uzimtumas(Duomenys , F_{i,0}, F_{i,1})
                                | return Rezultatas
```

```
Atrinkimas( Duomenys, A) :=
    n ← 1
    yra ← 1
    Rezultatas ← PjuvisPagalStebDydisDuomenys ( Duomenys, A)
    RezultatasG[0,0] ← Rezultatas[0,4]
    RezultatasG[0,1] ← Rezultatas[0,5]
    for i ∈ 1.. rows( Rezultatas) − 1
        a ← Rezultatas[i,4]
        b ← Rezultatas[i,5]
        for j ∈ 0.. n − 1
            if RezultatasG[j,1] = b ∧ RezultatasG[j,0] = a
                yra ← 1
                j ← n − 1
        if yra ≠ 1
            RezultatasG[n,0] ← a
            RezultatasG[n,1] ← b
            n ← n + 1
        yra ← 0
    RezultatasG ← Rikiavimas(csort(RezultatasG, 0))
    return RezultatasG

Vidutinis_ ilgis ( Duomenys) :=
    laikas ← 0
    rez ← 0
    i ← 1
    while i < rows( Duomenys)
        rez ← rez + (Duomenys[i,2] − Duomenys[i−1,2]) · Duomenys[i−1,7]
        laikas ← laikas + Duomenys[i,2] − Duomenys[i−1,2]
        i ← i + 1
    return rez / Duomenys[rows( Duomenys) −1,2]
```

```
AtrinkimasPrioritetu( Duomenys, A) :=
    n ← 1
    yra ← 1
    Rezultatas ← PjuvisPagalStebDydisDuomenys( Duomenys, A)
    RezultatasG0,0 ← Rezultatas0,4
    RezultatasG0,1 ← Rezultatas0,6
    for i ∈ 1.. rows( Rezultatas) − 1
        a ← Rezultatasi,4
        b ← Rezultatasi,6
        for j ∈ 0.. n − 1
            if RezultatasGj,1 = b ∧ RezultatasGj,0 = a
                yra ← 1
                j ← n − 1
        if yra ≠ 1
            RezultatasGn,0 ← a
            RezultatasGn,1 ← b
            n ← n + 1
        yra ← 0
    RezultatasG ← Rikiavimas(csort(RezultatasG, 0))
    return RezultatasG
```

```
Išrinkti( A, stulpelis) :=
    n ← 1
    m ← 1
    Rezultatas0 ← A0, stulpelis
    for i ∈ 1.. rows( A) − 1
        for j ∈ 0.. m − 1
            n ← 0 if (Ai, stulpelis) = (Rezultatasj)
        if n = 1
            Rezultatasm ← Ai, stulpelis
            m ← m + 1
        n ← 1
    sort( Rezultatas)
```

```
Paraiškų buvimo laikas eilėje ( Q) := | B0,0 ← "Įrenginys"
                                      | B0,1 ← "Eilė"
                                      | B0,2 ← "Imties dydis"
                                      | B0,3 ← "Vidurkis"
                                      | B0,4 ← "Dispersija"
                                      | B0,5 ← "Mediana"
                                      | F ← Atrinkimas(Q, "T_ QU" )
                                      | for i ∈ 0.. rows( F) − 1
                                      |     Lentele ← Pjuvis(Q, "T_ QU" , Fi,0, Fi,1)
                                      |     Bi+1,0 ← Fi,0
                                      |     Bi+1,1 ← Fi,1
                                      |     Bi+1,2 ← rows( Lentele)
                                      |     Bi+1,3 ← mean(Lentele⟨7⟩)
                                      |     Bi+1,4 ← var(Lentele⟨7⟩)
                                      |     Bi+1,5 ← median(Lentele⟨7⟩)
                                      | return B
```

```
Paraiškų buvimo laikas įrenginyje ( Q) := B_{0,0} ← "Įrenginys"
                                         B_{0,1} ← "Linija"
                                         B_{0,2} ← "Imties dydis"
                                         B_{0,3} ← "Vidurkis"
                                         B_{0,4} ← "Dispersija"
                                         B_{0,5} ← "Mediana"
                                         F ← Atrinkimas(Q, "EX_ A")
                                         for i ∈ 0.. rows( F) − 1
                                             Lentele ← Pjuvis(Q, "EX_ A", F_{i,0}, F_{i,1})
                                             B_{i+1,0} ← F_{i,0}
                                             B_{i+1,1} ← F_{i,1}
                                             B_{i+1,2} ← rows( Lentele)
                                             B_{i+1,3} ← mean(Lentele^⟨7⟩)
                                             B_{i+1,4} ← var(Lentele^⟨7⟩)
                                             B_{i+1,5} ← median(Lentele^⟨7⟩)
                                         return B

Pjuvis( Duomenys, B, Agregatas, Apt_ Eile) := m ← −1
                                              Rezultatas ← 0
                                              for i ∈ 0.. rows( Duomenys) − 1
                                                if str2vec(Duomenys_{i,3}) = str2vec( B)  if Agregatas = Duomenys_{i,4}  if Apt_ Eile = Duomenys_{i,5}
                                                    m ← m + 1
                                                    for j ∈ 0.. 7
                                                        Rezultatas_{m,j} ← Duomenys_{i,j}
                                              return Rezultatas
```

```
Laikas_sistemoje_iki_išeinant_iš_įrenginio ( Duomenys) :=
    Rezultatas[0,0] ← "Įrenginys"
    Rezultatas[0,1] ← "Imties dydis"
    Rezultatas[0,2] ← "MIN"
    Rezultatas[0,3] ← "MAX"
    Rezultatas[0,4] ← "Vidurkis"
    Rezultatas[0,5] ← "Dispersija"
    Q ← Išrinkti(Duomenys, 4)
    for i ∈ 0.. rows( Q) − 1
        StatistinesAnalizesMatrica ← PjuvisPagalStebDydiAgregata (Duomenys, "EX_ SA" , i, 0)
        Rezultatas[i+1,0] ← i
        Rezultatas[i+1,1] ← rows(StatistinesAnalizesMatrica^⟨7⟩)
        Rezultatas[i+1,2] ← min(StatistinesAnalizesMatrica^⟨7⟩)
        Rezultatas[i+1,3] ← max(StatistinesAnalizesMatrica^⟨7⟩)
        Rezultatas[i+1,4] ← mean(StatistinesAnalizesMatrica^⟨7⟩)
        Rezultatas[i+1,5] ← var(StatistinesAnalizesMatrica^⟨7⟩)
    return Rezultatas
```

```
PjuvisPrioritetas( Duomenys, B, Agregatas, Apt_ Eile , Prioritetas) :=
    m ← −1
    Rezultatas ← 0
    for i ∈ 0.. rows( Duomenys) − 1
        if str2vec(Duomenys[i,3]) = str2vec( B)  if Agregatas = Duomenys[i,4]  if Apt_ Eile = Duomenys[i,5] ∧ Prioritetas = Duomenys[i,6]
            m ← m + 1
            for j ∈ 0.. 7
                Rezultatas[m,j] ← Duomenys[i,j]
    return Rezultatas
```

```
Paraiškų buvimo laikas įrenginyje pagal prioritetą ( Q) := B0,0 ← "Įrenginys"
                                                          B0,1 ← "Prioritetas"
                                                          B0,2 ← "Imties dydis"
                                                          B0,3 ← "Vidurkis"
                                                          B0,4 ← "Dispersija"
                                                          B0,5 ← "Mediana"
                                                          F ← AtrinkimasPrioritetu(Q, "EX_ A")
                                                          for i ∈ 0.. rows( F) − 1
                                                              Lentele ← PjuvisStebDydisIrenginysPrioritetas(Q, "EX_ A", Fi,0, Fi,1)
                                                              Bi+1,0 ← Fi,0
                                                              Bi+1,1 ← Fi,1
                                                              Bi+1,2 ← rows( Lentele)
                                                              Bi+1,3 ← mean(Lentele⟨7⟩)
                                                              Bi+1,4 ← var(Lentele⟨7⟩)
                                                              Bi+1,5 ← median(Lentele⟨7⟩)
                                                          return B
```

```
Rikiavimas( A) := | n ← 0
                  | k ← 0
                  | while n < rows( A) − 1
                  |     | elementas ← A[n,0]
                  |     | while elementas = A[n,0] ∧ n < rows( A) − 1
                  |     |     | B[k] ← A[n,1]
                  |     |     | n ← n + 1
                  |     |     | k ← k + 1
                  |     | if n = rows( A) − 1 if elementas = A[n,0]
                  |     |     | B[k] ← A[n,1]
                  |     |     | n ← n + 1
                  |     |     | k ← k + 1
                  |     | B ← sort( B)
                  |     | m ← 0
                  |     | while k > 0
                  |     |     | A[n−k,1] ← B[m]
                  |     |     | k ← k − 1
                  |     |     | m ← m + 1
                  | return A
```

```
Paraišku buvimo laikas sistemoje iki išeinant iš įrenginio pagal prioritetą ( Q) := B0,0 ← "Įrenginys"
                                                                                  B0,1 ← "Prioritetas"
                                                                                  B0,2 ← "Imties dydis"
                                                                                  B0,3 ← "Vidurkis"
                                                                                  B0,4 ← "Dispersija"
                                                                                  B0,5 ← "Mediana"
                                                                                  F ← AtrinkimasPrioritetu(Q, "EX_ SA")
                                                                                  for i ∈ 0.. rows( F) − 1
                                                                                      Lentele ← PjuvisStebDydisIrenginysPrioritetas(Q, "EX_ SA", F i,0, F i,1)
                                                                                      B i+1,0 ← F i,0
                                                                                      B i+1,1 ← F i,1
                                                                                      B i+1,2 ← rows( Lentele)
                                                                                      B i+1,3 ← mean(Lentele⟨7⟩)
                                                                                      B i+1,4 ← var(Lentele⟨7⟩)
                                                                                      B i+1,5 ← median(Lentele⟨7⟩)
                                                                                  return B
```

5 PRIEDAS. SIMAS++/ASAIM PROGRAMOS TEKSTAS

„Agregatai.cpp"

```
//---------------------------------------------------------------------------
#pragma hdrstop
#include "Agregatai.h"
#include "Pagrininis.h"

#include "math.h"
#include "time.h"
#include <ctime>    // For time()
#include <cstdlib>  // For srand() and rand()

#include <string>
#include <fstream>
#include <iostream>
using namespace std;
#include <iomanip>
void TAgregatai::Valdymas()
{
 time_t t;
 srand((unsigned) time(&t));
 numeris = 0;
 ofstream out;
 out.open("out.out");
```

```
}
//------------------------------
void TAgregatai::PradineParametruIrBusenuAibesBusena(SAgregatuSarasas *agregatas)
{
  agregatas->Parametrai.isejimo_kanalu_kiekis = 1;
  agregatas->Parametrai.eiliu_kiekis = 1;
  agregatas->Parametrai.maksimalus_eiles_ilgis = -1;
  agregatas->Parametrai.atp_liniju_kiekis = 1;
  agregatas->Parametrai.eiles_strategija = 11;
  agregatas->Parametrai.linijos_strategija = 1;
  agregatas->Parametrai.eiles_parinkimas_naujai_paraiskai = 1;
  agregatas->Parametrai.isijimo_kanalo_strategija = 1;
  agregatas->Parametrai.paraisk_pasir_aptarn_eileje_i = 1;
  for (int i=0; i<Cmax;i++)
  { agregatas->BusenuAibe.linijos_uzemimas[i] = 0;
    agregatas->BusenuAibe.aptarnautu_paraisku_kiekis[i] = 0;
    agregatas->BusenuAibe.prastova[i]=0;
    agregatas->Parametrai.isejimo_kanalai[i]=-1;
    agregatas->BusenuAibe.linijos_apkrautumas[i]=0;
  }
  agregatas->BusenuAibe.paskut_pasirinkta_eile_aptarnavimui = -1;
  agregatas->BusenuAibe.paskut_paskirtos_paraiskos_eile = -1;
  agregatas->BusenuAibe.paskut_paraisk_aptarnavimo_linija = -1;
}
//---------------------------------------------------------------------------
void TAgregatai::PradinesBusenos(SAgregatuSarasas *agregatas)
{for (int i=0; i<Cmax;i++)
  { agregatas->BusenuAibe.linijos_uzemimas[i] = 0;
    agregatas->BusenuAibe.aptarnautu_paraisku_kiekis[i] = 0;
    agregatas->BusenuAibe.prastova[i]=0;
    agregatas->BusenuAibe.linijos_apkrautumas[i]=0;
  }
  agregatas->BusenuAibe.paskut_pasirinkta_eile_aptarnavimui = -1;
  agregatas->BusenuAibe.paskut_paskirtos_paraiskos_eile = -1;
  agregatas->BusenuAibe.paskut_paraisk_aptarnavimo_linija = -1;
}
int TAgregatai::ParaiskosEileseParinkimas(SAgregatuSarasas *agregatas)
{
    //s1=1 - aptarnaujama po vienà ið pasirinktos eilës,
    //s1=2 - pasirinkta eilë aptarnaujama tol, kol baigias,
    //s2=1 - pasirenkama ilgiausia eilë,
    //s2=2 - eilës aptarnaujamos cikliðkai,
    //s2=3 - pasirenkama pirma netuðèia eilë nuo pradþiø,
    //s2=4 - atsitiktinai, naudojant lygias arba nurodytas  tikimybes.
    int pask_eile = agregatas->BusenuAibe.paskut_pasirinkta_eile_aptarnavimui;
    int strategija = agregatas->Parametrai.eiles_strategija;
    int max =0;
    int pasirinkta_eile = -1;
 if (strategija<20)
 {switch(strategija)
  {case 11:
   {
    for (int i = 0; i<agregatas->Parametrai.eiliu_kiekis; i++)
      if (max<agregatas->BusenuAibe.kiekis_eileje[i])
        { max = agregatas->BusenuAibe.kiekis_eileje[i];
          if (max != 0)
           pasirinkta_eile = i;   }
     break;
   };
   case 12:
   {
    if (agregatas->Parametrai.eiliu_kiekis>1)
     { if (pask_eile != -1)
```

