



VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

Mokomosios praktikos ataskaita

Zonos padengimo kelio planavimo simuliacinės įrangos internetinės sėsajos projektavimas, kūrimas ir diegimas

Justas Remeikis

Praktikos institucija : Vilniaus Universitetas Duomenų mokslo ir skaitmeninių technologijų institutas

Praktikos vadovas : Prof. dr. Virginijus Marcinkevičius

Vilnius
2026

Turinys

Dokumente naudojamos sąvokos ir santrumpos	4
1. Įvadas	6
1.1. Tikslas	6
1.2. Uždaviniai	6
2. Analitinė dalis	7
2.1. ZPKP algoritmai	8
2.1.1. Klasikiniai algoritmai	8
2.1.1.1 Trapecinė dekompozicija	9
2.1.1.2 Bustrofedono dekompozicija	11
2.1.1.3 Iškiloji dekompozicija	12
2.1.2. Euristikomis grįsti algoritmai	14
2.1.2.1 Iškiloji dekompozicija (modifikacija)	14
2.1.2.2 Prisitaikantis zonas padengimo kelio planavimas	14
2.1.2.3 Re — DQN algoritmas	16
2.1.2.4 C* algoritmas	17
2.2. ZPKP algoritmų apibendrinimas	19
2.3. Sukurti sprendimai ZPKP tyrimų srityje	21
3. Metodinė dalis	23
3.1. Sistemos paskirtis	23
3.2. Keliami reikalavimai sistemai	23
3.3. Sistemos naudotojai	24
3.4. Panaudojimo atvejų notacija	24
3.5. Veiklos diagramos notacija	26
3.6. Detalizuoti funkciniai reikalavimai	27
3.6.1. Funkcinis reikalavimas „Tvarkyti ZPKP“	28
3.6.1.1 Funkcinis reikalavimas „Konfigūruoti ZPKP algoritmą“	29
3.6.1..1.0.1 Funkcinis reikalavimas „Pasirinkti ZPKP algoritmą“	30
3.6.1..1.0.2 Funkcinis reikalavimas „Keisti ZPKP algoritmo parametrus“	31
3.6.1.2 Funkcinis reikalavimas „Konfigūruoti ZPKP testavimą“	32
3.6.1..2.0.1 Funkcinis reikalavimas „Kurti ZPKP testavimo scenarijų“	33
3.6.1..2.0.2 Funkcinis reikalavimas „Redaguoti ZPKP testavimo scenarijų“	37
3.6.1..2.0.3 Funkcinis reikalavimas „Trinti ZPKP testavimo scenarijų“	39

3.6.2.	ZPKP funkcinių reikalavimų grupė	41
3.6.2..1	Funkcinis reikalavimas „Inicijuoti ZPKP“	42
3.6.2..2	Funkcinis reikalavimas „Vykdyti ZPKP“	43
3.6.2..3	Funkcinis reikalavimas „Vykdyti ZPKP testavimą“	45
3.6.2..4	Funkcinis reikalavimas „Peržiūrėti ZPKP rezultatus“	47
3.6.2..5	Funkcinis reikalavimas „Atvaizduoti ZPKP rezultatus“	47
3.6.3.	Aplinkos valdymo funkciniai reikalavimai	49
3.6.3..1	Funkcinis reikalavimas „Įrašyti aplinkos duomenis“	49
3.6.3..2	Funkcinis reikalavimas „Tvarkyti aplinką“	50
3.6.3..2..1	Funkcinis reikalavimas „Konfigūruoti aplinką“	51
3.6.3..2..1..1	Funkcinis reikalavimas „Išsaugoti darbinę aplinką“	51
3.6.3..2..1..2	Funkcinis reikalavimas „Atidaryti išsaugotą aplinką“	52
3.6.3..2..1..3	Funkcinis reikalavimas „Sukurti naują aplinką“	54
3.6.3..2..1..4	Funkcinis reikalavimas „Importuoti aplinką“	56
3.6.3..2..1..5	Funkcinis reikalavimas „Redaguoti aplinkos pavadinimą“	57
3.6.3..2..1..6	Funkcinis reikalavimas „Redaguoti leidimą keisti zonos ir kliūčių tipą“	59
3.6.3..2..1..7	Funkcinis reikalavimas „Nustatyti aplinkos tipą“	60
3.6.3..2..1..8	Funkcinis reikalavimas „Nustatyti aplinkos formatą“	62
3.6.3..2..1..9	Funkcinis reikalavimas „Konvertuoti aplinką“	65
3.6.3..2..2	Funkcinis reikalavimas „Konfigūruoti objektą“	67
3.6.3..2..2..1	Funkcinis reikalavimas „Pasirinkti objektą“	67
3.6.3..2..2..2	Funkcinis reikalavimas „Redaguoti objekto ribas“	70
3.6.3..2..2..3	Funkcinis reikalavimas „Nustatyti objekto tipą“	72
3.6.3..2..2..4	Funkcinis reikalavimas „Sukurti naują objektą“	74
3.6.3..2..2..5	Funkcinis reikalavimas „Ištrinti objektą“	76
3.6.4.	Funkcinis reikalavimas „Eksportuoti duomenis“	77
3.6.5.	Funkcinis reikalavimas „Valdyti darbalaukio peržiūrą“	79
3.7.	Vartotojo sąsajos schema	81
3.8.	Loginė duomenų bazės struktūros schema	83
4.	Igyvendinimo dalis	86
4.1.	Realizacija	86
4.2.	Diegimas	86
4.3.	Testavimas	86
5.	Rezultatai ir apibendrinimas	88

Dokumente naudojamos sąvokos ir santrumpos

Terminas ar santrumpa	Apašymas
CPP	Zonos padengimo kelio planavimas (angl. Coverage Path Planning).
ZPKP	Zonos padengimo kelio planavimas.
AoI	Padengimo zona / sritis (angl. Area of Interest).
TWPS	Dvikryptė artumo paieška (angl. Two-Way Proximity Search).
Re — DQN	Sustiprintas gilusis Q-tinklas (angl. Reinforced Deep Q-Network).
SPG	Sparčiai padengiantis grafas (angl. Rapidly Covering Graph).
ROS	Robotų operacinė sistema (angl. Robot Operating System).
UML	Unifikuota modeliavimo kalba (angl. Unified Modeling Language).
API	Programų sąsaja (angl. Application Programming Interface).
JSON	Duomenų formatas (angl. JavaScript Object Notation).
CSV	Kableliais atskirtų reikšmių formatas (angl. Comma-Separated Values).
Ląstelė	Nesikertanti posritis, į kurią skaidoma padengimo zona.
Ląstelinė dekompozicija	Zonos skaidymas į nesikertančias posritis (ląsteles), kuriose padengimo maršrutą galima lengvai nustatyti.
Gretimumo grafas	Grafas, kurio viršūnės žymi ląsteles, o briaunos — gretimumo ryšius tarp jų (angl. Adjacency graph).
Pjūvis	Vertikali linija, braukama per aplinką dekompozicijos metu (angl. Slice).
Kritinis taškas	Taškas, kuriame pasikeičia pjūvio jungiamumo būsena (angl. Critical point).
Jvykis	Momento, kai pjūvis kerta daugiakampio viršūnę, žymėjimas (angl. Event).
Bustrofedono kelias	Padengimo maršrutas, atliekamas paprastais judėjimais pirmyn-atgal (angl. Boustrophedon path).
Trapecinė dekompozicija	Dekompozicijos metodas, kuriame zona skaidoma į trapecines ląsteles naudojant vertikalią tiesę (angl. Trapezoidal decomposition).
Bustrofedono dekompozicija	Trapecinės dekompozicijos patobulinimas, sujungiantis ląsteles tarp vidinio ir išorinio jvykių (angl. Boustrophedon decomposition).
Iškiloji dekompozicija	Dekompozicijos metodas, skaidantis zoną pagal vidinių kampų geometriją (angl. Convex decomposition).

Neprisitaikantis algoritmas	Algoritmas, besiremiantis tik stacionariai, iš anksto žinoma informacija (angl. Off-line).
Prisitaikantis algoritmas	Algoritmas, naudojantis realaus laiko jutiklių matavimus (angl. On-line).
Orientacinis taškas	Taškas, pažymimas grįžimo maršruto planavimui (angl. Backtracking point).
Daugiakampio plotis	Trumpiausias atstumas tarp dviejų lygiagrečių tiesių, kurios pilnai aprėpia daugiakampį (angl. Width).
Godžioji strategija	Optimizavimo strategija, kiekvienu žingsniu pasirenkanti lokaliai geriausią sprendimą (angl. Greedy strategy).

1. Įvadas

Šioje ataskaitoje aprašomi mokomosios praktikos, atliktos Vilniaus universiteto Duomenų mokslo ir skaitmeninių technologijų institute, iškelti tikslai, atlikti uždaviniai ir gauti rezultatai.

1.1. Tikslas

Mokomosios praktikos tikslas — suprojektuoti, sukurti ir įdiegti internetinę sąsają, skirtą vizualiai simuliuoti zonas padengimo kelio planavimo algoritmų veikimą. Sukurtas įrankis turėtų palengvinti tiriamajį darbą šioje srityje, leidžiant tyrejams koncentruotis į algoritmų kūrimą, o ne į tai, kaip tuos algoritmus analizuoti ir palyginti tarpusavyje.

1.2. Uždaviniai

Praktikos uždaviniai:

1. Egzistuojančių zonas padengimo kelio planavimo (ZPKP) algoritmų analizė.
2. Zonas padengimo kelio planavimo internetinės sąsajos funkcių ir nefunkcinių reikalavimų identifikavimas ir aprašymas.
3. Internetinės sąsajos:
 - Dizaino kūrimas.
 - Projektavimas.
 - Programavimas.
 - Testavimas.
4. Dokumentacijos parengimas.

2. Analitinė dalis

Mokomosios praktikos metu siekiama sukurti simuliacinę aplinką, skirtą tirti bei testuoti zonos padengimo kelio planavimo (angl. Coverage Path Planning, CPP) algoritmus. Zonos padengimo kelio planavimo (toliau — ZPKP) algoritmų tikslas — nustatyti maršrutą, kurį robotas sekdamas padengtų visą zoną (angl. Area of Interest), vengdamas kliūčių. Ši užduotis yra esminė daugelyje robotikos taikymų [18], tokį kaip patalpų valymas, žolės pjovimas, konstrukcijų inspektavimas, žemės ūkis bei stebėjimas, išskaitant tyrinėjimą, žemėlapių braižymą, paiešką ir gelbėjimą.

Viename iš ankstyvųjų ZPKP darbų [1] apibrėžiami reikalavimai, kuriuos robotas turi įvykdysti, kad padengimo užduotis būtų atlikta. Šie reikalavimai aprašyti dvimatei aplinkai, tačiau tie patys kriterijai taikomi ir kitiems padengimo scenarijams:

1. Robotas turi aplankytи visus zonos taškus, visiškai ją padengdamas.
2. Robotas turi padengti zoną be persidengiančių maršrutų.
3. Reikalinga nepertraukiama ir nuosekli operacija be jokio maršruto pasikartojimo.
4. Robotas turi išvengti visų kliūčių.
5. Turi būti naudojamos paprastos judėjimo trajektorijos (pvz., tiesios linijos ar apskritimai), kad būtų paprastesnis roboto valdymas.
6. Esamomis sąlygomis pageidaujamas „optimalus“ maršrutas.

Tačiau ne visada įmanoma tenkinti visas šias sąlygas sudėtingose aplinkose, todėl kartais tenka taikyti prioritetinj vertinimą, nustatant, kurie kriterijai yra svarbiausi konkretiam taikymui.

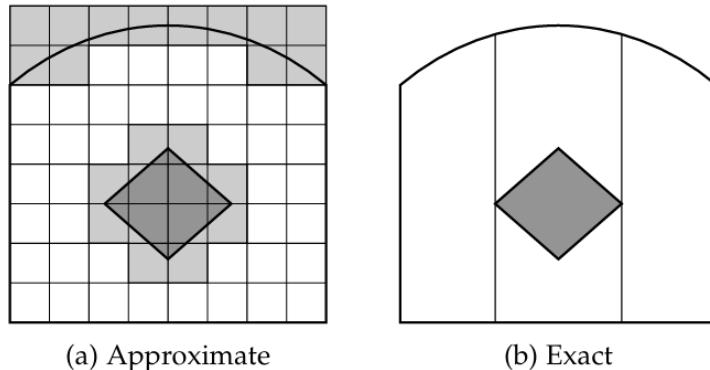
Pagal padengimo užduoties reikalavimus išskiriami kiekybiniai rodikliai, pagal kuriuos vertinami ZPKP algoritmai:

1. Zonos padengimas — kokia zonos dalis padengiama.
2. Maršruto persidengimas — kokia maršruto dalis yra išnaudota neefektyviai.
3. Posūkių skaičius.
4. Maršruto vykdymo laikas.
5. Energijos sunaudojimas.

Pagal [7, 10], norint sugeneruoti optimalų maršrutą, ypač be piločiams orlaiviams, efektyviausia yra minimizuoti posūkių (brangios judėjimo operacijos, dėl kurių robotui reikia stabdyti ir akseleruoti) skaičių. Tai leidžia sumažinti bendrą maršruto ilgį, o taip ir vykdymo laiką bei energijos sąnaudas, nesumažinant zonos padengimo.

ZPKP algoritmai skirtomi į klasikinius (garantuojančius visišką padengimą, 1 a dalis) ir euristinius (ieškančius pakankamai gero sprendimo greičiau, 1 b dalis). Pagal Choset [3] pasiūlytą klasifikaciją, algoritmai gali būti skirtomi į neprisitaikančius (angl. Off-line) arba prisitaikančius (angl. On-line).

Neprisitaikantys algoritmai remiasi tik stacionaria informacija, o aplinka laikoma iš anksto žinoma. Tačiau priešlaida, kad aplinka visiškai žinoma iš anksto, daugelyje scenarijų gali būti nerealistiška. Tuo tarpu prisitaikantys algoritmai nesiremia išankstinėmis žiniomis apie aprépiamą aplinką ir naudoja realaus laiko jutiklių matavimus, kad padengtų zoną. Todėl šie algoritmai dar vadinami jutikliais pagrįstais padengimo algoritmais (angl. Sensor-based coverage algorithms).



1 pav. Dvi pagrindinės ląstelinės dekompozicijos klasės: a) — aproksimuota, b) — tiksliai [7].

Ankstyvuose zonas padengimo tyrimuose [5] maršrutai buvo užprogramuojami iš anksto, ne-naudojant jokio automatinio kelio generavimo algoritmo – kelią reikėdavo apibrėžti rankiniu būdu. Procesas buvo lengvinamas taikant standartinius maršruto šablonus, bet jie neveikė aplinkose su kliūtimis.

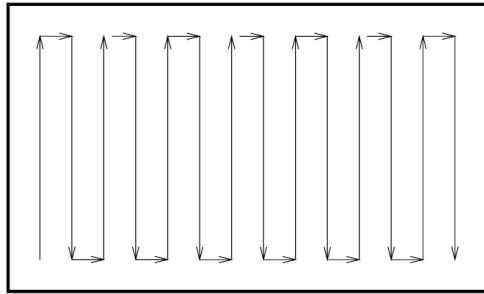
Kitas paprastas būdas — atsitiktinis maršrutas [15]. Tam tikrais atvejais tai gali būti tinkamas problemos sprendimo būdas. Jei grindys ilgą laiką valomos atsitiktine tvarka, jos pakankamai ilgai valant turėtų būti švarios. Privalumas tas, kad tokios sistemos įgyvendinimas nėra sudėtingas — ne-reikia sudėtingų lokalizacijos jutiklių ir brangių skaičiavimo resursų. Tačiau dengiant dideles teritorijas, ypač povandenines ar oro, elektros energijos ir laiko sąnaudos būtų per didelės. Todėl praktikoje dažniausiai taikomi labiau struktūruoti metodai, kurie leidžia sistemingai padengti visą aplinką.

2.1. ZPKP algoritmai

Šiame skyriuje apžvelgiama skirtinių ZPKP algoritmai, siekiant nustatyti kokio kuriamo įrankio funkcionalumo reikia, norint juos analizuoti ir tirti.

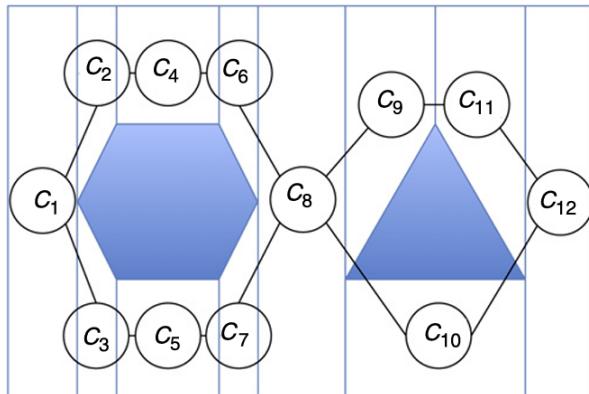
2.1.1. Klasikiniai algoritmai

Klasikiniai tikslūs sritinės dekompozicijos metodai (angl. Classical exact cellular decomposition methods) grindžiami geometrine struktūra, kurioje zona skaidoma į nesikertančių posričių – ląstelių – visumą. Šių ląstelių sąjunga užpildo visą zoną, o kiekvienoje ląstelėje padengimo maršrutas nustatomas paprastais judėjimais, pavyzdžiui, pirmyn-atgal modeliu (žr. 2 pav.). Tokiu būdu padengimo problema supaprastinama iki judėjimo sekos tarp ląstelių planavimo.



2 pav. Bustrofedono kelias [4].

Dvi ląstelės laikomos gretimomis (angl. Adjacent), jei jos turi bendrą ribą. Gretimumo grafas (angl. Adjacency graph) gali būti naudojamas ląstelinės dekompozicijos atvaizdavimui: grafo viršūnės žymi ląsteles, o briaunos — gretimumo ryšius tarp jų (žr. 3 pav.). Tikslias ląstelinės dekompozicijas galima sudaryti brėžiant per erdvę tiesę (pvz., iš kairės į dešinę). Ląstelių ribos formuoojamos tada, kai braukiančioji tiesė (angl. Sweep line) susiduria su tam tikru jvykiu. Pavyzdžiui, jvykiu gali būti laikoma kliūties ar zonas viršūnė.



3 pav. Zonas suskaidymas ląstelėmis ir gretimumo grafo sudarymas naudojant trapecinės dekompozicijos metodą [6].

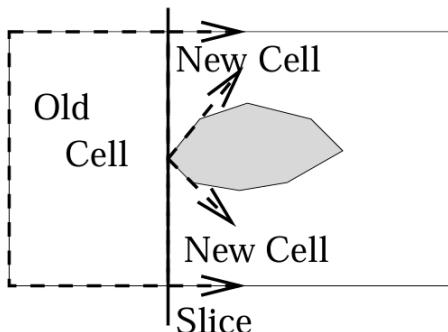
Paprastai, algoritmai, paremti tikslios sritinės dekompozicijos metodais, padengimo maršrutą sugeneruoja per 2 žingsnius. Pirmiausia, zona suskaidoma į ląsteles ir dekompozicija išsaugoma kaip gretimumo grafas. Toliau, apskaičiuojama seka per gretimumo grafą, kuri aplanko kiekvieną grafo viršūnę tiksliai vieną kartą. Gauta seka néra faktinis maršrutas, kiekvienai ląstelei ir keliui tarp ląstelių yra generuojamas atskiras, konkretus maršrutas.

Toliau aptariami skirtingi dekompozicijos metodai. Trapecinė dekompozicija (angl. Trapezoidal decomposition) ir bustrofedono dekompozicija (angl. Boustrophedon decomposition), tai du puikiai žinomi neprisitaikantys ląstelinės dekompozicijos metodai, kurie padėjo pagrindus ZPKP uždavinio sprendime. Iškiloji dekompozicija (angl. Convex Decomposition), tai metodas [13], kurio neriboj braukimo tiesė, nes ląstelės skaidomos pagal vidinių kampų geometriją.

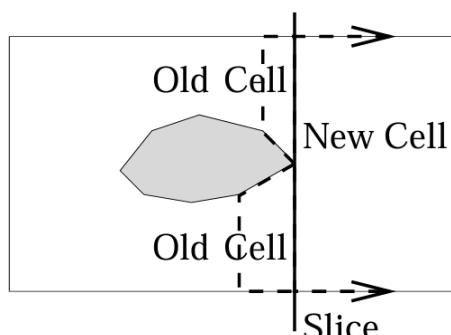
2.1.1.1 Trapecinė dekompozicija

Trapecinės dekompozicijos metodas [9] remiasi tuo, kad vertikali linija, vadinama pjūviu (angl. Slice), braukiamą iš kairės į dešinę per apribotą aplinką, kurioje yra daugiakampių kliūčių. Ląstelės

formuojamos atliekant atidarymo ir uždarymo operacijų seką, kurios jvyksta, kai pjūvis susiduria su jvykiu, t. y. kai pjūvis kerta daugiakampio viršūnę. Yra trys jvykių tipai: vidinis, išorinis ir vidurinės. Vidinio jvykio metu esama ląstelė uždaroma (taip užbaigiamos jos konstrukcija) ir atidaromos dvi naujos ląstelės (taip pradedama jų konstrukcija) (žr. 4 pav.). Išorinės jvykis yra atvirkščias: dvi ląstelės uždaromos, o viena nauja atidaroma (žr. 5 pav.). Vidinė jvykė galima suvokti kaip vienos ląstelės suskaidymą į dvi, o išorinė jvykė — kaip dviejų ląstelių susijungimą į vieną. Vidurinio jvykio metu esama ląstelė uždaroma ir suformuojama nauja. Šių operacijų rezultatas — apribota aplinka, suskaidyta į trapecinės ląsteles.

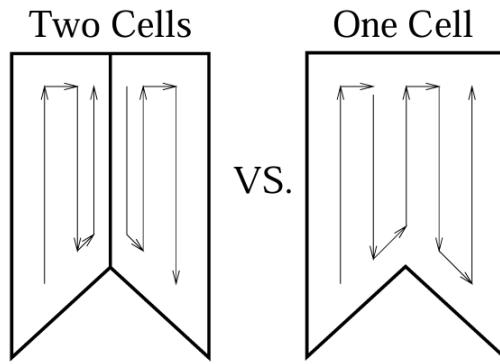


4 pav. Vidinis jvykis [4].

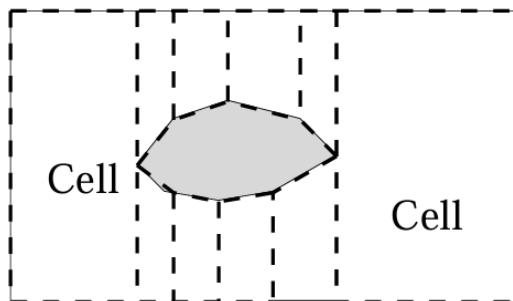


5 pav. Išorinis jvykis [4].

Trapecinės dekompozicijos metodui būdinga tai, kad dėl pilno aprėpties užtikrinimo robotui tenka atlikti daug perteklinių judesių pirmyn ir atgal. Kairėje 6 paveikslė pusėje matyti, jog robotui reikia atlikti dar vieną papildomą išilginį judesį, kad padengtų likusią trapecinės ląstelės dalį. Tokį judesį galima laikyti tam tikra „kaina“, leidžiančia garantuoti visos aplinkos išsamų padengimą. Jeigu daugiakampės kliūties ar apribota aplinka turi daug viršūnių, tarp kurių susidaro už robotą mažesni tarpai, tuomet sukuriama daug tarpusavyje nepersidengiančių ląstelių, tačiau roboto aprėptis vis tiek persidengia ir yra perteklinė. Kitas trapecinės dekompozicijos trūkumas yra tas, kad ši technika taikoma tik daugiakampėms aplinkoms.



6 pav. Mažiau ląstelių yra geriau [4].

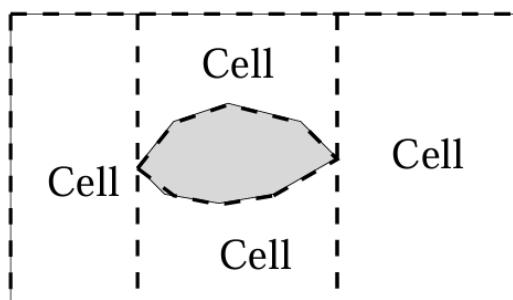


7 pav. Trapecinė dekompozicija [4].

2.1.1.2 Bustrofedono dekompozicija

Bustrofedono ląstelinė dekompozicija yra trapecinės dekompozicijos patobulinimas, sukurtas siekiant sumažinti perteklinių išilginių judesių skaičių. Kaip ir trapecinė dekompozicija, šis metodas daro prielaidą, jog kliūtys yra daugiakampės, o aplinka iš anksto žinoma, todėl jis priskiriamas prie neprisitaikančių metodų.

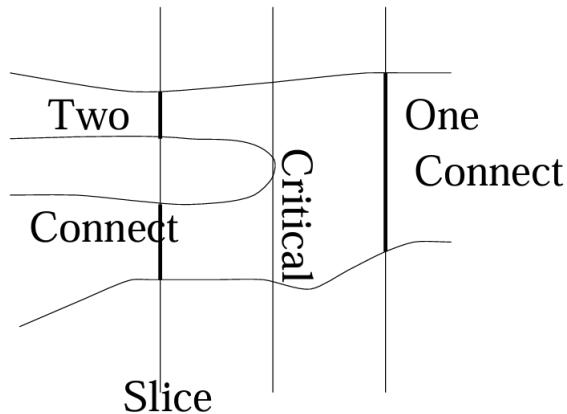
Visos ląstelės tarp vidinio ir išorinio įvykių sujungiamos į vieną ląstelę. Palyginus trapecinę dekompoziciją (žr. 7 pav.) su bustrofedono dekompozicija (žr. 8 pav.), matyti, kad bustrofedono dekompozicija turi mažesnį ląstelių skaičių.