```
       { int eile_temp = pask_eile;
         do
         { if (eile_temp == agregatas->Parametrai.eiliu_kiekis)
            eile_temp = -1;
            eile_temp++;
           if (agregatas->BusenuAibe.kiekis_eileje[eile_temp]>0)
            {pasirinkta_eile = eile_temp;
             break;
            }
         } while(eile_temp!=pask_eile);
       }
      else
       { int eile_temp = 0;
           do
            { if (agregatas->BusenuAibe.kiekis_eileje[eile_temp]>0)
               {pasirinkta_eile = eile_temp;
                break;
               }
              else eile_temp++;

            }while (eile_temp!=agregatas->Parametrai.eiliu_kiekis);
       }
    }
    else
    { if (agregatas->BusenuAibe.kiekis_eileje[0]>0) pasirinkta_eile = 0;
      else pasirinkta_eile = -1;
      break;
    }
   break;
 };// case12 pabaiga
 case 13:
  {int i = 0;
   while(i<agregatas->Parametrai.eiliu_kiekis)
   {if (agregatas->BusenuAibe.kiekis_eileje[i]>0)
      {  pasirinkta_eile = i;
         break; }
     i++;
   } break;
  }
 case 14:
  { int i = -1;
    float tikimybe = rand()%100;
    float tarpas = 100/agregatas->Parametrai.eiliu_kiekis;
     while (tikimybe>0)
       {tikimybe = tikimybe - tarpas;
         i++;
         if (agregatas->BusenuAibe.kiekis_eileje[i]>0)
          {
            pasirinkta_eile = i;
            agregatas->BusenuAibe.paskut_pasirinkta_eile_aptarnavimui = i;
          }
       }
   break;
  }
 } //switcho pabaiga
} //if pabaiga
else
{
  switch(strategija)
  {case 21:
    {if (pask_eile!=-1 && agregatas->BusenuAibe.kiekis_eileje[pask_eile]>0)
      {pasirinkta_eile = pask_eile;
      }
```

```
  else
  {for (int i = 0; i<agregatas->Parametrai.eiliu_kiekis; i++)
     if (max<agregatas->BusenuAibe.kiekis_eileje[i])
     { max = agregatas->BusenuAibe.kiekis_eileje[i];
        if (max != 0)
          pasirinkta_eile = i;
     }
   }
  break;
 }
case 22:
{if(pask_eile !=-1)
  {if (agregatas->BusenuAibe.kiekis_eileje[pask_eile]>0)
    { pasirinkta_eile = pask_eile;
      break;
    }
   else
    {int eile = pask_eile+1;
      if (eile>=agregatas->Parametrai.eiliu_kiekis)
        eile = 0;

   while (eile != pask_eile)
    {if (agregatas->BusenuAibe.kiekis_eileje[eile]>0)
      { pasirinkta_eile = eile;
        break;
      }
     else
      { eile++;
        if (eile>=agregatas->Parametrai.eiliu_kiekis)
         eile = 0;
      }
    }
   }
  }
  else
   {for (int i = 0; i<agregatas->Parametrai.eiliu_kiekis; i++)
     {if (agregatas->BusenuAibe.kiekis_eileje[i]>0)
      {pasirinkta_eile = i;
        break;
      }
     }
   break;
   }
 }
case 23:
 {if (pask_eile!=-1)
    {if (agregatas->BusenuAibe.kiekis_eileje[pask_eile]>0)
        { pasirinkta_eile = pask_eile;
          break;
        }
      else
       {for (int i = 0; i<agregatas->Parametrai.eiliu_kiekis; i++)
          {if (agregatas->BusenuAibe.kiekis_eileje[i]>0)
            {pasirinkta_eile = i;
             break;
            }
          }
       }
    }
   else
    {for (int i = 0; i<agregatas->Parametrai.eiliu_kiekis; i++)
       {if (agregatas->BusenuAibe.kiekis_eileje[i]>0)
         {pasirinkta_eile = i;
```

```
                break;
               }
             }
           }
         }
    case 24:
      { int i = -1;
        if (pask_eile!=-1 && agregatas->BusenuAibe.kiekis_eileje[pask_eile]>0)
        {
          pasirinkta_eile = pask_eile;
        }
        else
        {
         float tikimybe = rand()%100;
         float tarpas = 100/agregatas->Parametrai.eiliu_kiekis;
         while (tikimybe>0)
         {tikimybe = tikimybe - tarpas;
           i++;
           if (agregatas->BusenuAibe.kiekis_eileje[i]>0)
            {pasirinkta_eile = i;
              agregatas->BusenuAibe.paskut_pasirinkta_eile_aptarnavimui = i;
            }
          }
         }
       break;
       }
     }//switcho 20 pabaiga.
  }
  agregatas->BusenuAibe.paskut_pasirinkta_eile_aptarnavimui = pasirinkta_eile;
  return pasirinkta_eile;
}
//---------------------------------------------------------------------------
void TAgregatai::ParaiskosPriemimas(ofstream &out,SAgregatuSarasas *agregatas,
SParaiska paraiska)
{
  int eile = EilesParinkimasAtejusiaiParaiskai(agregatas);
  if (eile!=-1)
  {
  int stulpelis = agregatas->BusenuAibe.kiekis_eileje[eile];
  agregatas->Eile[eile][stulpelis].prioritetas = paraiska.prioritetas;
  agregatas->Eile[eile][stulpelis].ID = paraiska.ID;
  agregatas->Eile[eile][stulpelis].sukur_laikas = paraiska.sukur_laikas;
  agregatas->Eile[eile][stulpelis].temp = sisteminis_laikas;
  agregatas->Eile[eile][stulpelis].generatoriaus_ID = paraiska.generatoriaus_ID;
  agregatas->BusenuAibe.kiekis_eileje[eile] = stulpelis + 1;
  if (Statistiku_pasirinkimas[0]==1)
StatistikosRinkimasAgregate(out,paraiska.ID,paraiska.generatoriaus_ID,paraiska.pri
oritetas,AGR_NR,eile,agregatas->BusenuAibe.kiekis_eileje[eile],"Q_Q");
   }
  else
  {StatistikosRinkimas(out,paraiska.ID,AGR_NR,0,paraiska,"EX_Q");
  }
}
//---------------------------------------------------------------------------
int  TAgregatai::EilesParinkimasAtejusiaiParaiskai(SAgregatuSarasas *agregatas)
{int eiliu_kiekis = agregatas->Parametrai.eiliu_kiekis;
  int strategija = agregatas->Parametrai.eiles_parinkimas_naujai_paraiskai;
  int min = Cmax_paraisku;
  int parinktaEile = -1;
  switch (strategija)
  {case 1:       // trumpiausia eile
     {  for (int i=0; i < eiliu_kiekis; i++)
         if (min > agregatas->BusenuAibe.kiekis_eileje[i])
```

```
          {   min = agregatas->BusenuAibe.kiekis_eileje[i];
              parinktaEile = i;
          }
         if (min >= agregatas->Parametrai.maksimalus_eiles_ilgis && agregatas-
>Parametrai.maksimalus_eiles_ilgis != -1)
          parinktaEile=-1;
        break;
     }
   case 2:      // cikliskai
     {if (agregatas->BusenuAibe.paskut_paskirtos_paraiskos_eile==-1)
        {parinktaEile=0;     }
       else
        { if (agregatas->BusenuAibe.paskut_paskirtos_paraiskos_eile + 1<
agregatas->Parametrai.eiliu_kiekis)
            parinktaEile = agregatas->BusenuAibe.paskut_paskirtos_paraiskos_eile +
1;
          else
            parinktaEile = 0;

          if (agregatas->BusenuAibe.kiekis_eileje[parinktaEile]>= agregatas-
>Parametrai.maksimalus_eiles_ilgis &&
              agregatas->Parametrai.maksimalus_eiles_ilgis != -1)
            parinktaEile = -1;}
      break;
     }
    case 3: //aatsitiktine pagal tikimybe
     {  int i = -1;
        float tikimybe = rand()%100;
        float tarpas = 100/agregatas->Parametrai.eiliu_kiekis;
        while (tikimybe>0)
         {tikimybe = tikimybe - tarpas;
          i++;
         }
        parinktaEile = i;
        agregatas->BusenuAibe.paskut_paskirtos_paraiskos_eile = i;
        if (agregatas->BusenuAibe.kiekis_eileje[parinktaEile]>= agregatas-
>Parametrai.maksimalus_eiles_ilgis &&
        agregatas->Parametrai.maksimalus_eiles_ilgis != -1)
        parinktaEile = -1;
        break;
     }
   }
 agregatas->BusenuAibe.paskut_paskirtos_paraiskos_eile = parinktaEile;
 return parinktaEile;
}

//---------------------------------------------------------------------------
int TAgregatai::ParaiskoSalinimasIsEiles(int eile,int apt_irenginys,int paraiska,
SAgregatuSarasas *agregatas)
{
  int eiles_strategija = agregatas->Parametrai.paraisk_pasir_aptarn_eileje_i;
  switch(eiles_strategija)
  {case 1 :      //FIFO
    { agregatas->BusenuAibe.AptarnaujamaParaiska[apt_irenginys]=agregatas-
>Eile[eile][0];
      for (int i = 0; i<agregatas->BusenuAibe.kiekis_eileje[eile];i++)
         agregatas->Eile[eile][i] = agregatas->Eile[eile][i+1];
      agregatas->BusenuAibe.kiekis_eileje[eile] = agregatas-
>BusenuAibe.kiekis_eileje[eile] - 1;
     break;
    }
    case 2:    //LIFO
```

```
    {agregatas->BusenuAibe.AptarnaujamaParaiska[apt_irenginys]=agregatas-
>Eile[eile][agregatas->BusenuAibe.kiekis_eileje[eile]-1];
     agregatas->BusenuAibe.kiekis_eileje[eile] = agregatas-
>BusenuAibe.kiekis_eileje[eile] - 1;
     break;
    }
    case 3:    //PRMAX
    {agregatas->BusenuAibe.AptarnaujamaParaiska[apt_irenginys] = agregatas-
>Eile[eile][paraiska] ;
     for (int i = paraiska; i< agregatas->BusenuAibe.kiekis_eileje[eile]-1; i++)
      agregatas->Eile[eile][i] = agregatas->Eile[eile][i+1];

     agregatas->BusenuAibe.kiekis_eileje[eile]--;
     break;
    }
  }
return 1;
}
//-------------------------------------------------------------
int TAgregatai::EilesTvarkymoMetodas(int eile, SAgregatuSarasas *agregatas)
{int eiles_strategija = agregatas->Parametrai.paraisk_pasir_aptarn_eileje_i;
  int paraiska = -1;
  switch (eiles_strategija)
  {case 1:      //FIFO
     {
      if (agregatas->BusenuAibe.kiekis_eileje[eile] > 0)
       paraiska = 0;
      break;
     }
    case 2:     //LIFO
     {
      if (agregatas->BusenuAibe.kiekis_eileje[eile] > 0)
        paraiska = agregatas->BusenuAibe.kiekis_eileje[eile]-1;
      break;
     }
    case 3:     //PRMAX
     { int prioritetas=-1;
       if (agregatas->BusenuAibe.kiekis_eileje[eile] > 0)
       { for (int i =0; i<agregatas->BusenuAibe.kiekis_eileje[eile]; i++)
          { if (agregatas->Eile[eile][i].prioritetas > prioritetas /* &&
agregatas->Eile[eile][i].sukur_laikas<=ankst_laikas*/)
             { paraiska = i;
               prioritetas = agregatas->Eile[eile][i].prioritetas;
             }
          }
       }
      break;
     }
  }
return paraiska;
}
//---------------------------------------------------------------------------
float TAgregatai::Skirstinys(int i, float par_1, float par_2)
{
   float a=0;
   float b=0;
     a =  float(double(rand())/double(RAND_MAX));
     b =  float(double(rand())/double(RAND_MAX));
   float S = 2;
    switch(i)
  {//Exponentinis
     case 1 :
        while (a==0 || b==0)
```

```
       {a =  float(double(rand())/double(RAND_MAX));
        b =  float(double(rand())/double(RAND_MAX));
       }
      return float (-log(a)/par_1);
     break;