8 pav. Bustrofedono dekompozicija [4].

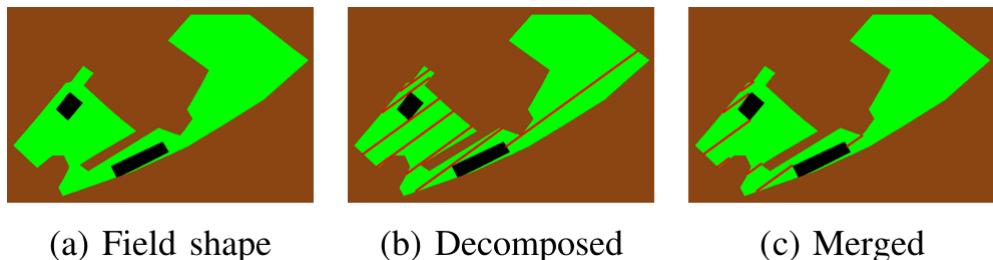
Užuot naudojus daugiakampių struktūrą vidiniams ir išoriniams įvykiams nustatyti, šiame metode remiamasi pjūvio (angl. Slice) jungiamumo pokyčiais, kuriais nustatomas įvykis (angl. Event). Toks įvykis paprastai vadinamas kritiniu tašku (angl. Critical point), ir jis taikomas įvairiose maršruto

planavimo metodikose. Naudodamas kritinius taškus robotas gali atlikti zonas padengimą ir kreivose, tolyginėse aplinkose (žr. 9 pav.).



9 pav. Kritiniai taškai yra taškai, kuriuose pasikeičia pjūvio būsena, pvz. iš dviejų pjūvio atkarų susijungia į vieną [4].

Šį metodą riboja tai, kad naudojama tik viena pjūvio kryptis. Kompleksiškose aplinkose vienos krypties pjūvis būtų optimalus tik daliai lastelių, todėl nukenčia bendras rezultatas. Rezultatui pagerinti yra atliekama metodo modifikacija [16], kuri parenka optimalų pjūvio tiesės kampą, minimuojant dengiamajį plotą dalijančiu lastelių skaičių ir atliekant gautų lastelių apjungimą (žr. 10 pav.), tai leidžia sumažinti persidengiančio kelio ilgį.



10 pav. Optimizuota lastelinė dekompozicija: (a) zona su kliūtimis, (b) dengiamas plotas padalintas į 12 lastelių, (c) galutinis rezultatas po lastelių apjungimo [16].

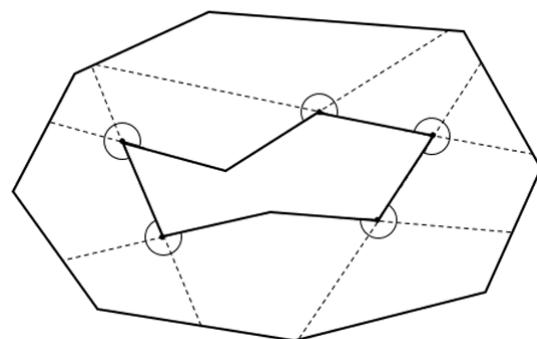
Bendrai, pjūviu pagrįstos dekompozicijos metodai turi pritaikymo apribojimą. Jie nepritaikyti situacijoms, kai daugiau nei viena viršūnė kerta pjūvio tiesę, todėl visos viršūnės turi būti unikalios pjūvio krypties atžvilgiu. Kai aplinka sudaryta iš sudėtingų daugiakampių, tai pasunkina optimalios krypties paiešką, nes ne visos kryptys gali būti naudojamos.

2.1.1.3 Iškiloji dekompozicija

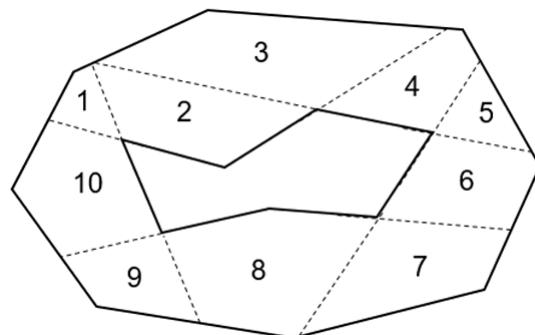
Iškiliosios dekompozicijos metodas [13] išvengia pjūvio tiesės apribojimų, atliekant dekompoziciją pagal aplinkos geometriją. Algoritmas susideda iš keturių žingsnių:

1. Nustatomi daugiakampių vidiniai (nukreipti į dengiamą plotą) kampai, viršijantys 180° .

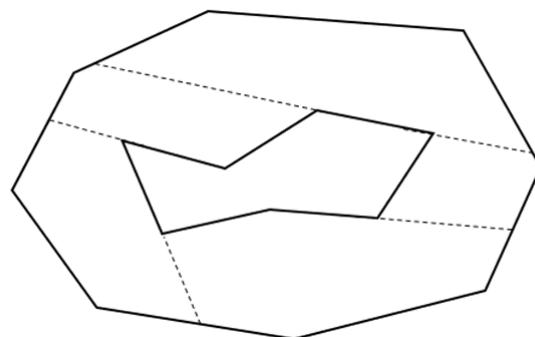
2. Šiuos kampus sudarančios kraštinės pratęsiamos dengiamame plote tol, kol pasiekia jo ribą arba susiduria su kliūtimi (žr. 11 pav.). Tokiu būdu dengiamasis plotas padalijamas į iškiluosius daugiakampius (daugiakampius, kurių visi vidiniai kampai mažesni nei 180°).
3. Iš gautų daugiakampių (žr. 12 pav.) sukuriamas rinkinys, su visais įmanomais iškilaisiais daugiakampiais, kurie gali būti sudaryti iš daugiakampių, gautų antrame žingsnyje.
4. Išrenkamas toks rinkinio poaibis (žr. 13 pav.), kuris padengia visą plotą be persidengimų ir minimuoja bendrą daugiakampių plotį (angl. width — trumpiausią atstumą tarp dviejų lygiagrečių tiesių, kurios pilnai aprėpia daugiakampį).



11 pav. Kraštinių pratęsimas [13].



12 pav. Dengiamojo ploto suskaidymas į iškiluosius daugiakampius [13].



13 pav. Galutinis rezultatas, radus optimalų apjungtų daugiakampių poaibj [13].

Trečiajame žingsnyje skaičiavimo laikas auga eksponentiškai dėl visų įmanomų kombinacijų tikrinimo. Todėl sudėtingose aplinkose būtina taikyti euristikomis grįstus metodus.

2.1.2. Euristikomis grįsti algoritmai

Euristiniai algoritmai, skirtingai nuo klasikinių, ne visada garantuoja optimalų maršrutą, tačiau siekia rasti „pakankamai gerą“ sprendimą per trumpesnį laiką. Šių metodų taikymas tampa būtinas sudėtingose aplinkose, kur tikslų skaičiavimų laikas auga eksponentiškai arba kai aplinka nėra pilnai žinoma iš anksto.

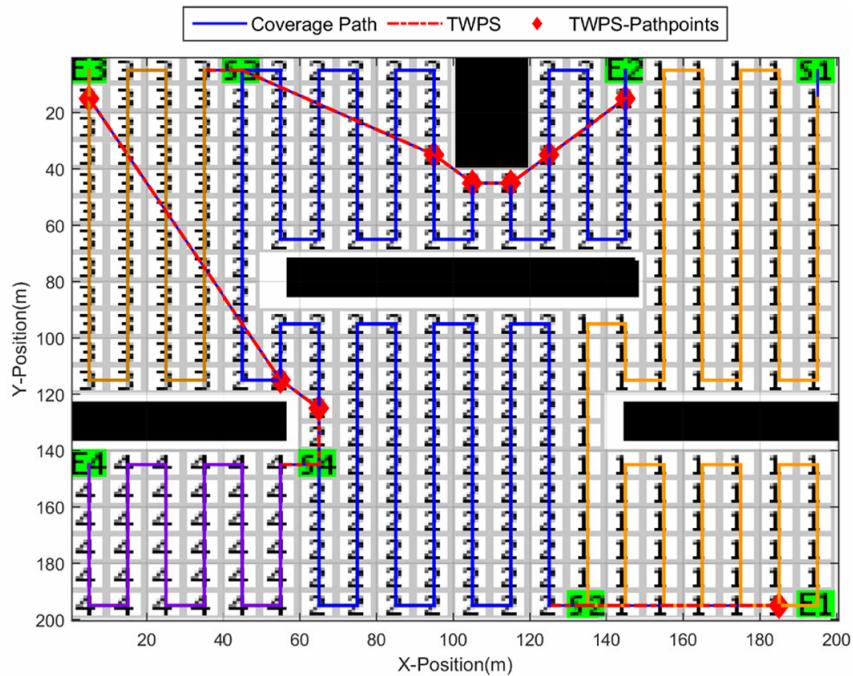
2.1.2.1 Iškiloji dekompozicija (modifikacija)

Iškilosios dekompozicijos metodo modifikacija [13] sugeneruotų daugiakampių apjungimui naudoja algoritmą, kuris, vietoj visų galimų daugiakampių kombinacijų tikrinimo, kuria kiekvieno dydžio i junginius tik iš dviejų mažesnių, vienodo dydžio $\frac{i}{2}$ junginių. Tai drastiškai sumažina paieškos erdvę, pagreitina skaičiavimus ir vis tiek pasiekia beveik optimalius rezultatus, nors kartais praleidžiamos geriausios kombinacijos.

2.1.2.2 Prisitaikantis zonos padengimo kelio planavimas

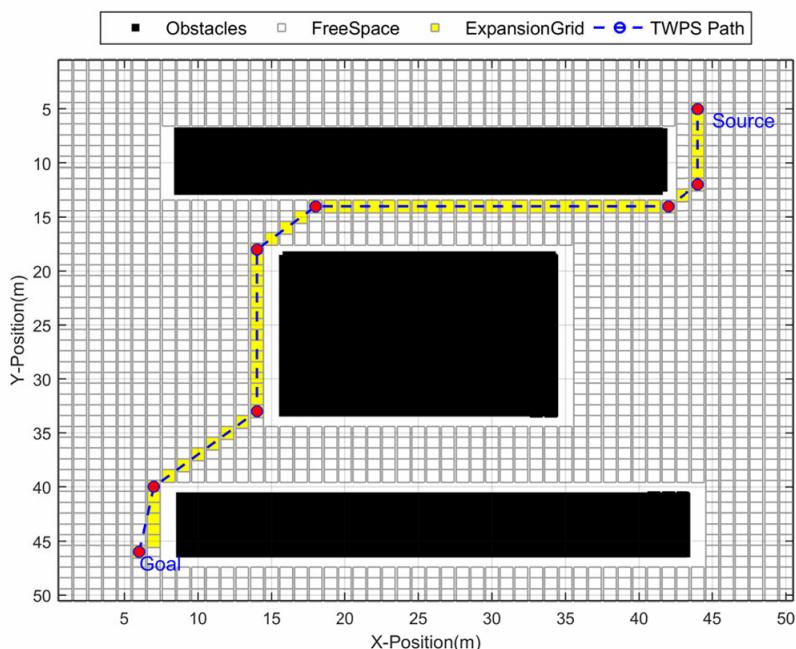
Ne visada aplinka žinoma iš anksto ar būna statiska, todėl atsiranda poreikis naudoti prisitaikančius metodus. Tokie metodai dažnai susidaro iš 2 dalių. Viena dalis atsakinga už zonos padengimą ir orientacinių (angl. Backtracking) taškų žymėjimą, o kita dalis — maršruto paieškai tarp kritinių taškų, kurie susidaro, kai sustabdomas zonos dengimas, ir orientacinių taškų.

Pasiūlytas skaičiavimams nebrangus ir efektyvus prisitaikantis zonos padengimo kelio planavimo metodas [8]. Padengimo užduotis atliekama taikant bustrofedoninį judėjimą kartu su efektyvia grįžimo technika (žr. 14 pav.), vadina dvikryptė artumo paieška (angl. Two-Way Proximity Search, TWPS).



14 pav. Žemėlapis su sugeneruotu padengimo keliu ir TWPS maršrutu [8].

Siūlomas algoritmas yra godžios (angl. Greedy) strategijos variacija, paremta Witkowskio algoritmu [19], kuri sugeneruoja trumpiausią įmanomą grjžimo kelią (žr. 15 pav.).

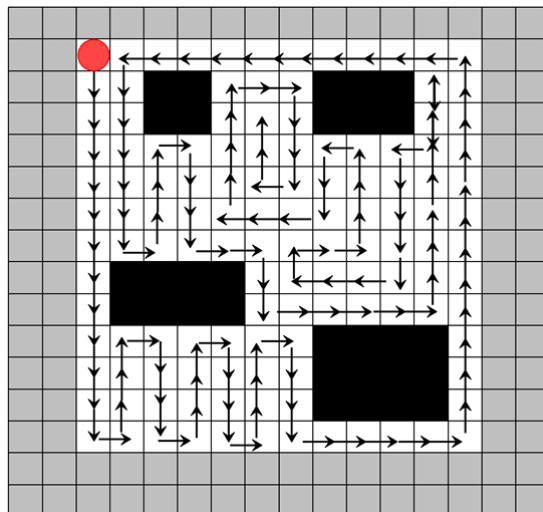


15 pav. Sugeneruotas kelias tarp 2 taškų naudojant TWPS algoritmu [8].

Norint pilnai pritaikyti šį algoritmą aplinkoms su dinamiškomis kliūtimis, reikėtų modifikuoti grjžimo algoritmą, kad pasikeitus kliūčių pozicijoms, jų būtų galima išvengti.

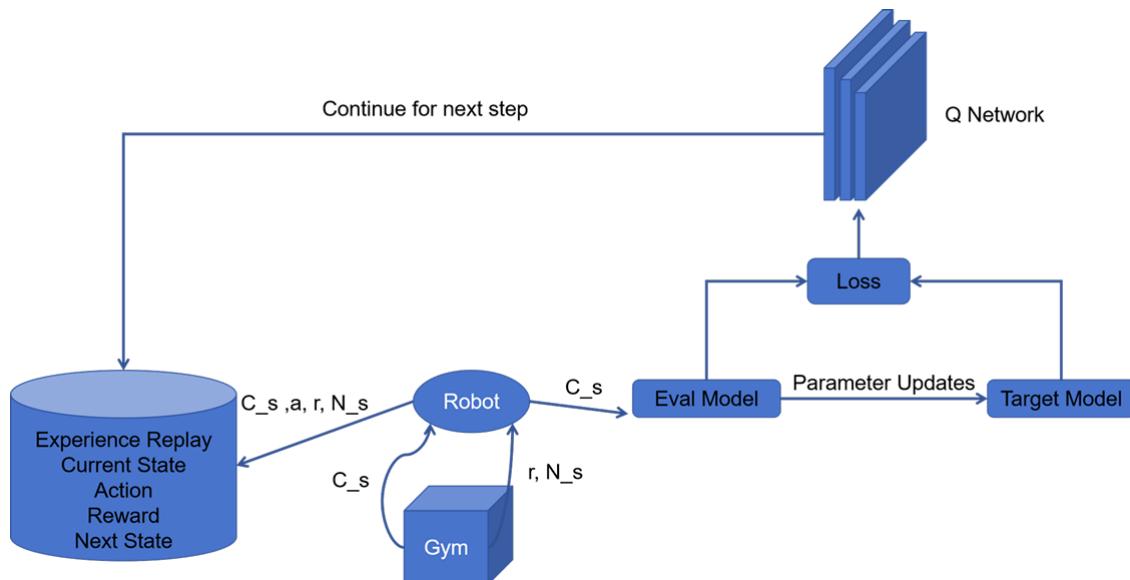
2.1.2.3 Re — DQN algoritmas

Re — DQN (angl. Reinforced Deep Q-Network) yra giliojo stiprinimo mokymosi algoritmas [2], skirtas pilnam vejos piovimo robotų teritorijos padengimo planavimui. Algoritmas veikia tinklelio pagrindo aplinkoje (žr. 16 pav.). Aplinkos modelis apima tiek statinius, tiek dinamiškai judančias kliūtis, o pats algoritmas veikia prisitaikymo režimu — aplinka gali būti dalinai arba visiškai nežinoma.



16 pav. Padengimo trajektorija, tinklelio aplinkoje [2].

Algoritmo veikimo principas grindžiamas neuroniniu tinklu (žr. 17 pav.), kuris mokosi prognozuoti geriausią veiksmą kiekvienoje situacijoje, gaudamas grįztamąjį ryšį iš aplinkos per atlygio funkciją. Pasiūlytas Re — DQN modelis išsiskiria keliomis naujovėmis: naudoja dinaminę tyrinėjimo strategiją su smalsumo skatinimu, kuri skatina robotą tyrinėti neaplankytas teritorijas, taiko dinaminę įvesties struktūrą, gebančią prisitaikyti prie kintamo kliūčių skaičiaus ir naudoja sudėtinės atlygio funkcijas, kurios vertina naujo ploto padengimą, baudžia už susidūrimus ir skatina efektyvesnius maršrutus. Metodas integruoja papildomą paskatinimo sluoksnį, kuris padeda išvengti kliūčių ir stabilizuojama mokymosi procesą.

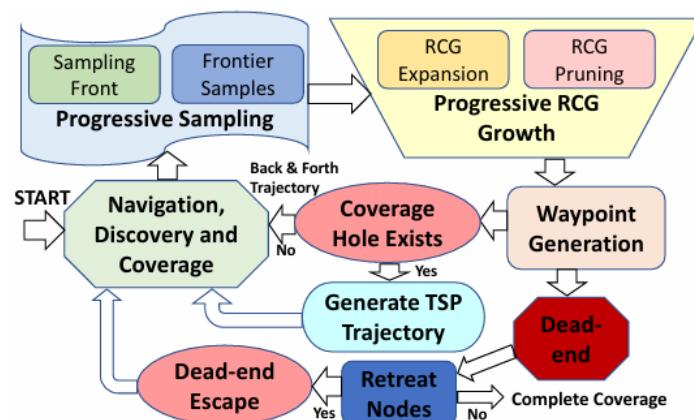


17 pav. DQN karkasas [2].

Pagrindiniai Re — DQN privalumai yra aukštas prisitaikymo lygis dinamiškai besikeičiančioje aplinkoje, efektyvesnis naujo ploto tyrinėjimas lyginant su standartiniu DQN, trumpesni maršrutai su mažesniu pasikartojimų kiekiu, ir geresnis stabilumas mokymosi metu. Pagrindiniai trūkumai apima vidutinę algoritmo sudėtingumą, kuris reikalauja papildomų skaičiavimo resursų, ilgą mokymosi laiką tinkleliuose su dideliu kliūčių tankiu, ir riziką įstrigtį lokaliuose optimumuose labai sudėtingose aplinkose.

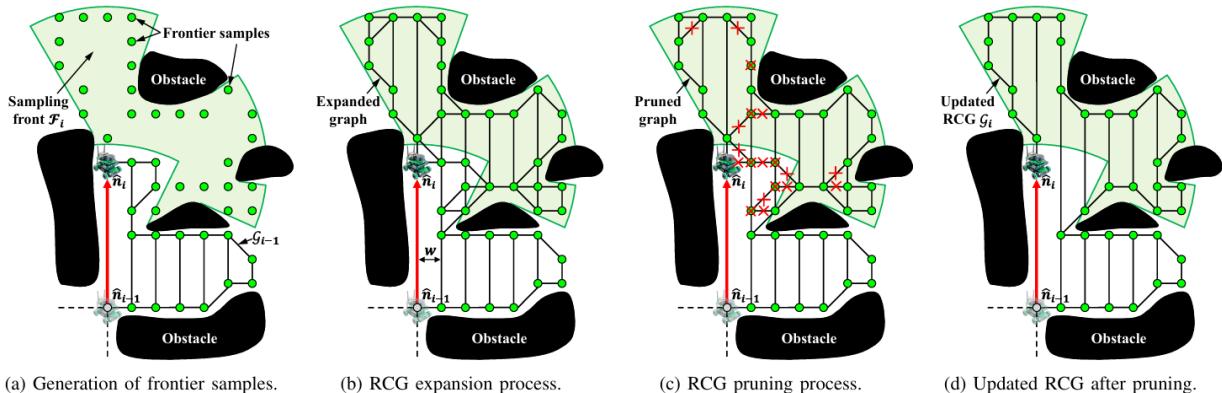
2.1.2.4 C* algoritmas

C* — intimis pagrįstas algoritmas [17], skirtas realaus laiko padengimo maršrutų planavimui nežinomose aplinkose. C* algoritmas yra paremtas sparčiai padengiančio grafo (angl. Rapidly Covering Graph, SPG) modeliu, kuris seka padengimo eigą ir generuoja padengimo trajektoriją dar nepadengtose srityse. SPG yra konstruojamas palaipsniui imant mėginius iš aplinkos, roboto navigacijos metu. Algoritmo pilna operacijų schema pateikta 18 paveiksle.



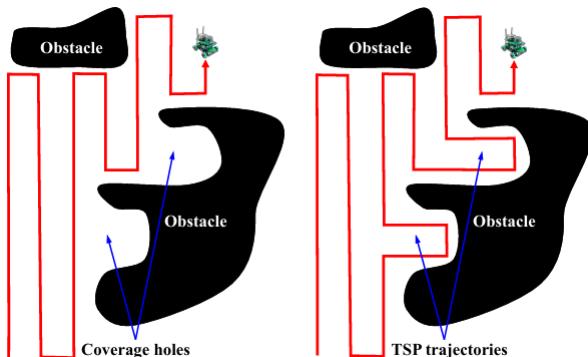
18 pav. Algoritmo operacijos [17].

SPG pasižymi reto grafo (angl. sparse-graph) struktūra, suformuota taikant efektyvius mēginių ēmimo ir genējimo metodus. Jo viršūnės ir briaunos sudaro galimus aplinkos trajektorijos tarpinius taškus ir segmentus. Efektyvi retos struktūros SPG leidžia generuoti ne trumparegiškus aplinkos trajektorijos tarpinius taškus, kurie nukreipia robotą iš aklaviečių, atsitraukiant iki artimiausių neištirtų viršūnių.



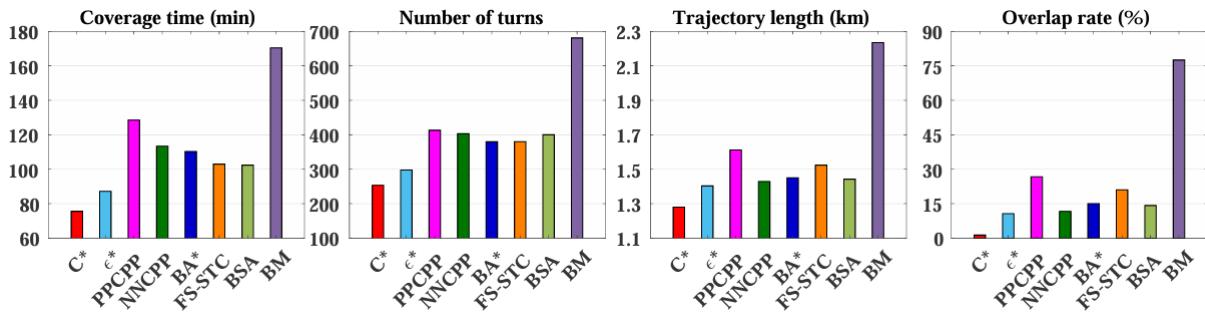
19 pav. SPG sudarymas, plečiant grafo pridedant naujus aplinkos mēginius ir pašalinant nenaudingas viršūnes bei briaunas [17].

Išskirtinė C* algoritmo savybė yra progresyvus mēginių ēmimas (žr. 19), kuris panaikina būtinybę taikyti aplinkos ląstelinę dekompoziciją. C* sukuria efektyvų pirmyn — atgal padengimo maršrutą paprastose srityse, o sudėtingesnėse, mažo ploto srityse maršrutas randamas keliaujančiojo pirklio uždavinio sprendimo pagrindu. Šios mažo ploto, sudėtingos sritys, vadinamos padengimo skylėmis (angl. coverage holes), yra apsuotos kliūčių ir jau padengtų sričių. C* aktyviai aptinka ir padengia padengimo skylių vykdymo eigoje (žr. 20, taip sumažinamas bendras padengimo laikas, nes išvengiama ilgesnių grįžimo trajektorijų iš nutolusių sričių, reikalingų šioms skylėms padengti vėliau.



20 pav. Padengimo skylių prevencija, naudojant keliaujančio pirklio metodus [17].

Analitiškai įvertintas (žr. 21 pav.) algoritminis C* paprastumas ir mažas skaičiavimo sudėtingumas lyginant su kitais moderniais ZPKP metodais, leidžia algoritmą įgyvendinti ir naudoti realaus laiko, robotų sistemose su vidiniais skaičiavimo resursais.



21 pav. Rezultatų palyginimas su kitais pažangiais metodais [17].