   //Gauso
     case 2 :          //par_2>0
     {float M=-1;
      while (M<0)
      { float V1, V2, R;
        float S = 2;
        while (S>1)
        { V1= float(double(rand())/double(RAND_MAX))*2-1;
          V2= float(double(rand())/double(RAND_MAX))*2-1;
          S = V1*V1+V2*V2;
         }
        R = sqrt(-2*log(S)/S);
        M=R*V1*par_2+par_1;
       }
       return fabs(M);
     }
    //Tolygusis
      case 3:
        { return par_1+a*(par_2-par_1);      //par_2>par_1
        }
    //Puasono
      case 4:
      {S = 1;
       int N = 1;
       par_1 = exp(-par_1+1);
       S = S*float(double(rand()+1)/double(RAND_MAX));
       while(par_1<S)
        { S = S*float(double(rand()+1)/double(RAND_MAX));
          N++;
        }
          return N;                          // par_1=>1
       }
      case 5:
       {  S= a;
         for (int i = 1; i<12; i++)
          {S = S + float(double(rand()+1)/double(RAND_MAX));
          }
         S = S - 6;
        return S;
       }
     }

}
//---------------------------------------------------------------------------
int TAgregatai::AptarnavimoIrenginioParinkimas(SAgregatuSarasas *agregatas)
{ int temp, apt_irenginys = -1;
  int apt_ireng_parinkimo_strategija = agregatas->Parametrai.linijos_strategija;
  int pask_linija = agregatas->BusenuAibe.paskut_paraisk_aptarnavimo_linija;
  if (agregatas->Parametrai.atp_liniju_kiekis == 1 )
   { if (agregatas->BusenuAibe.linijos_uzemimas[0]==0)return 0;
     else return -1;
   }
  switch (apt_ireng_parinkimo_strategija)
   {
    case 1:
    {if (pask_linija != -1)
      {
```

```
        temp = pask_linija+1;
         do
         { if (temp>=agregatas->Parametrai.atp_liniju_kiekis)
           temp = 0;
           if (agregatas->BusenuAibe.linijos_uzemimas[temp] == 0)
            {agregatas->BusenuAibe.paskut_paraisk_aptarnavimo_linija = temp;
             apt_irenginys = temp;
            }
             temp++;
         }while(temp!= pask_linija+1);
       }
      else
       { agregatas->BusenuAibe.paskut_paraisk_aptarnavimo_linija = 0;
        agregatas->BusenuAibe.linijos_uzemimas[0] = 1;
        apt_irenginys = 0;
       }
      break;
     }

    case 2:
    { temp = 0;
      do
       {if (agregatas->BusenuAibe.linijos_uzemimas[temp]==0)
         { apt_irenginys = temp;
           break;
         }
        else temp++;
       }while (temp< agregatas->Parametrai.atp_liniju_kiekis);
     }
     case 3:
     { float min = 1000;
        for (int i = 0; i <agregatas->Parametrai.atp_liniju_kiekis; i++)
        { if (agregatas->BusenuAibe.aptarnautu_paraisku_kiekis>0)
            if (min > (agregatas->BusenuAibe.linijos_apkrautumas[i]/agregatas-
>BusenuAibe.aptarnautu_paraisku_kiekis[i]))
             { if (agregatas->BusenuAibe.linijos_uzemimas[i]==0)
                { min = agregatas->BusenuAibe.linijos_apkrautumas[i]/agregatas-
>BusenuAibe.aptarnautu_paraisku_kiekis[i];
                  apt_irenginys = i;
                }
             }
        } // s=3  - pagal minimalø vidutiná apkrautumà,
       break;
     }
     case 4:
     { double min = -1;
        // pagal ilgiausià paskutinës prastovos trukmæ,
       for (int i = 0; i <agregatas->Parametrai.atp_liniju_kiekis; i++)
       { if (agregatas->BusenuAibe.linijos_uzemimas[i]==0)
           if (sisteminis_laikas-agregatas->BusenuAibe.prastova[i] > min)
             {min = sisteminis_laikas-agregatas->BusenuAibe.prastova[i];
              apt_irenginys = i;
             }
        }
      break;
     }
     case 5:
     { double min = sisteminis_laikas*2;
       for (int i = 0; i <agregatas->Parametrai.atp_liniju_kiekis; i++)
       { if (agregatas->BusenuAibe.linijos_uzemimas[i]==0)
           if (sisteminis_laikas-agregatas->BusenuAibe.prastova[i] < min)
             {min = sisteminis_laikas-agregatas->BusenuAibe.prastova[i];
              apt_irenginys = i;
```

```
            }
        } // pagal trumpiausios paskutinës prastovos trukmæ.
    }
  }
return apt_irenginys;
}
//---------------------------------------------------------------------------
double TAgregatai::ParaiskosAptarnavimas(ofstream &out,int eile, int
apt_irenginys, SAgregatuSarasas *agregatas)
{
 double laikas = Skirstinys(agregatas-
>Parametrai.apt_trukm_linijoje_skirstinys,agregatas->Parametrai.par_1,agregatas-
>Parametrai.par_2);
 int paraiska = EilesTvarkymoMetodas(eile,agregatas);
 int id = agregatas->Eile[eile][paraiska].ID;
 int genID =agregatas->Eile[eile][paraiska].generatoriaus_ID;
 int prioritetas = agregatas->Eile[eile][paraiska].prioritetas;
 if (Statistiku_pasirinkimas[2]==1)
 StatistikosRinkimasAgregate(out,agregatas->Eile[eile][paraiska].ID,agregatas-
>Eile[eile][paraiska].generatoriaus_ID,agregatas-
>Eile[eile][paraiska].prioritetas,AGR_NR,eile,sisteminis_laikas - agregatas-
>Eile[eile][paraiska].temp,"T_QU");
 agregatas->BusenuAibe.linijos_apkrautumas[apt_irenginys] = agregatas-
>BusenuAibe.linijos_apkrautumas[apt_irenginys]+laikas;
 agregatas->BusenuAibe.aptarnautu_paraisku_kiekis[apt_irenginys]++;
 AptarnavimoIrenginioUzemimas(apt_irenginys, agregatas);
 if (Statistiku_pasirinkimas[3]==1)
 StatistikosRinkimasAgregate(out,agregatas->Eile[eile][paraiska].ID,agregatas-
>Eile[eile][paraiska].generatoriaus_ID,agregatas-
>Eile[eile][paraiska].prioritetas,AGR_NR,apt_irenginys,agregatas-
>BusenuAibe.linijos_uzemimas[apt_irenginys],"A_BU");
 ParaiskoSalinimasIsEiles(eile,apt_irenginys,paraiska,agregatas);
 if (Statistiku_pasirinkimas[0]==1)
 StatistikosRinkimasAgregate(out,id,genID,prioritetas,AGR_NR,eile,agregatas-
>BusenuAibe.kiekis_eileje[eile],"Q_Q");
 return laikas;
}
//---------------------------------------------------------------------------
int TAgregatai::ParaiskosPasalinimasIsAptarnavimoIrenginio(int
apt_irenginys,SAgregatuSarasas *agregatas)
{agregatas->BusenuAibe.linijos_uzemimas[apt_irenginys] = 0;
  agregatas->BusenuAibe.prastova[apt_irenginys]= sisteminis_laikas;
}
//---------------------------------------------------------------------------
int TAgregatai::AptarnavimoIrenginioUzemimas(int apt_irenginys,SAgregatuSarasas
*agregatas)
{
 agregatas->BusenuAibe.linijos_uzemimas[apt_irenginys] = 1;
 agregatas->BusenuAibe.prastova[apt_irenginys]= sisteminis_laikas;
}
//---------------------------------------------------------------------------
int TAgregatai::ParaiskosIsejimasIsAgregato(ofstream &out, int
apt_irenginys,SAgregatuSarasas *agregatas)
{
  SParaiska temp;
  int agregatoNumeris = ParaiskosIsejimas(apt_irenginys,agregatas);
  temp = agregatas->BusenuAibe.AptarnaujamaParaiska[apt_irenginys];
  if(agregatoNumeris!=-1)
   {ParaiskosPasalinimasIsAptarnavimoIrenginio(apt_irenginys,agregatas);
    if (Statistiku_pasirinkimas[3]==1)
      StatistikosRinkimasAgregate(out,temp.ID,temp.generatoriaus_ID,
temp.prioritetas,AGR_NR,apt_irenginys,agregatas-
>BusenuAibe.linijos_uzemimas[apt_irenginys],"A_BU");
```

```
    if (Statistiku_pasirinkimas[4] == 1)
     StatistikosRinkimas(out,temp.ID,AGR_NR,apt_irenginys,temp,"EX_A");
    if (Statistiku_pasirinkimas[1] == 1)
     StatistikosRinkimas(out,temp.ID,AGR_NR,apt_irenginys,temp,"EX_SA");
    AGR_NR = agregatoNumeris;
    ParaiskosPriemimas(out,&Agregatas[AGR_NR],temp);
   }
  else
  {Nurasytos_paraiskos[numeris] = temp;
    numeris++;
    if (Statistiku_pasirinkimas[3]==1)
    StatistikosRinkimasAgregate(out,temp.ID,temp.generatoriaus_ID,
temp.prioritetas,AGR_NR,apt_irenginys,0,"A_BU");
    if (Statistiku_pasirinkimas[4] == 1)
     StatistikosRinkimas(out,temp.ID,AGR_NR,apt_irenginys,temp,"EX_A");
    if (Statistiku_pasirinkimas[1] == 1)
     StatistikosRinkimas(out,temp.ID,AGR_NR,apt_irenginys,temp,"EX_SA");
    if (Statistiku_pasirinkimas[5] == 1)
     StatistikosRinkimas(out,temp.ID,AGR_NR,apt_irenginys,temp,"EX_S");
    ParaiskosPasalinimasIsAptarnavimoIrenginio(apt_irenginys,agregatas);
  }
 return agregatoNumeris;
}
//---------------------------------------------------------------------------
int TAgregatai::AgregatoNumeris(int apt_irenginys)
{
  int agregato_Nr=0;
  int temp_liniju_kiekis = Agregatas[0].Parametrai.atp_liniju_kiekis;
  while(apt_irenginys>=temp_liniju_kiekis && agregato_Nr<AgregatuSkaicius)
  { agregato_Nr++;
    temp_liniju_kiekis = temp_liniju_kiekis +
Agregatas[agregato_Nr].Parametrai.atp_liniju_kiekis;
  }
 return agregato_Nr;
}
//---------------------------------------------------------------------------
int TAgregatai::Aptarnavimo_irenginys(int apt_irenginys)
{
  int agregato_Nr=0;
  int temp_liniju_kiekis_senas;
  while(apt_irenginys>0 && agregato_Nr<AgregatuSkaicius)
  { apt_irenginys = apt_irenginys -
Agregatas[agregato_Nr].Parametrai.atp_liniju_kiekis;
    agregato_Nr++;
  }
  if (apt_irenginys<0)
   apt_irenginys = apt_irenginys + Agregatas[agregato_Nr-
1].Parametrai.atp_liniju_kiekis;
 return apt_irenginys;
}
//---------------------------------------------------------------------------
int TAgregatai::ParaiskosIsejimas(int apt_irenginys, SAgregatuSarasas *agregatas)
{ int strategija = agregatas->Parametrai.isijimo_kanalo_strategija;
  int kanalu_skaicius = agregatas->Parametrai.isejimo_kanalu_kiekis;
  int tikimybe = (rand())%100 + 1;
  int kanalas = -1;
  switch(strategija)
  { case 1:                          // atsitiktinai naudojant lygias tikimybes
    { float tarpas = 100/kanalu_skaicius;
      float laikinas = 0;
       while (tikimybe > laikinas)
        { laikinas = laikinas + tarpas;
          kanalas++;
```

```
        }
      break;
    }
   case 2:
    { if (kanalu_skaicius == agregatas->Parametrai.atp_liniju_kiekis)
        kanalas = apt_irenginys;
      else
      {  if (kanalu_skaicius/agregatas->Parametrai.atp_liniju_kiekis>1)
         { float laikinas = 0;
           while (laikinas<apt_irenginys)
            { laikinas = laikinas + kanalu_skaicius/agregatas-
>Parametrai.atp_liniju_kiekis;
              kanalas++;
            }
         }
         else
         { float laikinas = 0;
           while (laikinas<apt_irenginys)
            { laikinas = laikinas + agregatas-
>Parametrai.atp_liniju_kiekis/kanalu_skaicius;
              kanalas++;
            }
         }
      }
    }
  }
return agregatas->Parametrai.isejimo_kanalai[kanalas];
}
//---------------------------------------
void TAgregatai::StatistikosRinkimas(ofstream &out,long ID, int eile_aptIrenginys
,int numeris, SParaiska paraiska, char *statistika)
{
 if (nuemimo_laikas<sisteminis_laikas)
 {int k = 15;
  double t;
  int o = strcmp(statistika,"EX_S");
  int b = strcmp(statistika,"EX_SA");
  if (o==0 || b==0)
  { t = sisteminis_laikas - paraiska.sukur_laikas;}
  else t = sisteminis_laikas-paraiska.temp;
  out.precision(10);
  out<<setw(k)<< ID<<setw(k)<< paraiska.generatoriaus_ID<<setw(k)
<<sisteminis_laikas <<setw(k)<< statistika << setw(k) << eile_aptIrenginys<<
setw(k) << numeris <<setw(k)<<paraiska.prioritetas<< setw(k)<<t<<endl;
 }
}
//-------------------------------------------------------
void TAgregatai::StatistikosRinkimasAgregate(ofstream &out,long ID,int
gener_ID,int prioritetas, int eile_aptIrenginys ,int numeris, int dydis, char
*statistika)
{
 if (nuemimo_laikas<sisteminis_laikas)
 {int k = 15;
  out.precision(10);
  out<<setw(k)<< ID<<setw(k)<<gener_ID<<setw(k)<< sisteminis_laikas <<setw(k)<<
statistika << setw(k) << eile_aptIrenginys<< setw(k) << numeris << setw(k)
<<prioritetas<< setw(k)<<dydis<<endl;
 }
}
//-------------------------------------------
void TAgregatai::StatistikosRinkimasAgregate(ofstream &out, long ID, int
gener_ID,int prioritetas, int eile_aptIrenginys ,int numeris, double dydis, char
*statistika)
```