2.2. ZPKP algoritmų apibendrinimas

Zonos padengimo kelio planavimo algoritmai (žr. 2 lentelę) skirstomi į klasikinius (garantuojančius visišką padengimą) ir euristinius (ieškančius pakankamai gero sprendimo, nustatant specifines taisykles). Apžvelgti klasikiniai algoritmai remiasi ląstelinės dekompozicijos principais, zoną skaidant į nesikertančias posritis, o judėjimas tarp jų planuoojamas per gretumumo grafą. Trapecinė dekompozicija naudoja vertikalų braukimą per daugiakampę aplinką, sukurdama trapecines ląsteles prie kiekvienos viršūnės. Bustrofedono dekompozicija tobulina šį metodą, sujungdama ląsteles tarp vidinio ir išorinio žvykių, taip sumažindama perteklinius judeisius. Iškiloji dekompozicija išvengia pjūvio tiesės apribojimų, skaidydama zoną pagal vidinių kampų geometriją į iškiliuosius daugiakampius. Prisitinkantys metodai, tokie kaip TWPS algoritmas, nesiremia išankstinėmis žiniomis apie aplinką ir naudoja realaus laiko jutiklių duomenis zonos padengimui dinamiškose aplinkose. C^* yra imtimis pagristas algoritmas, naudojantis sparčiai padengiančio grafo modelj su progresiniais mēginiiais, vykdymo eigoje padengiantis padengimo skyles. Re — DQN algoritmas naudoja giliojo stiprinimo mokymosi metodą, su patobulintomis tyrinėjimo strategijomis ir atlygio funkcijomis.

2 lentelė. ZPKP algoritmų palyginimas

Algoritmas	Formatas	Tipas	Komentarai	
			Privalumai	Trūkumai
Trapezinė dekompozicija	Daugia-kampė	Neprisitaikanti	<ul style="list-style-type: none"> Paprasta realizacija su vertikaliu braukimu. 	<ul style="list-style-type: none"> Aplinkose su daugybe viršūnių sukuriama daug ląstelių. Neveikia, kai daugiau nei 1 viršūnė kerta pjūvio tiesę.
Bustrofedono dekompozicija	Daugia-kampė	Neprisitaikanti	<ul style="list-style-type: none"> Mažesnis ląstelių skaičius lyginant su trapezine. 	<ul style="list-style-type: none"> Neveikia, kai daugiau nei 1 viršūnė kerta pjūvio tiesę.

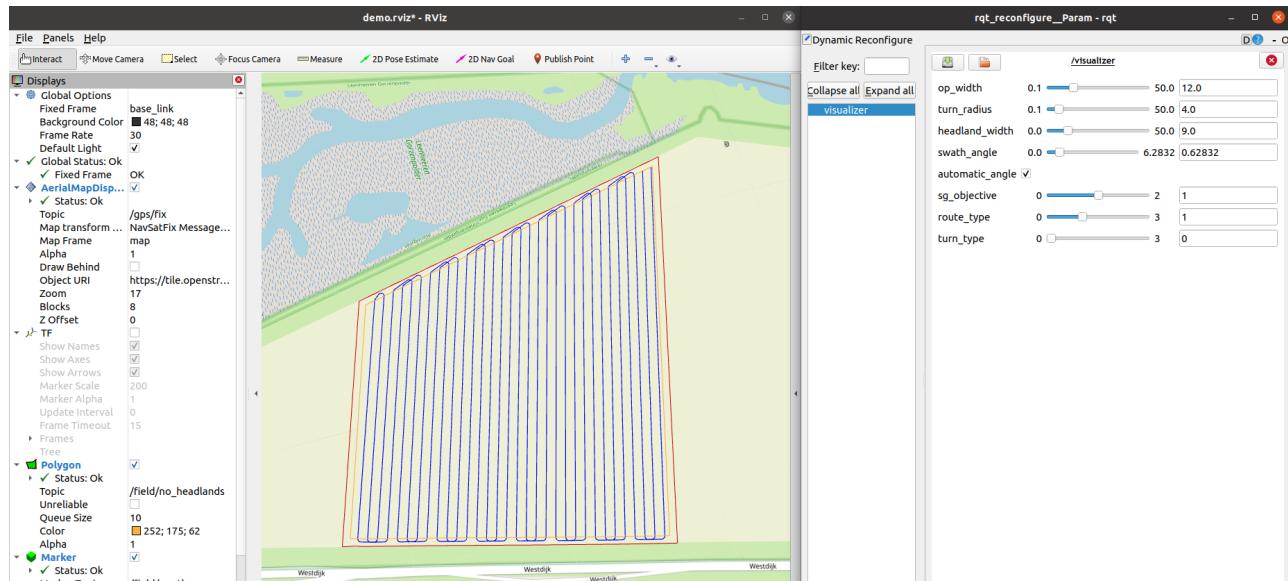
Iškiloji dekompozicija	Daugia-kampė	Neprisitaikanti	<ul style="list-style-type: none"> Išvengia pjūvio tiesės apribojimų. Optimalesnės ląstelės lyginant su Bustrofeno dekompozicija. 	<ul style="list-style-type: none"> Skaičiavimo laikas auga eksponentiškai.
Iškiloji dekompozicija (modifikacija)	Daugia-kampė	Neprisitaikanti	<ul style="list-style-type: none"> Euristiniai metodai pagreitina skaičiavimus, pasiekiant beveik optimalius rezultatus. 	<ul style="list-style-type: none"> Kartais praleidžiamos geriausios kombinacijos.
Prisitaikantis (TWPS)	Tinklelis	Prisitaikanti	<ul style="list-style-type: none"> Skaičiavimams nebėrangus ir efektyvus. Nesiremia išankstinėmis žiniomis apie aplinką. Generuoja trumpiausią įmanomą grįžimo kelią. 	<ul style="list-style-type: none"> Reikia modifikuoti grįžimo algoritmą dinamiškoms kliūtimis.
Re — DQN	Tinklelis	Prisitaikanti	<ul style="list-style-type: none"> Aukštas prisitaikymo lygis. 	<ul style="list-style-type: none"> Reikalangi dideli skaičiavimo resursai. Rizika įstrigtį lokaliuose optimumuose.
C*	Grafas	Prisitaikanti	<ul style="list-style-type: none"> Skaičiavimams nebėrangus ir efektyvus, galima naudoti realiu laiku. Nesiremia išankstinėmis žiniomis apie aplinką. Efektyvus ir paprastas padengimas. Nepalieka padengimo skylių. Pritaikoma kelių robotų komandai. Atsižvelgiama į energijos sąnaudų apribojimus. 	<ul style="list-style-type: none"> Reikalauja išskirtinės aplinkos realizacijos.

2.3. Sukurti sprendimai ZPKP tyrimų srityje

ZPKP sritis yra plačiai tiriamą, tačiau beveik nėra ZPKP algoritmų įgyvendinimo kodo. Mokslinių straipsnių autorai pateikia tik algoritmų aprašymus ar pseudo kodą. Dėl šios priežasties, ZPKP tyrimai vyksta lėtai, nes kiekvieną algoritmą, kurį norima palyginti, tenka įgyvendinti pačiam.

Sukurti atvirojo kodo sprendimai turi labai ribotas galimybes ir yra specializuoti, kas apsunkina funkcionalumo praplėtimą. Išties naudingos ir pažangios, naudotojui patogios žiniatinklio programos kuriamos bendradarbiaujant su jmonėmis, todėl galimybė viešai paskelbtį jų išeities kodą yra apribota.

Projektas „Fields2Cover“ [12] yra vienas pažangiausių šios srities atvirojo kodo projektų. „Fields2Cover“ – tai atvirojo kodo biblioteka, skirta žemės ūkio transporto priemonių padengimo trajektorijų planavimui. „Fields2Cover“ suteikia vartotojo sąsają (22 pav.) ir programinį karkasą padengimo trajektorijų planavimui, naujų metodų kūrimui ir pažangių algoritmų palyginimui. Biblioteka pasižymi moduline ir plečiamą architektūrą, palaikančią įvairias transporto priemones ir tinkama skirtingoms taikymo sritims, įskaitant žemės ūkį. Pagrindiniai jos moduliai yra: galulaukių (angl. headlands) generatorius, važiavimo juostų (angl. swaths) generatorius, maršruto planuotojas ir trajektorijos planuotojas. Taip pat kaip papildinys pateikiama sąsaja su Robotų operacine sistema (ROS).



22 pav. „Fields2Cover“ vartotojo sąsaja [11]

Nors įrankis ir palengvina ZPKP tyrimų atlikimą, tačiau yra trūkumų, kurie apriboja ir pasunkina darbą su sistema ir reikalauja papildomo laiko, kad susipažinti ir išmokti dirbti su sistema:

- Palaikomas tik daugiakampiai grjstos, pastovios, iš anksto žinomos aplinkos.
- Sistema yra priklausoma nuo platformos. Sistemos instaliacija ištestuota tik „Ubuntu“ operaciniuje sistemoje, nėra instrukcijų kaip sistemą instaliuoti naudojant kitas operacines sistemas.
- Instaliavimo procesas nėra paprastas, reikalauja papildomo konfigūravimo.
- Sistemos turi vartotojo sąsaja, bet nėra aprašyta kaip ja naudotis.

- Naudotis biblioteka kodo lygmeniu nėra paprasta, reikalauja gilių bibliotekos žinių. Kodas limi-tuotas „C++“ ir „Python“ kalbomis.
- Projektas nėra aktyviai vystomas, paskutiniai pakeitimai atlikti prieš 10 mėnesių (2025 m. ba-landžio mėnesį).
- Pasirinkta architektūra nėra moderni, kas apsunkina tolimesnį projekto vykdymą.

Taigi, nors yra realizuotų ZPKP įrankių, norint pradėti atlikti tyrimus reikia jidéti daug pastangų ir laiko. Igyvendinti įrankiai nėra patogūs vartotojui, reikalauja stiprių programavimo žinių, gilaus bibliotekų supratimo.

3. Metodinė dalis

3.1. Sistemos paskirtis

Kuriama internetinė ZPKP (zonos padengimo kelio planavimo) simuliacinės įrangos sasaja skirta tyrėjams ir algoritmų kūrėjams, siekiantiems greitai konfigūruoti ir lyginti skirtingus padengimo algoritmus. Sistema suteikia vieningą darbalaukį aplinkų kūrimui, algoritmų paleidimui bei rezultatų analizei. Sasaja leidžia interaktyviai modeliuoti realistiškas zonas, įtraukiant statinius ir dinaminius objektus, bei vykdyti tiek pavienius, tiek pakartotinius scenarijus. Surinkti duomenys vizualizuojami, kaupiami ir padengiami kiekybiniais rodikliais.

3.2. Keliami reikalavimai sistemai

Identifikuoti šie reikalavimai sistemai:

- Sistema turi leisti kurti, importuoti, redaguoti ir trinti daugiakampėmis ribomis apibrėžtas zonas bei kliūtis, valdant jų parametrus viename darbalaukyje.
- Sistema turi leisti konvertuoti daugiakampes aplinkas į tinkleliu gręstas ir atvirkščiai.
- Sistema turi automatiškai įrašyti aplinkos būseną į duomenų bazę, leisti dirbtį skirtingose aplinkose.
- Sistema turi suteikti priemones algoritmo pasirinkimui ir parametru konfigūravimui bei sinchronizuoti algoritmų ir aplinkų suderinamumą.
- Sistema turi leisti kurti, redaguoti, kopijuoti ir trinti ZPKP testavimo scenarijus, kurie apjungia kelias aplinkas ir algoritmus.
- Sistema turi paleisti tiek pavienius, tiek testavimo scenarijais aprašytus ZPKP vykdymus, perduodant aplinkos ir algoritmo duomenis „JSON“ formatu išorinėms skaičiavimo sistemoms ir grąžintus rezultatus išsaugant duomenų bazę.
- Sistema turi apskaičiuoti ir saugoti padengimo kiekybinius rodiklius bei susieti juos su vykdymų istorija.
- Sistema turi vizualiai pateikti padengimo maršrutus darbalaukyje, pateikti suvestinius rezultatus.
- Sistema turi užtikrinti duomenų eksportavimą „JSON“ ir „CSV“ formatais.
- Sistema turi pateikti paaiškinamuosius pranešimus apie klaidas.
- Sistema turi suteikti darbalaukio navigacijos priemones, kad naudotojai galėtų tiksliai koreguoti ir peržiūrėti aplinkas.

3.3. Sistemos naudotojai

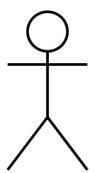
Sistema turi tik vieną naudotoją (žr. 3 lentelę). Kadangi programa kuriama lokaliam naudojimui, autentifikacijos funkcionalumas nėra reikalingas. Naudotojas turi pilną duomenų kontrolę, yra pilnai atsakingas už įrankio konfigūravimą ir naudojimą.

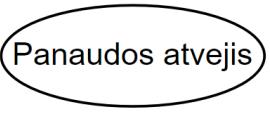
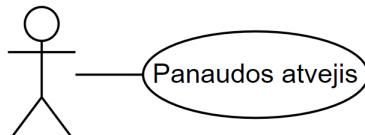
3 lentelė. Sistemos aktoriai

Pavadinimas	Apaščias
Naudotojas	<p>Naudotojas yra ZPKP algoritmų tyrėjas, kuris nori ištestuoti ir išanalizuoti savo algoritmą skirtingose aplinkose, palyginti rezultatus su kitais algoritmais.</p> <p>Naudotojas, turi pilnas teises naudoti sistemą.</p> <p>Naudotojas gali:</p> <ul style="list-style-type: none">• Naršyti darbalaukyje;• Valdyti ZPKP procesą;• Kontroliuoti, keisti aplinką;• Eksportuoti, importuoti duomenis.

3.4. Panaudojimo atvejų notacija

4 lentelė. Panaudojimo atvejų notacija

Elementas	Apaščias
<p>Aktorius (angl. Actor)</p>  <p>Aktorius</p>	<p>Aktorius yra sistemos naudotojas, kuris gali būti žmogus, mašina, kita sistema ar posistemė. Bet kas, kas sąveikauja su specifikuojama sistema.</p> <p>Aktoriai yra siejami su panaudojimo atvejais, kuriuose aprašomi sistemos panaudojimo scenarijai tenkinantys naudotojo reikalavimus.</p> <p>Skirtingi aktoriai gali vykdyti skirtingus panaudojimo atvejus.</p>

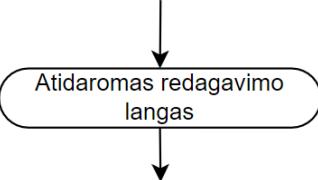
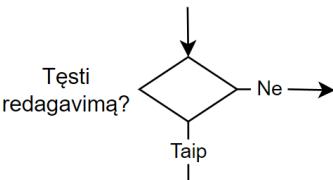
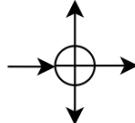
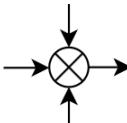
<p>Panaudojimo atvejis (angl. Use Case)</p> 	<p>Elementas, kuris aprašo sąveiką tarp naudotojo ir sistemos. T.y. kokių žingsnius turi atlikti naudotojas sistemoje, kad pasiektų tam tikrą tikslą.</p> <p>Kiekvienas panaudojimo atvejis yra aprašomas lentelėje, kuri susideda iš:</p> <ol style="list-style-type: none"> 1. Pavadinimas — nusakantis naudotojo siekiama tikslą. 2. Aprašymas — trumpai pakomentuojant kokiu tikslu yra vykdomas panaudojimo atvejis. 3. Diagrama — pateikiamas grafinis panaudojimo atvejo scenarijaus vykdymas arba naudotojo sasajos schema. 4. Apribojimai — prieš ir po sąlygos, kurios turi būti tenkinamos prieš pradedant vykdyti scenarijų arba norint pilnai jvykdyti scenarijų. 5. Reikalavimai — apibrėžia tam tikras taisyklės, kurios turi būti tenkinamos panaudojimo atvejo vykdymo metu. Bendrieji reikalavimai taikomi ne vienam, o keliems panaudojimo atvejams. 6. Scenarijai — sąveikos tarp naudotojo ir sistemos aprašymas, nurodant kiekvieną naudotojo vykdomą žingsnį sistemoje. Kiekvienas panaudojimo atvejis savyje gali turėti tik vieną pagrindinį scenarijų, kuriu pašiekiamas panaudojimo atvejo tikslas. Ir daug alternatyvų (išimčių), kurios apibrėžia tam tikrus pagrindinio scenarijaus neprivalomus vykdyti išsišakojimus (kitaip atliekami veiksmų sekas padedančias pasiekti tą patį tikslą).
<p>Panaudojimo ryšys (angl. Use)</p> 	<p>Nurodo, jog tam tikras aktorius gali vykdyti nurodytą panaudojimo atvejj.</p>

<p>Praplėtimo ryšys (angl. Extend)</p> <pre> graph TD A([Peržiūrėti ZPKP rezultatus]) --> <<extend>> B([Eksportuoti duomenis]) </pre>	<p>Panaudojimo atvejai gali būti sujungti praplėtimo ryšiu. Tai reiškia, kad vykdant vieno panaudojimo atvejo scenarijų, tam tikrame žingsnyje gali būti pradėtas vykdyti kitas alternatyvus (išimtinis) scenarijus. Tačiau alternatyva nebūtinai turi būti vykdoma.</p> <p>Paveiksle pavyzdys, kai peržiūrint ZPKP rezultatus tam tikrame žingsnyje turi būti galima eksportuoti rezultatų duomenis, tačiau eksportavimas gali būti ir nevykdomas priklausomai nuo naudotojo ar kitų apibrėžtų scenarijuje sąlygų pasirinkimo.</p>
<p>Įtraukimo ryšys (angl. Include)</p> <pre> graph TD A([Redaguoti objekto ribas]) --> <<include>> B([Pasirinkti objektą]) </pre>	<p>Panaudojimo atvejai gali būti sujungti įtraukimo ryšiu. Tai reiškia, kad vykdant vieno panaudojimo atvejo pagrindinį scenarijų tam tikrame žingsnyje yra būtina atlikti įtraukiamu panaudojimo atvejo scenarijų, kitaip norimas tikslas nebus pasiekamas.</p> <p>Paveiksle pavyzdys, kai redaguojant objekto ribas turi būtinai būti atliktas objekto pasirinkimo scenarijus, kitaip objekto ribų redaguoti nepavyks.</p>

3.5. Veiklos diagramos notacija

5 lentelė. Veiklos diagramos notacija

Elementas	Aprašymas
<p>Pradžios įvykis</p>	<p>Įvykis, nuo kurio pradedama vykdyti veikla. Pradžios įvykis gali turėti sąlygą, nuo kurios priklauso vykdymo eiga.</p>
<p>Pabaigos įvykis</p>	<p>Įvykis, kuris užbaigia veiklos vykdymą.</p>

Užduotis 	Atliekamas elementarus arba sudėtinis procesas. Gali būti vykdomas kitas funkcinis reikalavimas.
Vartai 	Veiklos srauto išsiskyrimas, pagal aprašytą sąlygą.
Loginis „OR“ operatorius 	Veiklos srautas išsiskiria dėl neapibrėžtų sąlygų, išsiskyrimo metu vykdoma viena, kelios ar visos atšakos.
Sujungimo operatorius 	Kelios srauto atšakos apjungiamos į vieną bendrą, naudojama pasiekti stabilią sistemos būseną nuo kurios procesas tęsiasi toliau.

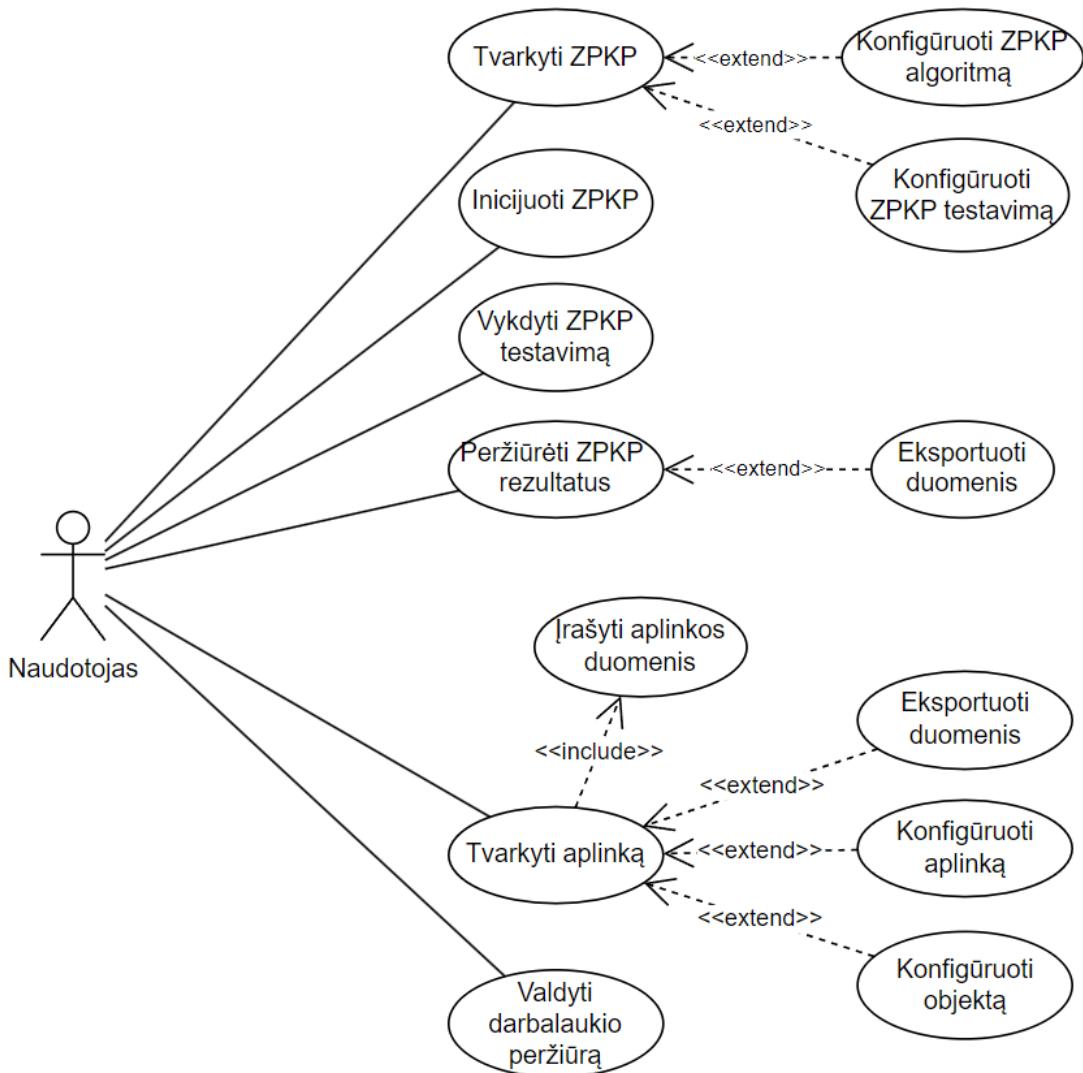
3.6. Detalizuoti funkciniai reikalavimai

Naudotojas yra pagrindinis sistemos aktorius, atsakingas už ZPKP ir aplinkos valdymą.

Naudotojas gali vykdyti ZPKP procesą 2 skirtingais būdais: vykdant su sukonfigūruota konkrečia aplinka ir algoritmu (vykdomas vienas, darbalaukyje užkrautas ir naudotojui atvaizduojamas scenarijus) arba vykdant testavimo scenarijus (scenarijus(-ai) sukurti iš anksto su sukonfigūruotomis aplinkomis ir algoritmais). Prieš pradedant galima konfigūruoti ZPKP proceso parametrus ir kurti testavimo scenarijus. Atlikus ZPKP procesą, rezultatus galima peržiūrėti ir eksportuoti, taip pat peržiūrėti aplinkos duomenis bei valdyti darbalaukio peržiūrą.

Naudotojas turi pilną aplinkos valdymo kontrolę: gali konfigūruoti aplinką ir joje esančius objektus, o aplinkos duomenys jrašomi automatiškai. Aplinkos konfigūracija gali būti importuojama ir eksportuojama naudojant išorinius failus.

Funkciniai naudotojo grafinės sąsajos reikalavimai pateikti grafine notacija (žr. 4 lentelę) — UML funkcių reikalavimų diagrama (žr. 23 pav.).

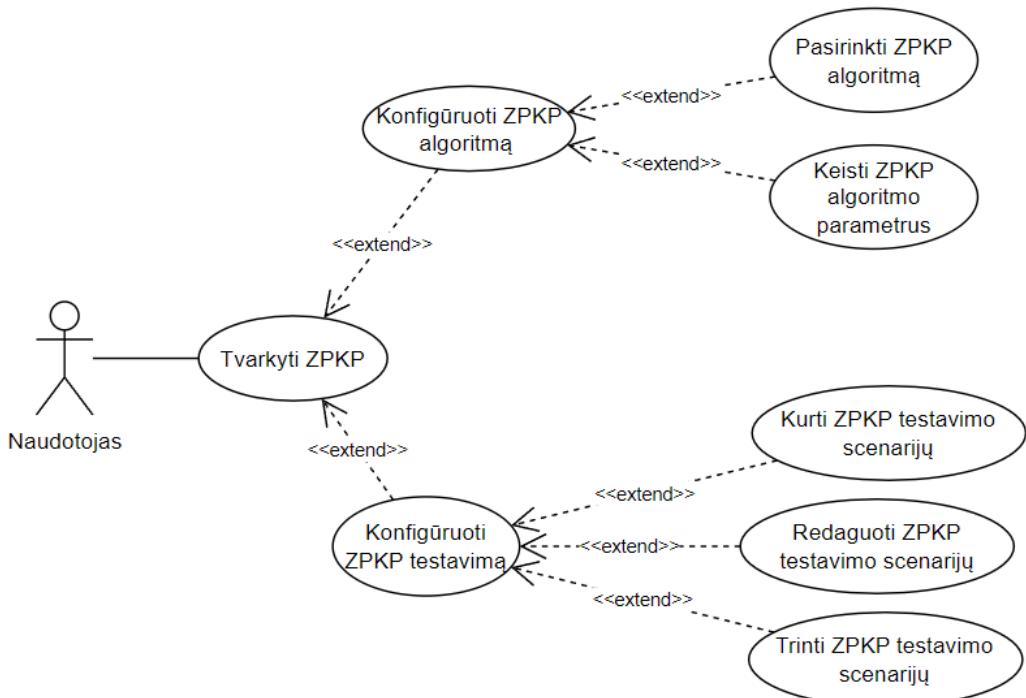


23 pav. Funkciniai reikalavimai

Toliau pateikti kiekvieno funkcinio reikalavimo aprašymai ir veiklos diagramos, pateiktos 5 lentelės notacija.