```
{if (nuemimo_laikas<sisteminis_laikas)
 {int k = 15;
  out.precision(10);
  out<< setw(k)<< ID<<setw(k)<<gener_ID<<setw(k)<<sisteminis_laikas <<setw(k)<<
statistika << setw(k) << eile_aptIrenginys<< setw(k) << numeris <<
setw(k)<<prioritetas<< setw(k) <<dydis<<endl;
 }
}
//---------------------------------------------------------------------------
#pragma package(smart_init)
```

„Agregatai.h"

```
//---------------------------------------------------------------------------

#ifndef AgregataiH
#define AgregataiH
#include <string>
#include <fstream>
#include <iostream>

using namespace std;
#include <iomanip>
const long int Cmax_paraisku = 320000;
const long int Cmax = 100;
  struct SParaiska
 { long ID, // paraiðkos numeris
        prioritetas;  // prioritetas
   double sukur_laikas, // sukûrimo laikas
          temp; // pagalbinis kintamasis
   int pagalbinis;
   int generatoriaus_ID;
 };
 struct SAgrParametrai
 { int isejimo_kanalu_kiekis, // L - iðëjimo kanalø kiekis
       eiliu_kiekis, //M - eiliø kiekis agregate
       atp_liniju_kiekis, // N - aptarnavimo linijø kiekis
       eiles_strategija, // sA - aptarnavimo eilës pasirinkimo strategija
       linijos_strategija, // sL -aptarnavimo linijos pasirinkimo startegija
       eiles_parinkimas_naujai_paraiskai, // sQ - eilës pasirinkimas atvykusiai
paraiðkai strategija
       isijimo_kanalo_strategija, // s0 - iðëjimo kanalo pasirinkimo strategija
       maksimalus_eiles_ilgis,
       apt_trukm_linijoje_skirstinys;
   float par_1,
         par_2;

   int paraisk_pasir_aptarn_eileje_i;//sPi - paraiðkos pasirinkimo aptarnavimui
eilëje Qi strategija

   int isejimo_kanalai[Cmax]; // (masyvas) nurodo i kuriuos agregatus nukreiptos
isejimo linijos
 };

  struct SAgrBusenuAibe
 { int paskut_pasirinkta_eile_aptarnavimui, // paskutinë pasirinkta aptarnavimui
eilë
       paskut_paskirtos_paraiskos_eile, // eilë, á kurià buvo paskirta paskutinë
paraiðka
       paskut_paraisk_aptarnavimo_linija, // La - aptarnavimo linija, á kurià buvo
paskirta paskutinë atëjusi aptarnavimui paraiðka
       kiekis_eileje[Cmax_paraisku], // Qi - (masyvas) paraiðkø kiekis eilëje i
```

```
       linijos_uzemimas[Cmax]; // Lj - (masyvas) uþimta (1) ar ne (0) nurodytu
laiko momentu aptarnavimo linija

   int aptarnautu_paraisku_kiekis[Cmax];
   float linijos_apkrautumas[Cmax]; // Uj - (masyvas) linijos apkrautumas
   float prastova[Cmax];

   double pask_aparn_pabaig_momentas[Cmax]; // Vj - (masyvas) paskutinio
aptarnavimo pabaigos momentas
   SParaiska AptarnaujamaParaiska[Cmax]; // aptarnaujama paraiska
 };

 struct SAgregatuSarasas
  { SAgrParametrai Parametrai;
    SAgrBusenuAibe BusenuAibe;
    SParaiska Eile[10][10000];
  };

 class TAgregatai
{
 private:
  int  AptarnavimoIrenginioUzemimas(int apt_irenginys,SAgregatuSarasas
*agregatas);
  int  EilesTvarkymoMetodas(int eile, SAgregatuSarasas *agregatas);
  int  ParaiskosPasalinimasIsAptarnavimoIrenginio(int
apt_irenginys,SAgregatuSarasas *agregatas);
  int  ParaiskoSalinimasIsEiles(int eile,int apt_irenginys,int paraiska,
SAgregatuSarasas *agregatas);

 public:
  double sisteminis_laikas;
  double nuemimo_laikas;
  int Statistiku_pasirinkimas[10];
  int numeris;
  int AGR_NR;
  SAgregatuSarasas *SarasoPradzia;
  SAgregatuSarasas *DarbinisAgregatas;
    int AgregatuSkaicius;
  SAgregatuSarasas Agregatas[21];
  void PradineParametruIrBusenuAibesBusena(SAgregatuSarasas *agregatas);
  void PradinesBusenos(SAgregatuSarasas *agregatas);
  void Valdymas();
  SParaiska Nurasytos_paraiskos[Cmax_paraisku];
  int  ParaiskosEileseParinkimas(SAgregatuSarasas *agregatas);
  int  EilesParinkimasAtejusiaiParaiskai(SAgregatuSarasas *agregatas);
  double  ParaiskosAptarnavimas(ofstream &out,int eile, int
apt_irenginys,SAgregatuSarasas *agregatas);
  void ParaiskosPriemimas(ofstream &out,SAgregatuSarasas *agregatas, SParaiska
paraiska);
  float Skirstinys(int i, float par_1, float par_2);
  int AptarnavimoIrenginioParinkimas(SAgregatuSarasas *agregatas);
  int ParaiskosIsejimas(int apt_irenginys,SAgregatuSarasas *agregatas);
  void LaikasAptarnavimoIrenginiuose(SAgregatuSarasas *agregatas, int
apt_irenginys, float sisteminis_laikas, float aptarnavimo_laikas);
  void StatistikosRinkimas(ofstream &out,long ID,int eile_aptIrenginys ,int
numeris, SParaiska paraiska,char *statistika);
  void StatistikosRinkimasAgregate(ofstream &out, long ID,int gener_ID,int
prioritetas, int eile_aptIrenginys ,int numeris, int dydis, char *statistika);
  void StatistikosRinkimasAgregate(ofstream &out, long ID,int gener_ID,int
prioritetas, int eile_aptIrenginys ,int numeris, double dydis, char *statistika);
  int ParaiskosIsejimasIsAgregato(ofstream &out,int apt_irenginys,SAgregatuSarasas
*agregatas);
  int AgregatoNumeris(int apt_irenginys);
```

```
  int Aptarnavimo_irenginys(int apt_irenginys);