3.6.1. Funkcinis reikalavimas „Tvarkyti ZPKP“

ZPKP tvarkymas (24 pav.) — tai galimybė konfigūruoti ZPKP algoritmą ir ZPKP testavimą.



24 pav. Funkcinis reikalavimas „Tvarkyti ZPKP“

Konfigūruoti ZPKP algoritmą — tai veikla, kurios metu naudotojas pasirinktam ZPKP algoritmui gali keisti naudojamus parametrus.

Pasirinkti ZPKP algoritmą — tai ZPKP algoritmo konfigūravimo veikla, kurios metu naudotojas gali pasirinkti algoritmą su kuriuo nori dirbtį.

Keisti ZPKP algoritmo parametrus — tai ZPKP algoritmo konfigūravimo veikla, kurios metu naudotojas gali keisti pasirinkto algoritmo parametrus, tokius kaip kelio plotis ir kelio persidengimas.

Konfigūruoti ZPKP testavimą — tai veikla, kurios metu naudotojas gali kontroliuoti testavimo scenarijus, paruošti juos testavimo vykdymui.

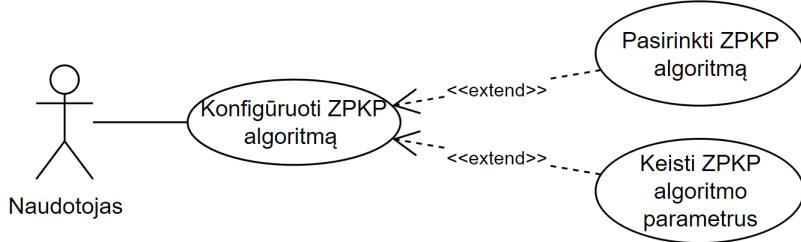
Kurti ZPKP testavimo scenarijų — tai ZPKP testavimo konfigūravimo veikla, kurios metu naudotojas gali sukurti testavimo scenarijų pasirinkdamas aplinką ir algoritmą.

Redaguoti ZPKP testavimo scenarijų — tai ZPKP testavimo konfigūravimo veikla, kurios metu naudotojas gali redaguoti ZPKP testavimo scenarijų, keisdamas scenarijaus aplinką, algoritmą.

Trinti ZPKP testavimo scenarijų — tai ZPKP testavimo konfigūravimo veikla, kurios metu naudotojas gali pašalinti ZPKP testavimo scenarijų.

3.6.1.1 Funkcinis reikalavimas „Konfigūruoti ZPKP algoritmą“

Konfigūruoti ZPKP algoritmą (žr. 25 pav.) — tai veikla, kurios metu naudotojas pasirinktam ZPKP algoritmui gali keisti naudojamus parametrus.

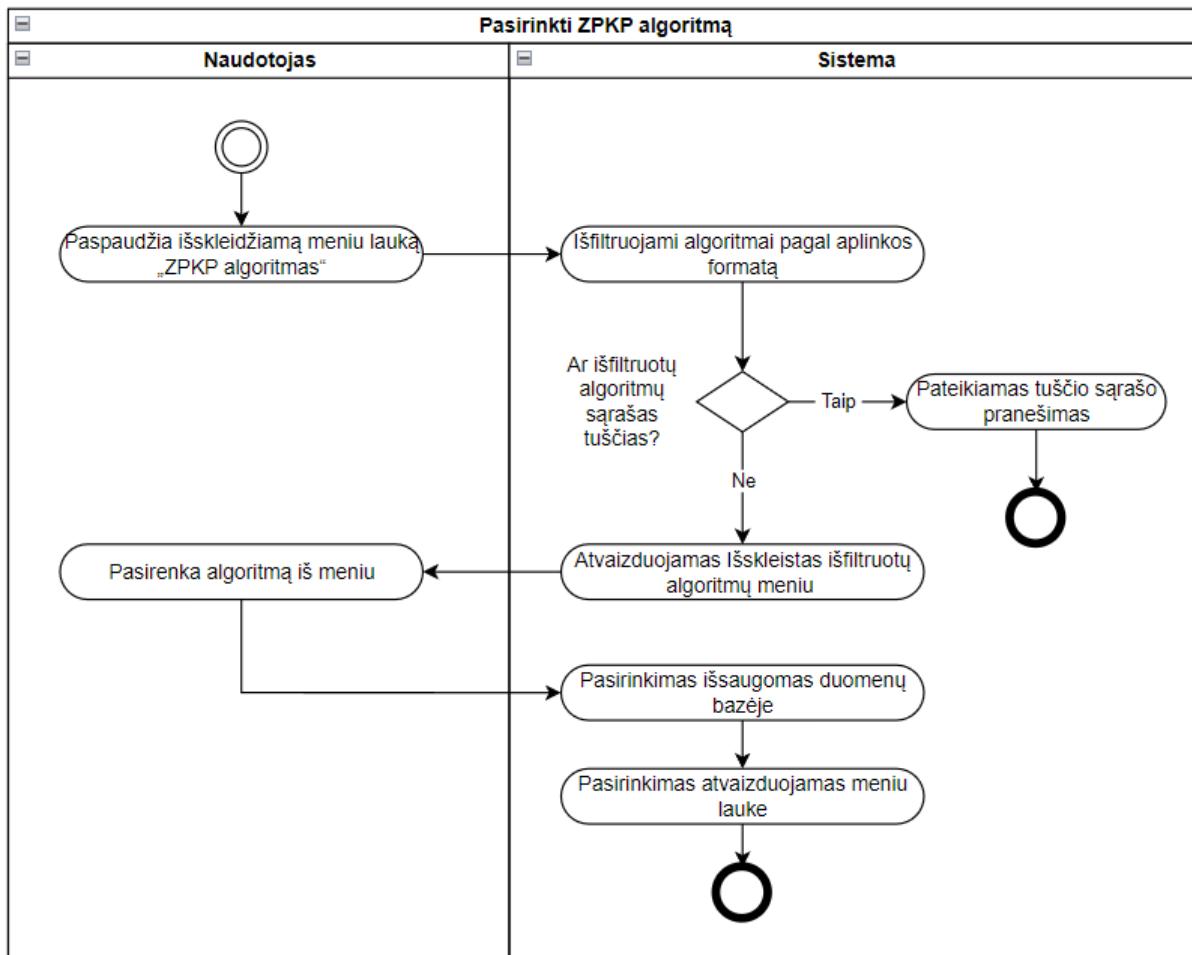


25 pav. Funkcinis reikalavimas „Konfigūruoti ZPKP algoritmą“

3.6.1..1.0.1 Funkcinis reikalavimas „Pasirinkti ZPKP algoritmą“

6 lentelė. Funkcinis reikalavimas „Pasirinkti ZPKP algoritmą“

Trumpas aprašymas	Naudotojas gali pasirinkti ZPKP algoritmą darbui pasirinktoje aplinkoje.
Scenarijai:	
Pagrindinis	<p>1. Naudotojas pasirenka ZPKP algoritmą.</p> <p>1.1. Naudotojas paspaudžia išskleidžiamą meniu lauką „ZPKP algoritmas“.</p> <p>1.2. Sistema pateikia išfiltruotų algoritmų sąrašą, kurie tinkamai pasirinktai aplinkai.</p> <p>1.3. Naudotojas pasirenka algoritmą iš meniu.</p> <p>1.4. Sistema išsaugo algoritmo pasirinkimą duomenų bazėje.</p> <p>1.5. Sistema atvaizduoja pasirinkimą meniu lauke.</p> <p>Pilnas scenarijus pavaizduotas 26 paveiksle.</p>
Išimtis	<p>2. Išfiltruotų algoritmų sąrašas yra tuščias.</p> <p>2.1. Sistema pateikia tuščio sąrašo pranešimą.</p> <p>2.2. Nutraukiamas ZPKP algoritmo pasirinkimo procesas.</p>
Apribojimai:	
Salygos prieš	Užkrauta aplinka
Salygos po	Pakeistas ZPKP algoritmo pasirinkimas išsaugotas duomenų bazėje, atnaujinta vartotojo sasajos lauko reikšmė.



26 pav. Veiklos diagrama funkciniam reikalavimui „Pasirinkti ZPKP algoritmą“

3.6.1..1.0.2 Funkcinis reikalavimas „Keisti ZPKP algoritmo parametrus“

7 lentelė. Funkcinis reikalavimas „Keisti ZPKP algoritmo parametrus“

Trumpas aprašymas	Naudotojas gali keisti ZPKP algoritmo parametrus.
Scenarijai:	
Pagrindinis	1. Naudotojas keičia ZPKP algoritmo parametrus.
	1.1. Naudotojas paspaudžia ant algoritmo parametro lauko (R01). 1.2. Naudotojas suveda naują lauko reikšmę. 1.3. Sistema jrašo naują lauko reikšmę į duomenų bazę. Pilnas scenarijus pavaizduotas 27 paveiksle.
Išimtis	2. Įvesta netinkama lauko reikšmė.
	2.1. Įvesta naudotojo reikšmė pažeidžia apribojimus (R01). 2.2. Sistema pateikia klaidos pranešimą. 2.3. Sistema lauko reikšmė nustato į pradinę.
Apribojimai:	

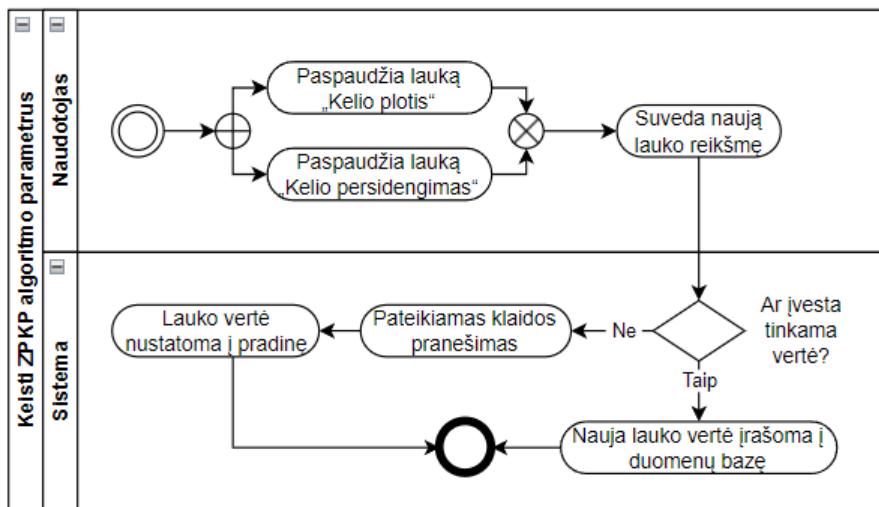
Sąlygos prieš	Pasirinktas ZPKP algoritmas.
Sąlygos po	Nauja parametru vertė įrašyta į duomenų bazę.
Reikalavimai:	
R01	ZPKP algoritmo parametrai

„Kelio plotis“:

- Nustato padengimo plotį, atitinkantį įrankio, sensoriaus darbinį plotį.
- Neneigiamą reikšmę.
- Reali reikšmę.
- Matavimo vienetai nustatomi pagal sisteminius parametrus.

„Kelio persidengimas“:

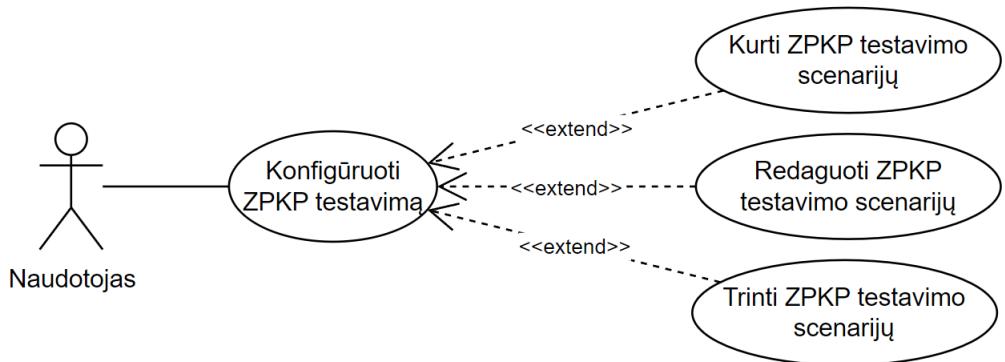
- Nustato persidengimo plotį, atitinkantį plotį persidengiantį tarp 2 gretimų padengimo kelių.
- Neneigiamą reikšmę.
- Reali reikšmę.
- Negali būti didesnė už „Kelio plotį“.
- Matavimo vienetai nustatomi pagal sisteminius parametrus.