};
//---------------------------------------------------------------------------
#endif
```

„Pagrindinis.cpp“

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Pagrininis.h"
#include "Apie.h"
#include <string>
#include <fstream>
#include <iostream>
using namespace std;
#include <iomanip>
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
   Skaitliukas = 0;
   modeliavimo_laikas=0;
   AgregObjektas.sisteminis_laikas = 0;
   AgregObjektas.Statistiku_pasirinkimas[0] = 0; // Priemimas i eile - skaicius
   AgregObjektas.Statistiku_pasirinkimas[1] = 0; // Laikas sistemoje iseinant is
agregato
   AgregObjektas.Statistiku_pasirinkimas[2] = 0; // Buvimas eileje -
laikas[trukme]
   AgregObjektas.Statistiku_pasirinkimas[3] = 0; // Aptarnavimo irenginio uzemimas
- 0 / 1
   AgregObjektas.Statistiku_pasirinkimas[4] = 0; // Paraiskos isejimas is agregato
- laikas
   AgregObjektas.Statistiku_pasirinkimas[5] = 0; //Paraiskos laikas aptarnavimo
sistemoje
   Form1->Enabled = false;
   GenDefault.desnis = 2;
   GenDefault.prioritetas = 1;
   GenDefault.aptarn_agregato_nr = 0;
   GenDefault.par_1 = 1;
   GenDefault.par_2 = 0.1;
   GenDefault.laikas_tarpinis = 0;
   Form1->Hide();
}
//---------------------------------------------------------------------------
void TForm1::PradineBusena(SParaiska *Paraiskos)
{
  Paraiskos->sukur_laikas = -1;
  Paraiskos->prioritetas = -1;
  Paraiskos->ID = -1;
  Paraiskos->temp = -1;
}
//-----------------
void TForm1::Modeliavimas()
{ int i,j=0;
  int generatoriaus_numeris=-1;
  int apt_irenginys = -1;
  int agregatas = -1;
```

```
  int eile = -1;
  int pabaiga = -1;
  AgregObjektas.sisteminis_laikas=0;
  AgregObjektas.nuemimo_laikas = 0;
  AgregObjektas.numeris = 0;
  Skaitliukas=0;
  SParaiska temp;

  ofstream out;
  out.open("out.txt",ios::out );
  for (i = 0; i<kiekis; i++)
   paraisku_laikas[i] = -1;
  AgregObjektas.Valdymas();
  for (i = 0; i<n; i++)
   {  paraisku_laikas[i] =
AgregObjektas.Skirstinys(GenParamMasyvas[i].desnis,GenParamMasyvas[i].par_1,GenPar
amMasyvas[i].par_2);
      paraisku_laikas[i] = float(double(rand())/double(RAND_MAX)*2);
   }
  while (AgregObjektas.sisteminis_laikas < modeliavimo_laikas)
   { double min = modeliavimo_laikas;
     for (i=0; i<kiekis;i++)
      if (paraisku_laikas[i] < modeliavimo_laikas )
        if ( paraisku_laikas[i] < min && paraisku_laikas[i]!=-1)
         { min = paraisku_laikas[i];
           generatoriaus_numeris = i;
         }
         else
         {  }
      else pabaiga = 1;
      if (pabaiga ==1) break;
      AgregObjektas.sisteminis_laikas = min;
     if(generatoriaus_numeris<n)
     {  int skaicusIkiIrenginio =0;
        for ( int i = 0;
i<GenParamMasyvas[generatoriaus_numeris].aptarn_agregato_nr; i++)
          skaicusIkiIrenginio = skaicusIkiIrenginio +
AgregObjektas.Agregatas[i].Parametrai.atp_liniju_kiekis;

        AgregObjektas.AGR_NR =
GenParamMasyvas[generatoriaus_numeris].aptarn_agregato_nr;
AgregObjektas.ParaiskosPriemimas(out,&AgregObjektas.Agregatas[GenParamMasyvas[gene
ratoriaus_numeris].aptarn_agregato_nr],Paraiskos_kurimas(AgregObjektas.sisteminis_
laikas,generatoriaus_numeris));
        eile =
AgregObjektas.ParaiskosEileseParinkimas(&AgregObjektas.Agregatas[GenParamMasyvas[g
eneratoriaus_numeris].aptarn_agregato_nr]);
        apt_irenginys =
AgregObjektas.AptarnavimoIrenginioParinkimas(&AgregObjektas.Agregatas[GenParamMasy
vas[generatoriaus_numeris].aptarn_agregato_nr]);
         if (apt_irenginys != -1)
          if(eile!= -1)
           {
             paraisku_laikas[skaicusIkiIrenginio+apt_irenginys+n]=
AgregObjektas.sisteminis_laikas +
AgregObjektas.ParaiskosAptarnavimas(out,eile,apt_irenginys,&AgregObjektas.Agregata
s[GenParamMasyvas[generatoriaus_numeris].aptarn_agregato_nr]);
           }
         if(AgregObjektas.sisteminis_laikas<modeliavimo_laikas)
          {
           paraisku_laikas[generatoriaus_numeris] =
AgregObjektas.sisteminis_laikas +
```

```
AgregObjektas.Skirstinys(GenParamMasyvas[generatoriaus_numeris].desnis,GenParamMas
yvas[generatoriaus_numeris].par_1,GenParamMasyvas[generatoriaus_numeris].par_2);
          }
     }
     else
     { int aptarn_temp=0; // rodo aptarnavimo irneginiu skaiciu iki atitinkamo
agregato
       int nuorodaIkitaAgregata;
       agregatas = AgregObjektas.AgregatoNumeris(generatoriaus_numeris-n);
       AgregObjektas.AGR_NR = agregatas;
       int aptarnaves_irenginys =
AgregObjektas.Aptarnavimo_irenginys(generatoriaus_numeris-n);
       nuorodaIkitaAgregata =
AgregObjektas.ParaiskosIsejimasIsAgregato(out,aptarnaves_irenginys,&AgregObjektas.
Agregatas[agregatas]);
       AgregObjektas.AGR_NR = agregatas;
       eile =
AgregObjektas.ParaiskosEileseParinkimas(&AgregObjektas.Agregatas[agregatas]);
//tikrina aptarnavusi irengini,
       //ar neturi daugiau neaptarnautu paraisku ir parenka eile, kurioje yra
paraisku
       apt_irenginys =
AgregObjektas.AptarnavimoIrenginioParinkimas(&AgregObjektas.Agregatas[agregatas]);
// suranda neuzimta irengini.
       for (int i = 0; i<agregatas; i++)
              aptarn_temp = aptarn_temp +
AgregObjektas.Agregatas[i].Parametrai.atp_liniju_kiekis;
       if (apt_irenginys != -1 && eile!= -1)
           {
             paraisku_laikas[generatoriaus_numeris] = -1;
             paraisku_laikas[apt_irenginys+n+aptarn_temp]=
AgregObjektas.sisteminis_laikas +
AgregObjektas.ParaiskosAptarnavimas(out,eile,apt_irenginys,&AgregObjektas.Agregata
s[agregatas]);
           }
       else paraisku_laikas[generatoriaus_numeris] = -1;
       //------------- nuejusios paraiskos i kita agregata aptarnavimas ------
       if (nuorodaIkitaAgregata != -1 && nuorodaIkitaAgregata !=
AgregObjektas.AGR_NR)
        { aptarn_temp = 0;
          AgregObjektas.AGR_NR = nuorodaIkitaAgregata;
          eile =
AgregObjektas.ParaiskosEileseParinkimas(&AgregObjektas.Agregatas[nuorodaIkitaAgreg
ata]);
          if (eile!=-1)
          apt_irenginys =
AgregObjektas.AptarnavimoIrenginioParinkimas(&AgregObjektas.Agregatas[nuorodaIkita
Agregata]);
           for (int i = 0; i<nuorodaIkitaAgregata; i++)
              aptarn_temp = aptarn_temp +
AgregObjektas.Agregatas[i].Parametrai.atp_liniju_kiekis;
           if (apt_irenginys != -1)
             if (eile!= -1)
               paraisku_laikas[apt_irenginys+n+aptarn_temp]=
AgregObjektas.sisteminis_laikas +
AgregObjektas.ParaiskosAptarnavimas(out,eile,apt_irenginys,&AgregObjektas.Agregata
s[nuorodaIkitaAgregata]);
             else
                if (nuorodaIkitaAgregata!= aptarnaves_irenginys)
paraisku_laikas[generatoriaus_numeris] = -1;
        }
     }
  Form1->CGauge1->MaxValue = modeliavimo_laikas;
```

```
  Form1->CGauge1->Progress = AgregObjektas.sisteminis_laikas;
  }
  Form1->CGauge1->Progress = modeliavimo_laikas;
  ShowMessage("Sukurtø paraiðkø kiekis:" +IntToStr(Skaitliukas) + '\n' +
"Aptarnautø paraiðkø kiekis:" +IntToStr(AgregObjektas.numeris));
  out.close();
}
//---------------------------------

SParaiska TForm1::Paraiskos_kurimas(double laikas, int gen_numeris)
{  SParaiska paraiska;
   paraiska.ID = Skaitliukas;
   Skaitliukas++;
   paraiska.prioritetas = GenParamMasyvas[gen_numeris].prioritetas;
   paraiska.sukur_laikas = laikas;
   paraiska.temp = laikas;
   paraiska.pagalbinis = laikas;
   paraiska.generatoriaus_ID = gen_numeris;
   return paraiska;
}
 AnsiString TForm1::ResultFileName()
{
   SaveDialog1->Filter = "(*.tsk)|*.tsk";
   if (SaveDialog1->Execute()) return SaveDialog1->FileName;
   else return "0";
}
//---------- OPEN dialogas -----------------------------------------------------
  // Metodas dialogo budu padeda issirinkti norima atidaryti faila
 AnsiString TForm1::DataFileName()
{  OpenDialog1->Filter = "Text failai(*.TSK)|*.TSK";
   if (OpenDialog1->Execute() && FileExists(OpenDialog1->FileName))
     return OpenDialog1->FileName;
   else return "0";
}
//---------------------------------------------------------------------
int TForm1::ReadFromFile(const char *fn)
{int gerasFailas = 1;
 ifstream in(fn);
 string tempString;
 string temp_int;
 in>>tempString>>temp_int; // "Aptarnavimo_Agregatu_Skaicius:"
  if (tempString.compare("Aptarnavimo_Agregatu_Skaicius:")==0)
   AgregObjektas.AgregatuSkaicius = StrToInt(temp_int.c_str());
  else
   { ShowMessage ("Blogas duomenu failas");
     gerasFailas = 0; }
  in>>tempString>>temp_int; // "Srauto_generatoriu_skaicius"
  if (tempString.compare("Srauto_generatoriu_skaicius:")==0)
   n = StrToInt(temp_int.c_str());
  else
   { ShowMessage ("Blogas duomenu failas");
     gerasFailas = 0; }
  in>>tempString>>temp_int; // "Laikas"
  if (tempString.compare("LAIKAS:")==0)
   modeliavimo_laikas = StrToInt(temp_int.c_str());
  else
   { ShowMessage ("Blogas modeliavimo laikas");
     gerasFailas = 0;
   }
  if (gerasFailas)
 {
  int i = 0;
  while (i < n)
```

```
  {
   in>>tempString>>temp_int;
   in>>tempString>>temp_int;
   GenParamMasyvas[i].desnis = StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   GenParamMasyvas[i].par_1 = StrToFloat(temp_int.c_str());
   in>>tempString>>temp_int;
   GenParamMasyvas[i].par_2 = StrToFloat(temp_int.c_str());
   in>>tempString>>temp_int;
   GenParamMasyvas[i].prioritetas = StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   GenParamMasyvas[i].aptarn_agregato_nr = StrToInt(temp_int.c_str());
   in>>tempString;
   i++;
  }
  for (i = 0; i<AgregObjektas.AgregatuSkaicius; i++)
  {
   AgregObjektas.PradineParametruIrBusenuAibesBusena(&AgregObjektas.Agregatas[i]);
   in>>tempString>>temp_int;
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.eiliu_kiekis =
StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.maksimalus_eiles_ilgis =
StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.eiles_parinkimas_naujai_paraiskai =
StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.eiles_strategija =
StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.paraisk_pasir_aptarn_eileje_i =
StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.atp_liniju_kiekis =
StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.linijos_strategija =
StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.apt_trukm_linijoje_skirstinys =
StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.par_1 = StrToFloat(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.par_2 = StrToFloat(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.isejimo_kanalu_kiekis =
StrToInt(temp_int.c_str());
   in>>tempString>>temp_int;
   AgregObjektas.Agregatas[i].Parametrai.isijimo_kanalo_strategija =
StrToInt(temp_int.c_str());
   for (int j = 0;
j<AgregObjektas.Agregatas[i].Parametrai.isejimo_kanalu_kiekis;j++)
     {
      in>>tempString>>temp_int;
      AgregObjektas.Agregatas[i].Parametrai.isejimo_kanalai[j] =
StrToInt(temp_int.c_str());
     }
   }
  }
return gerasFailas;
```

```
}
//------------------------------
void TForm1::WriteToFile(const char *fn)
{
  string tempString = "";
  ofstream out;
  out.open(fn);
  out<<"Aptarnavimo_Agregatu_Skaicius: "<<AgregObjektas.AgregatuSkaicius<<endl;
  out<<"Srauto_generatoriu_skaicius: "<<n<<endl;
  out<<"LAIKAS: "<< modeliavimo_laikas <<endl;
  out<<endl;
  for (int j = 0; j<n; j++)
  {out<<"GENERATORIAUS_NUMERIS: "<< j<<endl;
   out<<"GEN_SKIRSTINYS: "<< GenParamMasyvas[j].desnis<<endl;
   out<<"GEN_SKIRSTINIO_PAR1: "<< GenParamMasyvas[j].par_1<<endl;
   out<<"GEN_SKIRSTINIO_PAR2: "<< GenParamMasyvas[j].par_2<<endl;
   out<<"GEN_PRIORITETAS: "<< GenParamMasyvas[j].prioritetas<<endl;
   out<<"GEN_NUORODA_I_APTARN_AGR: "<<
GenParamMasyvas[j].aptarn_agregato_nr<<endl;
   out<<"-----------------------------------------------------------------"<<endl;
  }
  int i = 0;
  while (i<AgregObjektas.AgregatuSkaicius)
  { out<<"Agregato_numeris: "<< i<<endl;
    out<<"EILIU_KIEKIS: "<<
AgregObjektas.Agregatas[i].Parametrai.eiliu_kiekis<<endl;
    out<<"EILIU_ILGIS: "<<
AgregObjektas.Agregatas[i].Parametrai.maksimalus_eiles_ilgis<<endl;
    out<<"Eiles_parinkimas_naujai_paraiskai: "<<
AgregObjektas.Agregatas[i].Parametrai.eiles_parinkimas_naujai_paraiskai<<endl;
    out<<"Eiles_pasitinkimas_aptarnavimui: "<<
AgregObjektas.Agregatas[i].Parametrai.eiles_strategija<<endl;
    out<<"Eiles_strategija: "<<
AgregObjektas.Agregatas[i].Parametrai.paraisk_pasir_aptarn_eileje_i<<endl;
    out<<"Apt_liniju_skaicius: "<<
AgregObjektas.Agregatas[i].Parametrai.atp_liniju_kiekis<<endl;
    out<<"Apt_liniju_strategija: "<<
AgregObjektas.Agregatas[i].Parametrai.linijos_strategija<<endl;
    out<<"Apt_trukmes_skirstinys: "<<
AgregObjektas.Agregatas[i].Parametrai.apt_trukm_linijoje_skirstinys<<endl;
    out<<"Apt_skirstinio_param1: "<<
AgregObjektas.Agregatas[i].Parametrai.par_1<<endl;
    out<<"Apt_skirstinio_param2: "<<
AgregObjektas.Agregatas[i].Parametrai.par_2<<endl;
    out<<"Isejimo_kanalu_kiekis: "<<
AgregObjektas.Agregatas[i].Parametrai.isejimo_kanalu_kiekis<<endl;
    out<<"Isejimo_kanalo_strategija: "<<
AgregObjektas.Agregatas[i].Parametrai.isijimo_kanalo_strategija<<endl;
    int j = 0;
    for (j = 0; j<AgregObjektas.Agregatas[i].Parametrai.isejimo_kanalu_kiekis;j++)
     {
      out<<"Isejimo_kanalo_nuoroda: "<<
AgregObjektas.Agregatas[i].Parametrai.isejimo_kanalai[j]<<endl;
     }
    out<<endl;
    i++;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::SaugotiClick(TObject *Sender)
{
 WriteToFile(ResultFileName().c_str());
}
```

```
//---------------------------------------------------------------------------
void TForm1::Vartotojo_sasaja()
{edLaikas->Text = modeliavimo_laikas;
 ListBox1->Clear();
 ComboBox8->Clear();
 for (int i = 0; i<AgregObjektas.AgregatuSkaicius; i++)
   { ListBox1->Items->Add(IntToStr(i));
     ComboBox8->Items->Add(IntToStr(i));
   }
 ListBox3->Clear();
 for (int i = 0; i<n; i++)
   ListBox3->Items->Add(IntToStr(i));
 if (ListBox1->Items->Count>0)
  Pildyti(0);
 if (ListBox3->Items->Count>0)
  PildytiGeneratorius(0);
 ListBox1->ItemIndex = 0;
 ListBox3->ItemIndex = 0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ListBox1Click(TObject *Sender)
{  ListBox2->Clear();
   int m = ListBox1->ItemIndex;
   Pildyti(m);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::AtidarytiClick(TObject *Sender)
{
  if( ReadFromFile(DataFileName().c_str()))
 Vartotojo_sasaja();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::edEiliuKiekisExit(TObject *Sender)
{
  int m = ListBox1->ItemIndex;
  try
  {AgregObjektas.Agregatas[m].Parametrai.eiliu_kiekis = edEiliuKiekis-
>Text.ToInt();
   Pildyti(m);
  }catch (...)
  {ShowMessage("Áveskite sveikà skaièiø!" );}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::edEiliuIlgisExit(TObject *Sender)
{
  int m = ListBox1->ItemIndex;
  try
  {AgregObjektas.Agregatas[m].Parametrai.maksimalus_eiles_ilgis = edEiliuIlgis-
>Text.ToInt();
   Pildyti(m);
  }catch (...)
  {ShowMessage("Áveskite sveikà skaièiø!" );}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::edAptarnavimoLinijuSkaiciusExit(TObject *Sender)
{
  int l1 = ListBox1->ItemIndex;
  if (ListBox1->Items->Count==1) l1=0;
  try
  {AgregObjektas.Agregatas[l1].Parametrai.atp_liniju_kiekis =
edAptarnavimoLinijuSkaicius->Text.ToInt();
  Pildyti(l1);
  }catch (...)
```

```
  {ShowMessage("Áveskite sveikà skaièiø!" );}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ListBox2Click(TObject *Sender)
{
 int l1 = ListBox1->ItemIndex;
 if (ListBox1->Items->Count==1 || l1==-1) l1=0;
 int l2 = ListBox2->ItemIndex;
 edNuorodaKeisti->Text =
AgregObjektas.Agregatas[l1].Parametrai.isejimo_kanalai[l2];
}
//---------------------------------------------------------------------------
void __fastcall TForm1::KeistiClick(TObject *Sender)
{
 int l1 = ListBox1->ItemIndex;
 int l2 = ListBox2->ItemIndex;
 try{
 AgregObjektas.Agregatas[l1].Parametrai.isejimo_kanalai[l2] = edNuorodaKeisti-
>Text.ToInt();
 ListBox2->Clear();
 for (int i = 0;
i<AgregObjektas.Agregatas[l1].Parametrai.isejimo_kanalu_kiekis;i++)
   ListBox2->Items-
>Add(AgregObjektas.Agregatas[l1].Parametrai.isejimo_kanalai[i]);
 }
 catch(...)
 {
  ShowMessage("Klaida: nurodykite skaièiø!");
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox1Change(TObject *Sender)
{
   int l1 = ListBox1->ItemIndex;
   if (ListBox1->Items->Count==1) l1=0;
   AgregObjektas.Agregatas[l1].Parametrai.apt_trukm_linijoje_skirstinys =
ComboBox1->ItemIndex+1;
    switch (AgregObjektas.Agregatas[l1].Parametrai.apt_trukm_linijoje_skirstinys)
   { case 1:
     {
       edSkirstinioParametras2->Enabled = false;
       if(StrToFloat(edSkirstinioParametras1->Text) <1)
        ShowMessage("Neteisingas parametras 1!");
      break;
     };
      case 2:
     {
       edSkirstinioParametras2->Enabled = true;
       if(StrToFloat(edSkirstinioParametras1->Text) <0)
        ShowMessage("Neteisingas parametras 1!");
       if(StrToFloat(edSkirstinioParametras2->Text) <0)
        ShowMessage("Neteisingas parametras 2!");
      break;
     };
      case 3:
     {
       edSkirstinioParametras2->Enabled = true;
       if(StrToFloat(edSkirstinioParametras1->Text) <0)
        ShowMessage("Neteisingas parametras 1!");
       if(StrToFloat(edSkirstinioParametras2->Text) <0)
        ShowMessage("Neteisingas parametras 2!");
       if(StrToFloat(edSkirstinioParametras2-
>Text)<=StrToFloat(edSkirstinioParametras1->Text))
```

```
        ShowMessage("Parametras 2 turi bûti didesnis uþ Parametrà 1!");
       break;
      };
      }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox2Change(TObject *Sender)
{
   int l1 = ListBox1->ItemIndex;
   if (ListBox1->Items->Count==1) l1=0;

   int index = ComboBox2->ItemIndex;

   if (RadioButton1->Checked==true)
   AgregObjektas.Agregatas[l1].Parametrai.eiles_strategija = ComboBox2-
>ItemIndex+21;
   if (RadioButton2->Checked==true)
   AgregObjektas.Agregatas[l1].Parametrai.eiles_strategija = ComboBox2-
>ItemIndex+11;

}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox3Change(TObject *Sender)
{
   int l1 = ListBox1->ItemIndex;
   if (ListBox1->Items->Count==1) l1=0;

   AgregObjektas.Agregatas[l1].Parametrai.paraisk_pasir_aptarn_eileje_i =
ComboBox3->ItemIndex+1;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox4Change(TObject *Sender)
{
   int l1 = ListBox1->ItemIndex;
   AgregObjektas.Agregatas[l1].Parametrai.eiles_parinkimas_naujai_paraiskai =
ComboBox4->ItemIndex+1;
   Pildyti(l1);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox5Change(TObject *Sender)
{
   int l1 = ListBox1->ItemIndex;
   AgregObjektas.Agregatas[l1].Parametrai.linijos_strategija  = ComboBox5-
>ItemIndex+1;
   Pildyti(l1);
}
//---------------------------------------------------------------------------
void TForm1::PildytiGeneratorius(int m)
{
   AnsiString Skirstiniai[5];
   int k;
   if (GenParamMasyvas[m].desnis != -1)
   { ComboBox6->ItemIndex = GenParamMasyvas[m].desnis -1;
     switch (ComboBox6->ItemIndex)
     {case 0:
       {Edit1->Enabled = true;
        Edit2->Enabled = false;
        break;
       };
      case 1:
       {
        Edit1->Enabled = true;
        Edit2->Enabled = true;
```

```
         break;
        };
       case 2:
        {
         Edit1->Enabled = true;
         Edit2->Enabled = true;
         break;
        };
       case 3:
        {Edit1->Enabled = true;
         Edit2->Enabled = false;
         break;
        };
       }
     }
   Edit1->Text = FloatToStrF(GenParamMasyvas[m].par_1, ffFixed, 8, 5);
   Edit2->Text = FloatToStrF(GenParamMasyvas[m].par_2, ffFixed, 8, 5);

   if (GenParamMasyvas[m].prioritetas != -1)
    ComboBox7->Text = GenParamMasyvas[m].prioritetas;

   if (GenParamMasyvas[m].aptarn_agregato_nr !=-1)
    ComboBox8->Text = GenParamMasyvas[m].aptarn_agregato_nr;

}
//------------------------------
void TForm1::Pildyti(int m)
{
   edAptarnavimoLinijuSkaicius->Text =
IntToStr(AgregObjektas.Agregatas[m].Parametrai.atp_liniju_kiekis);

   if ( AgregObjektas.Agregatas[m].Parametrai.atp_liniju_kiekis == 1)
   { ComboBox5->Enabled = false;
   }
   else
   { ComboBox5->Enabled = true;
   }
   edEiliuIlgis->Text  =
IntToStr(AgregObjektas.Agregatas[m].Parametrai.maksimalus_eiles_ilgis);
   if (IntToStr(AgregObjektas.Agregatas[m].Parametrai.maksimalus_eiles_ilgis) == -
1)
    {
      CheckBox1->Checked = true;
      edEiliuIlgis->Enabled = false;
    }
   else
    {
      CheckBox1->Checked = false;
      edEiliuIlgis->Enabled = true;
    }
   edEiliuKiekis->Text  =
IntToStr(AgregObjektas.Agregatas[m].Parametrai.eiliu_kiekis);
   if (IntToStr(AgregObjektas.Agregatas[m].Parametrai.eiliu_kiekis) == 1)
    { ComboBox2->Enabled = false;
      RadioButton1->Enabled = false;
      RadioButton2->Enabled = false;
      ComboBox4->Enabled = false;
    }
   else
   {  ComboBox2->Enabled = true;
      RadioButton1->Enabled = true;
      RadioButton2->Enabled = true;
      ComboBox4->Enabled = true;
```

```
   }
   edSkirstinioParametras1->Text =
FloatToStr(AgregObjektas.Agregatas[m].Parametrai.par_1);
   edSkirstinioParametras2->Text =
FloatToStr(AgregObjektas.Agregatas[m].Parametrai.par_2);
   edIsejimoKanaluKiekis->Text =
IntToStr(AgregObjektas.Agregatas[m].Parametrai.isejimo_kanalu_kiekis);
   edIsejimoStrategija->Text =
IntToStr(AgregObjektas.Agregatas[m].Parametrai.isijimo_kanalo_strategija);

   AnsiString Skirstiniai[5];
   Skirstiniai[0] = "Eksponentinis";
   Skirstiniai[1] = "Normalusis";
   Skirstiniai[2] = "Tolygusis";
   Skirstiniai[3] = "Puasono";

   AnsiString List[10];
   List[0] = "Ilgiausia eilë"; //11
   List[1] = "Cikliðkai aptanaujamos"; //12
   List[2] = "Pirma netuðèia eilë"; //13
   List[3] = "Atsitiktinë eilë"; //14
   List[4] = "Ilgiausia eilë iki pabaigos"; //21
   List[5] = "Cikliðkai aptanaujamos iki pabaigos"; //22
   List[6] = "Pirma netuðèia eilë iki pabaigos"; //23
   List[7] = "Atsitiktinë eilë iki pabaigos"; //24

   temp_listbox = m;
   if (AgregObjektas.Agregatas[m].Parametrai.eiles_strategija<20)
    {
    ComboBox2->Text = List[AgregObjektas.Agregatas[m].Parametrai.eiles_strategija-
11];
    RadioButton2->Checked = true;
    }
   else
    {ComboBox2->Text =
List[AgregObjektas.Agregatas[m].Parametrai.eiles_strategija-21];
     RadioButton1->Checked = true;
    }
   AnsiString ParaiskosEilese[3];
   ParaiskosEilese[0] = "FIFO - pirma atëjo, pirma iðëjo";
   ParaiskosEilese[1] = "LIFO - paskutinë atëjo, pirma iðëjo";
   ParaiskosEilese[2] = "PRMAX - aukðèiausias prioritetas";

   AnsiString EilesParinkimasParaiskai[3];
   EilesParinkimasParaiskai[0] = "Á trumpiausià eilæ";
   EilesParinkimasParaiskai[1] = "Cikliðkai parenkant eilæ";
   EilesParinkimasParaiskai[2] = "Á atsitiktinæ eilæ";

   AnsiString IrenginioParinkimas[5];
   IrenginioParinkimas[0] = "Neuþimta linija po paskutinës pasirinktos";
   IrenginioParinkimas[1] = "Pirmoji neuþimta linija";
   IrenginioParinkimas[2] = "Pagal minimalø vidutiná apkrautumà";
   IrenginioParinkimas[3] = "Pagal ilgiausià prastovos trukmæ";
   IrenginioParinkimas[4] = "Pagal trumpiausià prastovos trukmæ";

   if (AgregObjektas.Agregatas[m].Parametrai.linijos_strategija !=-1)
   ComboBox5->Text =
IrenginioParinkimas[AgregObjektas.Agregatas[m].Parametrai.linijos_strategija - 1];

   if (AgregObjektas.Agregatas[m].Parametrai.eiles_parinkimas_naujai_paraiskai !=-
1)
```

```
   ComboBox4->Text =
EilesParinkimasParaiskai[AgregObjektas.Agregatas[m].Parametrai.eiles_parinkimas_na
ujai_paraiskai-1];

   if (AgregObjektas.Agregatas[m].Parametrai.paraisk_pasir_aptarn_eileje_i !=-1)
   ComboBox3->Text =
ParaiskosEilese[AgregObjektas.Agregatas[m].Parametrai.paraisk_pasir_aptarn_eileje_
i - 1];

   if (AgregObjektas.Agregatas[m].Parametrai.apt_trukm_linijoje_skirstinys !=-1)
   ComboBox1->Text =
Skirstiniai[(AgregObjektas.Agregatas[m].Parametrai.apt_trukm_linijoje_skirstinys)-
1];

   ListBox2->Clear();
   for (int i = 0;
i<AgregObjektas.Agregatas[m].Parametrai.isejimo_kanalu_kiekis;i++)
   ListBox2->Items->Add(AgregObjektas.Agregatas[m].Parametrai.isejimo_kanalai[i]);
}
//---------------------------------------------------------------------------
---
void __fastcall TForm1::edSkirstinioParametras1Exit(TObject *Sender)
{
  int m = ListBox1->ItemIndex;
  try
  {AgregObjektas.Agregatas[m].Parametrai.par_1 =
StrToFloat(edSkirstinioParametras1->Text);
   switch (AgregObjektas.Agregatas[m].Parametrai.apt_trukm_linijoje_skirstinys)
   { case 1:
     {
       edSkirstinioParametras2->Enabled = false;
       if(StrToFloat(edSkirstinioParametras1->Text) <=0)
        ShowMessage("Neteisingas parametras 1");
      break;
     };
      case 2:
     {
       edSkirstinioParametras2->Enabled = true;
       if(StrToFloat(edSkirstinioParametras1->Text) <0)
        ShowMessage("Neteisingas parametras 1");
       if(StrToFloat(edSkirstinioParametras2->Text) <0)
        ShowMessage("Neteisingas parametras 2");
      break;
     };
      case 3:
     {
       edSkirstinioParametras2->Enabled = true;
       if(StrToFloat(edSkirstinioParametras1->Text) <0)
        ShowMessage("Neteisingas parametras 1");
       if(StrToFloat(edSkirstinioParametras2->Text) <0)
        ShowMessage("Neteisingas parametras 2");
       if(StrToFloat(edSkirstinioParametras2-
>Text)<=StrToFloat(edSkirstinioParametras1->Text))
        ShowMessage("Parametras 2 turi bûti didesnis uþ Parametrà 1");
      break;
     };
    }
  }catch (...)
  {ShowMessage("Parametras_1: Áveskite skaièiø!" );}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::edSkirstinioParametras2Exit(TObject *Sender)
{
```

```
  int m = ListBox1->ItemIndex;
  try
  {AgregObjektas.Agregatas[m].Parametrai.par_2 =
StrToFloat(edSkirstinioParametras2->Text);
  }catch (...)
  {ShowMessage("Parametras_2: Áveskite skaièiø!" );}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
  if (RadioButton1->Checked == true)
  {
   int l1 = ListBox1->ItemIndex;
   if (ListBox1->Items->Count==1) l1=0;
   if (l1==-1) l1=temp_listbox;
   int index = ComboBox2->ItemIndex;
   if (index==-1&& l1!=-1)
    if (AgregObjektas.Agregatas[l1].Parametrai.eiles_strategija >20)
     index = AgregObjektas.Agregatas[l1].Parametrai.eiles_strategija - 21;
    else index = AgregObjektas.Agregatas[l1].Parametrai.eiles_strategija - 11;

   if (l1!=-1)
   AgregObjektas.Agregatas[l1].Parametrai.eiles_strategija = index+21;
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
  if (RadioButton2->Checked == true)
  {
   int l1 = ListBox1->ItemIndex;
   if (ListBox1->Items->Count==1) l1=0;
   int index = ComboBox2->ItemIndex;
   if (index==-1 && l1!=-1)
    if (AgregObjektas.Agregatas[l1].Parametrai.eiles_strategija >20)
     index = AgregObjektas.Agregatas[l1].Parametrai.eiles_strategija - 21;
    else index = AgregObjektas.Agregatas[l1].Parametrai.eiles_strategija - 11;
   if (l1!=-1)
   AgregObjektas.Agregatas[l1].Parametrai.eiles_strategija = index+11;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ListBox3Click(TObject *Sender)
{
 PildytiGeneratorius(ListBox3->ItemIndex);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox7Change(TObject *Sender)
{
 int index = ListBox3->ItemIndex;
 if (index==-1) index = 0;
 GenParamMasyvas[index].prioritetas = ComboBox7->Text.ToInt();
 PildytiGeneratorius(index);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox8Change(TObject *Sender)
{
 int index = ListBox3->ItemIndex;
 GenParamMasyvas[index].aptarn_agregato_nr = ComboBox8->Text.ToInt();
 PildytiGeneratorius(index);
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::CB_KiekisEilejeClick(TObject *Sender)
{
 if (CB_KiekisEileje->Checked == 1)
   AgregObjektas.Statistiku_pasirinkimas[0]=1;
 else AgregObjektas.Statistiku_pasirinkimas[0]=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CB_LaikasEilejeClick(TObject *Sender)
{
 if (CB_LaikasEileje->Checked == 1)
   AgregObjektas.Statistiku_pasirinkimas[2]=1;
 else AgregObjektas.Statistiku_pasirinkimas[2]=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CB_LaikasSistemojeClick(TObject *Sender)
{
 if (CB_LaikasSistemoje->Checked == 1)
   AgregObjektas.Statistiku_pasirinkimas[5]=1;
 else AgregObjektas.Statistiku_pasirinkimas[5]=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CB_LaikasAptIreninyjeClick(TObject *Sender)
{
 if (CB_LaikasAptIreninyje->Checked == 1)
   AgregObjektas.Statistiku_pasirinkimas[4]=1;
 else AgregObjektas.Statistiku_pasirinkimas[4]=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CB_IrenginioUzimtumasClick(TObject *Sender)
{
 if (CB_IrenginioUzimtumas->Checked == 1)
   AgregObjektas.Statistiku_pasirinkimas[3]=1;
 else AgregObjektas.Statistiku_pasirinkimas[3]=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ToolButton1Click(TObject *Sender)
{
 if( ReadFromFile(DataFileName().c_str()))
 {Vartotojo_sasaja();
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ToolButton2Click(TObject *Sender)
{
 WriteToFile(ResultFileName().c_str());
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ToolButton4Click(TObject *Sender)
{
 Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox6Change(TObject *Sender)
{  int l3 = ListBox3->ItemIndex;
   if (ComboBox6->ItemIndex==0)
   {if ( StrToFloat(Edit1->Text)>0)
     {GenParamMasyvas[l3].desnis = ComboBox6->ItemIndex+1;
      Edit2->Enabled=false;
     }
    else ShowMessage("Parametras1 turi bûti teigiamas!");
   }
   if (ComboBox6->ItemIndex==1)
   { Edit2->Enabled = true;
```

```
    if ( StrToFloat(Edit1->Text)>0 && StrToFloat(Edit2->Text)>=0)
     if ( StrToFloat(Edit1->Text)> StrToFloat(Edit2->Text)*3)
        GenParamMasyvas[l3].desnis = ComboBox6->ItemIndex+1;
     else ShowMessage("Dispersija(Parametras2) perdidelë!");
    else ShowMessage("Parametrai turi bûti teigiami!");
   }
   if (ComboBox6->ItemIndex==2)
   { Edit2->Enabled = true;
    if ( StrToFloat(Edit1->Text)>=0 && StrToFloat(Edit2->Text)>0)
     {if ( StrToFloat(Edit1->Text)> StrToFloat(Edit2->Text))
       ShowMessage("Parametras1 turi bûti maþesnis uþ Parametras2!");
     }
    else ShowMessage("Parametrai turi bûti teigiami!");
   }
  if (ComboBox6->ItemIndex==3)
   {if ( StrToFloat(Edit1->Text)>=0)
     {GenParamMasyvas[l3].desnis = ComboBox6->ItemIndex+1;
      Edit2->Enabled=false;
     }
    else ShowMessage("Parametras1 turi bûti teigiamas!");
   }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
  int m = ListBox3->ItemIndex;
  try
  {
  if (ComboBox6->ItemIndex==0)
   { Edit1->Enabled = true;
    if ( StrToFloat(Edit1->Text)>0)
        { GenParamMasyvas[m].par_1 = StrToFloat(Edit1->Text);
          GenParamMasyvas[ListBox3->ItemIndex].desnis = ComboBox6->ItemIndex+1;
        }
    else ShowMessage("Parametras turi bûti teigiamas!");
   }
   if (ComboBox6->ItemIndex==1)
   { Edit1->Enabled = true;
    if ( StrToFloat(Edit1->Text)>0 && StrToFloat(Edit2->Text)>=0)
     if ( StrToFloat(Edit1->Text)> StrToFloat(Edit2->Text)*3)
        { GenParamMasyvas[m].par_1 = StrToFloat(Edit1->Text);
          GenParamMasyvas[ListBox3->ItemIndex].desnis = ComboBox6->ItemIndex+1;
        }
     else ShowMessage("Dispersija(Parametras2) perdidelë!");
    else ShowMessage("Parametrai turi bûti teigiami!");
   }
  if (ComboBox6->ItemIndex==2)
   { Edit1->Enabled = true;
     Edit2->Enabled = true;
    if ( StrToFloat(Edit1->Text)>=0 && StrToFloat(Edit2->Text)>0 &&
StrToFloat(Edit1->Text)<StrToFloat(Edit2->Text))
        { GenParamMasyvas[m].par_1 = StrToFloat(Edit1->Text);
          GenParamMasyvas[ListBox3->ItemIndex].desnis = ComboBox6->ItemIndex+1;
        }
    else ShowMessage("Parametrai turi bûti teigiami ir par1<par2!");
   }
  }catch (...)
  {ShowMessage("Parametras_1_(Generatorius): Áveskite skaièiø!" );}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Edit2Exit(TObject *Sender)
{
  int m = ListBox3->ItemIndex;
```

```
  try
  {
  if (ComboBox6->ItemIndex==1)
   {Edit2->Enabled = true;
    if ( StrToFloat(Edit1->Text)>0 && StrToFloat(Edit2->Text)>=0)
     if ( StrToFloat(Edit1->Text)> StrToFloat(Edit2->Text)*3)
       { GenParamMasyvas[m].par_2 = StrToFloat(Edit2->Text);
         GenParamMasyvas[m].desnis = ComboBox6->ItemIndex+1;
       }
     else ShowMessage("Dispersija(Parametras2) perdidelë!");
    else ShowMessage("Parametrai turi bûti teigiami!");
   }

  if (ComboBox6->ItemIndex==2)
   { Edit2->Enabled = true;
    if ( StrToFloat(Edit1->Text)>0 && StrToFloat(Edit2->Text)>=0)
     if ( StrToFloat(Edit1->Text)< StrToFloat(Edit2->Text))
        //GenParamMasyvas[l3].desnis = ComboBox6->ItemIndex+1;
        GenParamMasyvas[m].par_2 = StrToFloat(Edit2->Text);
     else ShowMessage("Parametras1 turi bûti maþesnis uþ Parametras2!");
    else ShowMessage("Parametrai turi bûti teigiami!");
   }
  }catch (...)
  {ShowMessage("Parametras_2_(Generatorius): Áveskite skaièiø!" );}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::edIsejimoKanaluKiekisExit(TObject *Sender)
{
  int m = ListBox1->ItemIndex;
//  if (ListBox1->Count == 1) m = 0;
  int buves_kiekis =AgregObjektas.Agregatas[m].Parametrai.isejimo_kanalu_kiekis;

  try
  {AgregObjektas.Agregatas[m].Parametrai.isejimo_kanalu_kiekis =
edIsejimoKanaluKiekis->Text.ToInt();
   ListBox2->Clear();

    for (int i = 0; i<AgregObjektas.Agregatas[m].Parametrai.isejimo_kanalu_kiekis;
i++)
     {  if
(buves_kiekis<AgregObjektas.Agregatas[m].Parametrai.isejimo_kanalu_kiekis &&
i>=buves_kiekis)
          AgregObjektas.Agregatas[m].Parametrai.isejimo_kanalai[i] = -1;

        ListBox2->Items-
>Add(AgregObjektas.Agregatas[m].Parametrai.isejimo_kanalai[i]);
     }
  }catch (...)
  {ShowMessage("Áveskite sveikà skaièiø!" );}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::edLaikasExit(TObject *Sender)
{
  try
  {
   modeliavimo_laikas = edLaikas->Text.ToDouble();
  }
  catch (...)
  {ShowMessage("Áveskite skaièiø!");}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CB_EX_SAClick(TObject *Sender)
{
```

```
  if (CB_EX_SA->Checked == 1)
   AgregObjektas.Statistiku_pasirinkimas[1]=1;
 else AgregObjektas.Statistiku_pasirinkimas[1]=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Saugoti1Click(TObject *Sender)
{
  WriteToFile(ResultFileName().c_str());
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Iseiti1Click(TObject *Sender)
{
 Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
 if (CB_EX_SA->Checked == false && CB_IrenginioUzimtumas->Checked==false
&&CB_KiekisEileje->Checked==false &&
     CB_LaikasAptIreninyje->Checked==false && CB_LaikasEileje->Checked==false &&
CB_LaikasSistemoje->Checked== false)
  { ShowMessage("Nepaþymëtas stebimas dydis!");}
 else
 {
 try
  {
   modeliavimo_laikas = edLaikas->Text.ToDouble();
   AgregObjektas.nuemimo_laikas = Edit3->Text.ToDouble();
   if (AgregObjektas.AgregatuSkaicius>0 && n>0)
   {
     for (int i=0; i<AgregObjektas.AgregatuSkaicius; i++)
     {  AgregObjektas.PradinesBusenos(&AgregObjektas.Agregatas[i]);
     }
     WriteToFile("last_model.tsk");
    Modeliavimas();
   }
   else ShowMessage("Blogi duomenys!");
  }
  catch (...)
  {ShowMessage("Modeliavimo laikas turi bûti skaièius!");}
 }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
 int l1 = ListBox1->ItemIndex;
 if (ListBox1->Items->Count==1) l1=0;
 if (l1!=-1 && ListBox1->Items->Count>1)
 { for (int i = l1; i<AgregObjektas.AgregatuSkaicius-1; i++)
     AgregObjektas.Agregatas[i]=AgregObjektas.Agregatas[i+1];
  AgregObjektas.AgregatuSkaicius--;
 }
 ListBox1->Clear();
 Vartotojo_sasaja();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn3Click(TObject *Sender)
{
 AgregObjektas.AgregatuSkaicius++;
 ListBox1->Items->Add(IntToStr(AgregObjektas.AgregatuSkaicius)-1);
AgregObjektas.PradineParametruIrBusenuAibesBusena(&AgregObjektas.Agregatas[AgregOb
jektas.AgregatuSkaicius-1]);
 Vartotojo_sasaja();
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn4Click(TObject *Sender)
{
 int index = ListBox3->ItemIndex;
 if (index==-1) index = 0;
 if (n==0)
 { ShowMessage("Generatoriu jau nëra!");
 }
 else
 {
  for (int i = index; i < n-1; i++)
   {
    GenParamMasyvas[i] = GenParamMasyvas[i+1];
   }
  n--;
 }
 Vartotojo_sasaja();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn5Click(TObject *Sender)
{
 if (n==0)
 {
  n=1;
  GenParamMasyvas[0]= GenDefault;
 }
 else
 {
  n++;
  GenParamMasyvas[n-1]= GenDefault;
 }
 Vartotojo_sasaja();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn6Click(TObject *Sender)
{
  AgregObjektas.Statistiku_pasirinkimas[0]=1;
  AgregObjektas.Statistiku_pasirinkimas[1]=1;
  AgregObjektas.Statistiku_pasirinkimas[2]=1;
  AgregObjektas.Statistiku_pasirinkimas[3]=1;
  AgregObjektas.Statistiku_pasirinkimas[4]=1;
  AgregObjektas.Statistiku_pasirinkimas[5]=1;
  AgregObjektas.Statistiku_pasirinkimas[6]=1;
 CB_EX_SA->Checked = true;
 CB_IrenginioUzimtumas->Checked = true;
 CB_KiekisEileje->Checked = true;
 CB_LaikasAptIreninyje->Checked = true;
 CB_LaikasEileje->Checked = true;
 CB_LaikasSistemoje->Checked = true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn7Click(TObject *Sender)
{
  AgregObjektas.Statistiku_pasirinkimas[0]=0;
  AgregObjektas.Statistiku_pasirinkimas[1]=0;
  AgregObjektas.Statistiku_pasirinkimas[2]=0;
  AgregObjektas.Statistiku_pasirinkimas[3]=0;
  AgregObjektas.Statistiku_pasirinkimas[4]=0;
  AgregObjektas.Statistiku_pasirinkimas[5]=0;
  AgregObjektas.Statistiku_pasirinkimas[6]=0;
 CB_EX_SA->Checked = false;
 CB_IrenginioUzimtumas->Checked = false;
```

```
 CB_KiekisEileje->Checked = false;
 CB_LaikasAptIreninyje->Checked = false;
 CB_LaikasEileje->Checked = false;
 CB_LaikasSistemoje->Checked = false;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox1Exit(TObject *Sender)
{
  int m = ListBox1->ItemIndex;
  if (CheckBox1->Checked == true)
    { AgregObjektas.Agregatas[m].Parametrai.maksimalus_eiles_ilgis = -1;
      edEiliuIlgis->Enabled = false;
    }
  if (CheckBox1->Checked == false)
  {AgregObjektas.Agregatas[m].Parametrai.maksimalus_eiles_ilgis = edEiliuIlgis-
>Text.ToInt();
   edEiliuIlgis->Enabled = true;
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  int m = ListBox1->ItemIndex;
  if (m != -1)
  {
  if (CheckBox1->Checked == true)
    { AgregObjektas.Agregatas[m].Parametrai.maksimalus_eiles_ilgis = -1;
      edEiliuIlgis->Enabled = false;
    }

  if (CheckBox1->Checked == false)
  {AgregObjektas.Agregatas[m].Parametrai.maksimalus_eiles_ilgis = edEiliuIlgis-
>Text.ToInt();
   edEiliuIlgis->Enabled = true;
  }
 }
}
//---------------------------------------------------------------------------
```

„Pagrindinis.h“

```
//---------------------------------------------------------------------------
#ifndef PagrininisH
#define PagrininisH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Forms.hpp>
#include "Apie.h"
#include "Agregatai.h"
#include "CGAUGES.h"
#include <Menus.hpp>
#include <ComCtrls.hpp>
#include <Dialogs.hpp>
#include <ExtCtrls.hpp>
#include <ToolWin.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Buttons.hpp>
int const kiekis = 100;
 struct GeneratoriuParametrai
 { int desnis,
       prioritetas,
       aptarn_agregato_nr;
```

```
   float   par_1,
           par_2,
           laikas_tarpinis;
 };

//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
        TCGauge *CGauge1;
        TSaveDialog *SaveDialog1;
        TOpenDialog *OpenDialog1;
        TEdit *edLaikas;
        TMainMenu *MainMenu1;
        TMenuItem *File;
        TMenuItem *Atidaryti;
        TMenuItem *Saugoti1;
        TMenuItem *Iseiti1;
        TListBox *ListBox1;
        TEdit *edEiliuKiekis;
        TLabel *laEiliuKiekis;
        TEdit *edEiliuIlgis;
        TLabel *laEiliuIlgis;
        TLabel *laEilesParNaujaiParaiskai;
        TLabel *laEilesParinkimasAptarnavimui;
        TLabel *laEilesAptarnavimoStrategija;
        TEdit *edAptarnavimoLinijuSkaicius;
        TLabel *laAptarnavimoLinijuSkaicius;
        TLabel *laAptLinijuStrategija;
        TLabel *laAptSkirstinys;
        TEdit *edSkirstinioParametras1;
        TLabel *laSkirstinioParametras1;
        TLabel *laSkirstinioParametras2;
        TEdit *edSkirstinioParametras2;
        TEdit *edIsejimoStrategija;
        TLabel *laIsejimoStrategija;
        TEdit *edIsejimoKanaluKiekis;
        TLabel *laIsejimoKanaluSkaicius;
        TListBox *ListBox2;
        TButton *Keisti;
        TEdit *edNuorodaKeisti;
        TComboBox *ComboBox1;
        TComboBox *ComboBox2;
        TComboBox *ComboBox3;
        TComboBox *ComboBox4;
        TComboBox *ComboBox5;
        TRadioButton *RadioButton1;
        TRadioButton *RadioButton2;
        TListBox *ListBox3;
        TComboBox *ComboBox6;
        TEdit *Edit1;
        TEdit *Edit2;
        TComboBox *ComboBox7;
        TComboBox *ComboBox8;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TBevel *Bevel1;
        TBevel *Bevel2;
        TStaticText *StaticText1;
```

```
TStaticText *StaticText2;
TLabel *Label6;
TCheckBox *CB_KiekisEileje;
TCheckBox *CB_LaikasEileje;
TCheckBox *CB_LaikasSistemoje;
TCheckBox *CB_LaikasAptIreninyje;
TCheckBox *CB_IrenginioUzimtumas;
TStaticText *StaticText3;
TBevel *Bevel3;
TLabel *Label7;
TCheckBox *CB_EX_SA;
TStatusBar *StatusBar1;
TBitBtn *BitBtn1;
TBitBtn *BitBtn2;
TBitBtn *BitBtn3;
TBitBtn *BitBtn4;
TBitBtn *BitBtn5;
TBitBtn *BitBtn6;
TBitBtn *BitBtn7;
TLabel *Label8;
TCheckBox *CheckBox1;
TEdit *Edit3;
TLabel *Label9;
TLabel *Label10;
void __fastcall SaugotiClick(TObject *Sender);

void __fastcall ListBox1Click(TObject *Sender);
void __fastcall AtidarytiClick(TObject *Sender);
void __fastcall edEiliuKiekisExit(TObject *Sender);
void __fastcall edEiliuIlgisExit(TObject *Sender);
void __fastcall edAptarnavimoLinijuSkaiciusExit(TObject *Sender);
void __fastcall ListBox2Click(TObject *Sender);
void __fastcall KeistiClick(TObject *Sender);
void __fastcall ComboBox1Change(TObject *Sender);
void __fastcall ComboBox2Change(TObject *Sender);
void __fastcall ComboBox3Change(TObject *Sender);
void __fastcall ComboBox4Change(TObject *Sender);
void __fastcall ComboBox5Change(TObject *Sender);
void __fastcall edSkirstinioParametras1Exit(TObject *Sender);
void __fastcall edSkirstinioParametras2Exit(TObject *Sender);
void __fastcall RadioButton1Click(TObject *Sender);
void __fastcall RadioButton2Click(TObject *Sender);
void __fastcall ListBox3Click(TObject *Sender);
void __fastcall ComboBox7Change(TObject *Sender);
void __fastcall ComboBox8Change(TObject *Sender);
void __fastcall CB_KiekisEilejeClick(TObject *Sender);
void __fastcall CB_LaikasEilejeClick(TObject *Sender);
void __fastcall CB_LaikasSistemojeClick(TObject *Sender);
void __fastcall CB_LaikasAptIreninyjeClick(TObject *Sender);
void __fastcall CB_IrenginioUzimtumasClick(TObject *Sender);
void __fastcall ToolButton1Click(TObject *Sender);
void __fastcall ToolButton2Click(TObject *Sender);
void __fastcall ToolButton4Click(TObject *Sender);
void __fastcall ComboBox6Change(TObject *Sender);
void __fastcall Edit1Exit(TObject *Sender);
void __fastcall Edit2Exit(TObject *Sender);
void __fastcall edIsejimoKanaluKiekisExit(TObject *Sender);
void __fastcall edLaikasExit(TObject *Sender);
void __fastcall CB_EX_SAClick(TObject *Sender);
void __fastcall Saugoti1Click(TObject *Sender);
void __fastcall Iseiti1Click(TObject *Sender);
void __fastcall BitBtn1Click(TObject *Sender);
void __fastcall BitBtn2Click(TObject *Sender);
```

```
        void __fastcall BitBtn3Click(TObject *Sender);
        void __fastcall BitBtn4Click(TObject *Sender);
        void __fastcall BitBtn5Click(TObject *Sender);
        void __fastcall BitBtn6Click(TObject *Sender);
        void __fastcall BitBtn7Click(TObject *Sender);
        void __fastcall CheckBox1Exit(TObject *Sender);
        void __fastcall CheckBox1Click(TObject *Sender);

private:    // User declarations
public:           // User declarations

 int n;
 int temp_listbox;
 float modeliavimo_laikas;
 long Skaitliukas;
 double paraisku_laikas[kiekis];
 GeneratoriuParametrai GenDefault;
 GeneratoriuParametrai GenParamMasyvas[Cmax];
 void Paraisku_generatorius(SParaiska *Paraiska, long laikas,int prioritetas,int
gener_kopija,int desnis, AnsiString vardas);
 SParaiska *Paraiskos;
 void PradineBusena(SParaiska *Paraiska);
 void Modeliavimas();
 SParaiska Paraiskos_kurimas(double laikas, int gen_numeris);
 AnsiString ResultFileName();
 AnsiString DataFileName();
 void WriteToFile(const char *fn);
 int ReadFromFile(const char *fn);
 void Vartotojo_sasaja();
 void PildytiGeneratorius(int m);
 void Pildyti(int m);
 TAgregatai AgregObjektas;
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

„Apie.cpp"

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Apie.h"
#include "Pagrininis.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
  Form2->Focused();
}

void __fastcall TForm2::Button1Click(TObject *Sender)
{
 Form1->Visible = true;
 Form1->Enabled = true;
 Form2->Close();
```

```
}
//---------------------------------------------------------------------------
```

8 priedas. „Apie.h“

```
//---------------------------------------------------------------------------
#ifndef ApieH
#define ApieH
//---------------------------------------------------------------------------
#include "Pagrininis.h"
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <jpeg.hpp>
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:      // IDE-managed Components
        TButton *Button1;
        TLabel *Label1;
        TImage *Image1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TImage *Image2;
        TLabel *Label5;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```

6 PREIDAS. MODELIAVIMO SISTEMOS IŠKVIETIMO PROGRAMINIS KODAS

```
Dim state
Sub PushBtnEvent_Start()
  Rem TODO: Add your code here

End Sub

Sub PushBtnEvent_Exec(Inputs,Outputs)
state = 0
PathName = Inputs(0).Value
FileName = Inputs(1).Value
reclength = Inputs(2).Value
programa = "Programa.exe"
cols = Inputs(3).Value
rows = 0

Dim StringResults()
   Set objShell = CreateObject("WScript.Shell")
   objShell.Run programa,1,True
'Required Objects
Set oFileSystem = CreateObject("Scripting.FileSystemObject")
```

```
   If oFileSystem.FileExists(PathName & FileName) = False Then
     MsgBox("Failas nerastas")

   Else
     Set Result = oFileSystem.OpenTextFile(PathName & FileName, 1)

   End If

   Do While Not Result.AtEndOfStream
    Contents = Result.ReadLine
    rows = rows+1
   Loop
    Result.Close


 If (rows>0) Then

    Set Result = oFileSystem.OpenTextFile(PathName & FileName, 1)
   Contents = Result.ReadLine

    i=0
   ReDim Result1(rows-1,cols-1)

    For i = 0 to rows-1
      j = 0
        For j = 0 to cols-1
           position = j*reclength +1
           If (Len(Contents) > position) Then
            entry = Mid(Contents, position, reclength)
              If IsNumeric(Trim(entry)) Then
                Result1(i,j) = CDbl(entry)
              Else
                Result1(i,j) = Trim(entry)
              End If
           Else
              Result(i,j) = 0
           End If
        Next
       If (Result.AtEndOfStream) Then
         Contents = "END"
       Else
         Contents = Result.ReadLine
       End If
      Next

    Result.Close
   Outputs(0).Value = Result1
    Outputs(1).Value = i

Else
  MsgBox("Modeliavimo rezultatų duomenų matrica tuščia!")
End If

End Sub

Sub ScriptObjEvent_Stop()
  Rem TODO: Add your code here
End Sub


Sub PushBtn_Click()
   If state = 0 Then
     state = 1
```

```
  Else
    state = 0
  End If
  PushBtn.Recalculate()
End Sub
```

7 PRIEDAS. STATISTINĖS ANALIZĖS PAKETO SAS IŠKVIETIMO PROGRAMINIS KODAS

```
Dim state
Sub PushBtnEvent_Start()
  Rem TODO: Add your code here
End Sub
Sub PushBtnEvent_Exec(Inputs,Outputs)
state = 0
SASPath = Inputs(0).Value
FileName = Inputs(1).Value
rez = Inputs(2).Value
HTMLViewer = Inputs(3).Value

Dim curDate,curDateTime
Dim newDate,newDateTime
curDate = Date
newDate = DateAdd("yyyy",-5,curDate)
Dim oShell
Set oShell = CreateObject("Wscript.shell")
oShell.run "cmd /K date " & CStr(newDate)& " & exit"

Dim StringResults()

   SYSIN ="-sysin "
   NOLOGO = "-nologo -nosplash -noicon "
   Set objShell = CreateObject("WScript.Shell")
   s = Chr(34) & SASPath & Chr(34) &  SYSIN & Chr(34)& FileName  & Chr(34) &
NOLOGO & Chr(34)
   objShell.Run s, 0, True

   Set ws = CreateObject("WScript.Shell")
   IE_PATH = Chr(34) & HTMLViewer & Chr(34)
   Dim sCurPath
   sCurPath = CreateObject("Scripting.FileSystemObject").GetAbsolutePathName(".")
   file_to_open = sCurPath & rez & Chr(34)
   ws.Run IE_PATH & Chr(34) &file_to_open, 1, False
oShell.run "cmd /K date " & CStr(CurDate)& " & exit"
End Sub

Sub PushBtnEvent_Stop()
  Rem TODO: Add your code here
End Sub

Sub PushBtn_Click()
  If state = 0 Then
    state = 1
  Else
    state = 0
  End If
  PushBtn.Recalculate()
End Sub
```