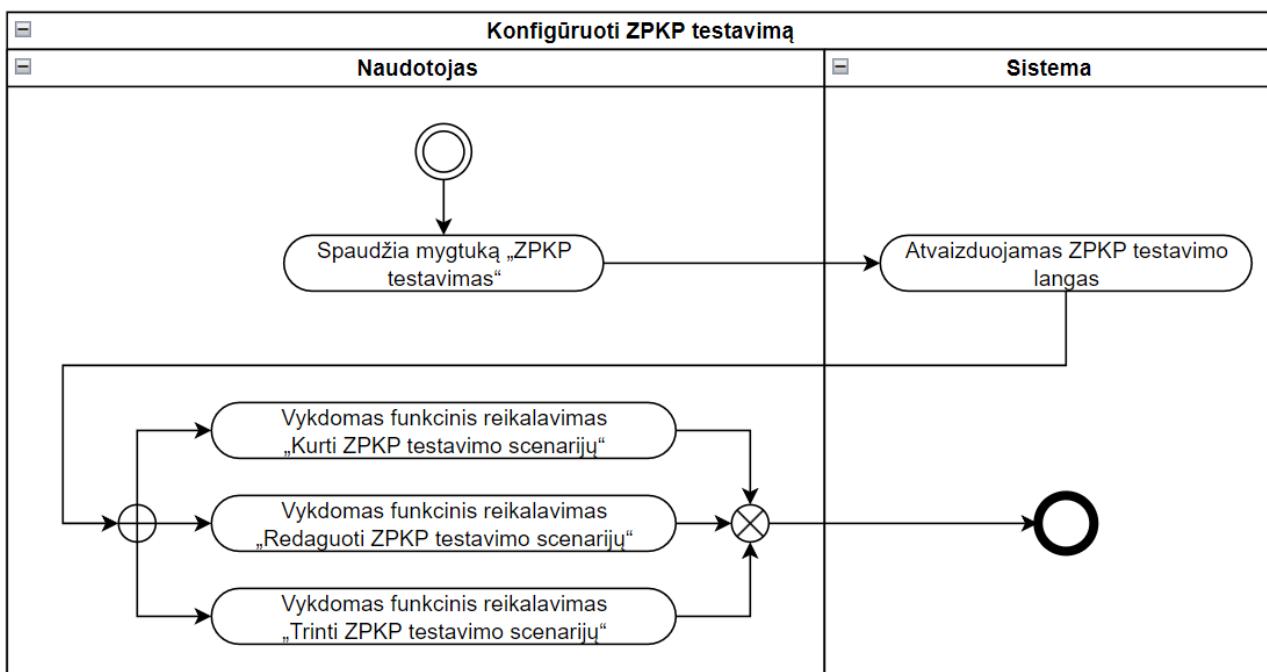
27 pav. Veiklos diagrama funkciniam reikalavimui „Keisti ZPKP algoritmo parametrus“

3.6.1.2 Funkcinis reikalavimas „Konfigūruoti ZPKP testavimą“

Konfigūruoti ZPKP testavimą (žr. 28 pav.) — tai veikla, kurios metu naudotojas gali kurti, redaguoti ir trinti ZPKP testavimo scenarijus.



28 pav. Funkcinis reikalavimas „Konfigūruoti ZPKP testavimą“



29 pav. Veiklos diagrama funkciniam reikalavimui „Konfigūruoti ZPKP testavimą“

3.6.1.2.0.1 Funkcinis reikalavimas „Kurti ZPKP testavimo scenarijų“

8 lentelė. Funkcinis reikalavimas „Kurti ZPKP testavimo scenarijų“

Trumpas aprašymas	Naudotojas gali kurti ZPKP testavimo scenarijus su išsaugotomis aplinkomis ir prieinamais algoritmais. Scenarijai leidžia daryti pa-kartotinius testavimus su keleta aplinkų ir / arba algoritmu.
Scenarijai:	
Pagrindinis	1. Naudotojas sukuria ZPKP testavimo scenarijų.

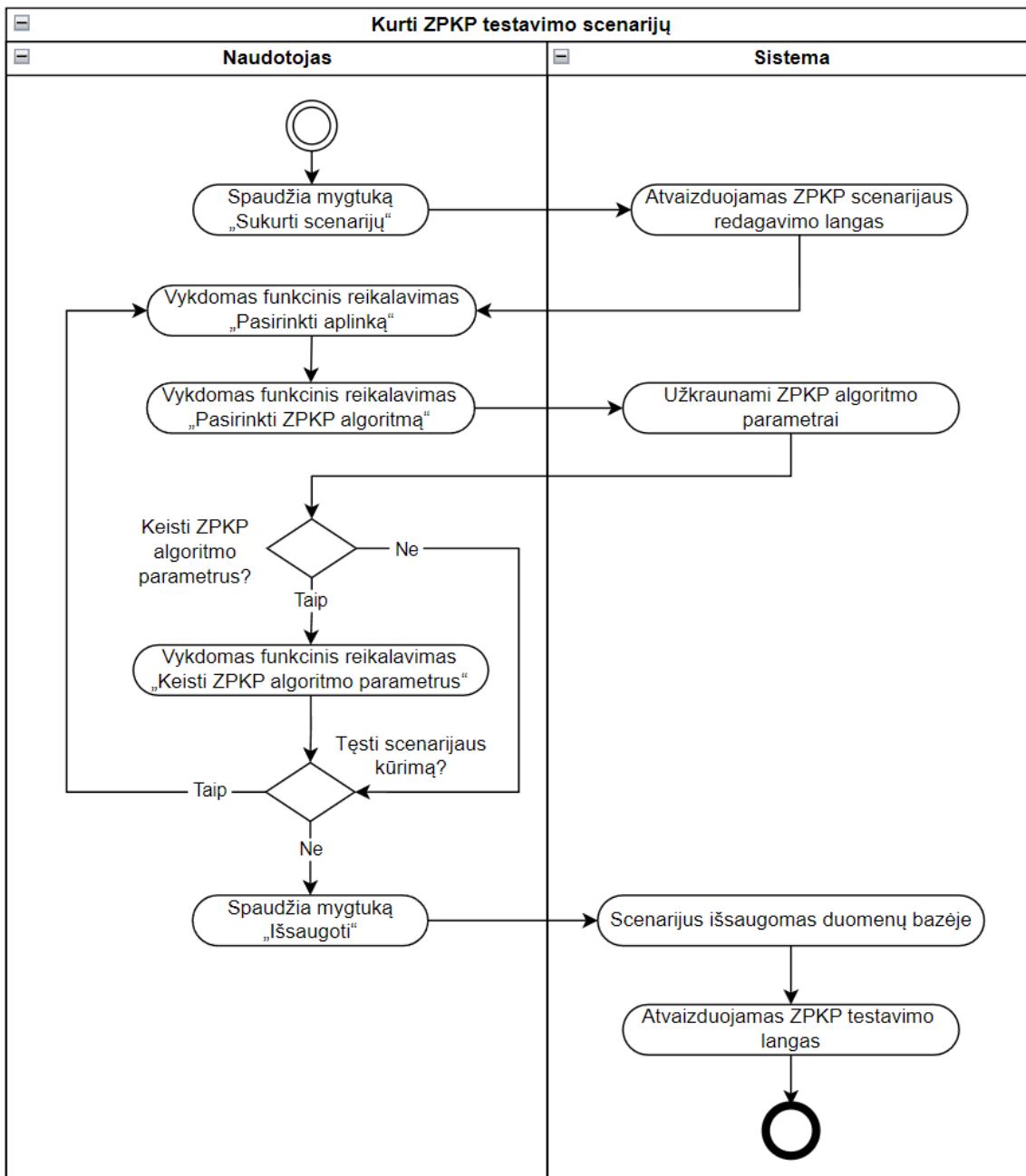
	<p>1.1. Naudotojas spaudžia mygtuką „Sukurti scenarijų“.</p> <p>1.2. Sistema atidaro ZPKP scenarijaus redagavimo langą (R02) su sukurtu nauju (priskiriamas unikalus ID) testavimo scenarijumi (R01).</p> <p>1.3. Naudotojas pasirenka išsaugotą aplinką, vykdomas funkcinis reikalavimas „Pasirinkti aplinką“.</p> <p>1.4. Naudotojas pasirenka algoritmą, kurį nori pritaikyti pasirinktai aplinkai, vykdomas funkcinis reikalavimas „Pasirinkti ZPKP algoritmą“.</p> <p>1.5. Sistema užkrauna algoritmo parametrus naudotus su pasirinkta aplinka.</p> <p>1.6. Jei reikalinga, naudotojas pakeičia algoritmo parametrus, vykdomas funkcinis reikalavimas „Keisti ZPKP algoritmo parametrus“.</p> <p>1.7. Kartojami 1.3 — 1.6 žingsniai, kol pilnai sukurtas scenarijus.</p> <p>1.8. Naudotojas spaudžia mygtuką „Išsaugoti“.</p> <p>1.9. Sistema išsaugo sukurtą testavimo scenarijų duomenų bazę.</p> <p>1.10. Sistema atvaizduoja ZPKP testavimo langą (R02).</p>
Pilnas scenarijus pavaizduotas paveiksle.	

Apribojimai:

Sąlygos prieš	Atidarytas testavimo scenarijų sąrašo langas (R03).
Sąlygos po	Sukurtas testavimo scenarijus, atidarytas testavimo langas paruoštas testavimo vykdymui.
Reikalavimai:	
R01	ZPKP testavimo scenarijus

	<p>Testavimo scenarijų sudaro:</p> <ul style="list-style-type: none"> • ID — unikalus testavimo scenarijaus identifikatorius. • Testavimo atvejai. <p>Testavimo atvejis:</p> <ul style="list-style-type: none"> • Scenarijaus ID — scenarijaus identifikatorius, kuriam priklauso atvejis. • Atvejo numeris — testavimo atvejo indikatorius, unikalus testavimo scenarijuje. • Aplinkos ID — išsaugotos aplinkos, naudojamos testavimo atvejyje, identifikatorius. • Algoritmo ID — algoritmo, naudojamo testavimo atvejyje, identifikatorius. • Algoritmo parametrai — parametrus galima keisti individualiai kiekvienam testavimo atvejui (žr. funkcinis reikalavimas „Keisti ZPKP algoritmo parametrus“ R01).
R02	ZPKP testavimo scenarijaus langas
	<ul style="list-style-type: none"> • Lange atvaizduojami ZPKP testavimo scenarijaus duomenys (R01). • Lango viršuje pateikiamas scenarijaus ID, pavadinimas. • Kiekvienas testavimo scenarijaus atvejis atvaizduojamas atskira eilute. • Lango viršuje pateikiami valdymo mygtukai: <ul style="list-style-type: none"> – Vykdymo mygtukas — vykdo testavimo scenarijų, po vykdymo pateikiami vykdymo rezultatai, ZPKP rezultatų lange. – Rezultatų peržiūros mygtukas — atidaro scenarijaus ZPKP rezultatų sąrašo langą. – Redagavimo mygtukas — jjungia / išjungia scenarijaus redagavimo režimą. – Trynimo mygtukas — ištrina testavimo scenarijų. • Redagavimo režimas, leidžia keisti: <ul style="list-style-type: none"> – Scenarijaus pavadinimą. – Scenarijaus atvejų eilučių duomenis.
R03	ZPKP testavimo scenarijų sąrašo langas

	<ul style="list-style-type: none"> • Lango viršuje pateikiamas valdymo mygtukas: – Naujo scenarijaus kūrimo mygtukas – sukuriamas naujas scenarijaus jrašas duomenų bazėje, atidaromas ZPKP testavimo scenarijaus langas, redagavimo režime R02. • Kiekvienas testavimo scenarijus atvaizduojamas atskira eilute. • Atvaizduojami duomenys eilutėje: <ul style="list-style-type: none"> – Scenarijaus ID. – Scenarijaus pavadinimas. • Kiekvienoje eilutėje pateikiami valdymo mygtukai: <ul style="list-style-type: none"> – Išskleidimo mygtukas – atvaizduojami / paslepiaomi visi scenarijaus atvejai. – Redagavimo mygtukas – atidaromas ZPKP testavimo scenarijaus langas, redagavimo režime. – Trynimo mygtukas – ištrina testavimo scenarijų.
--	---



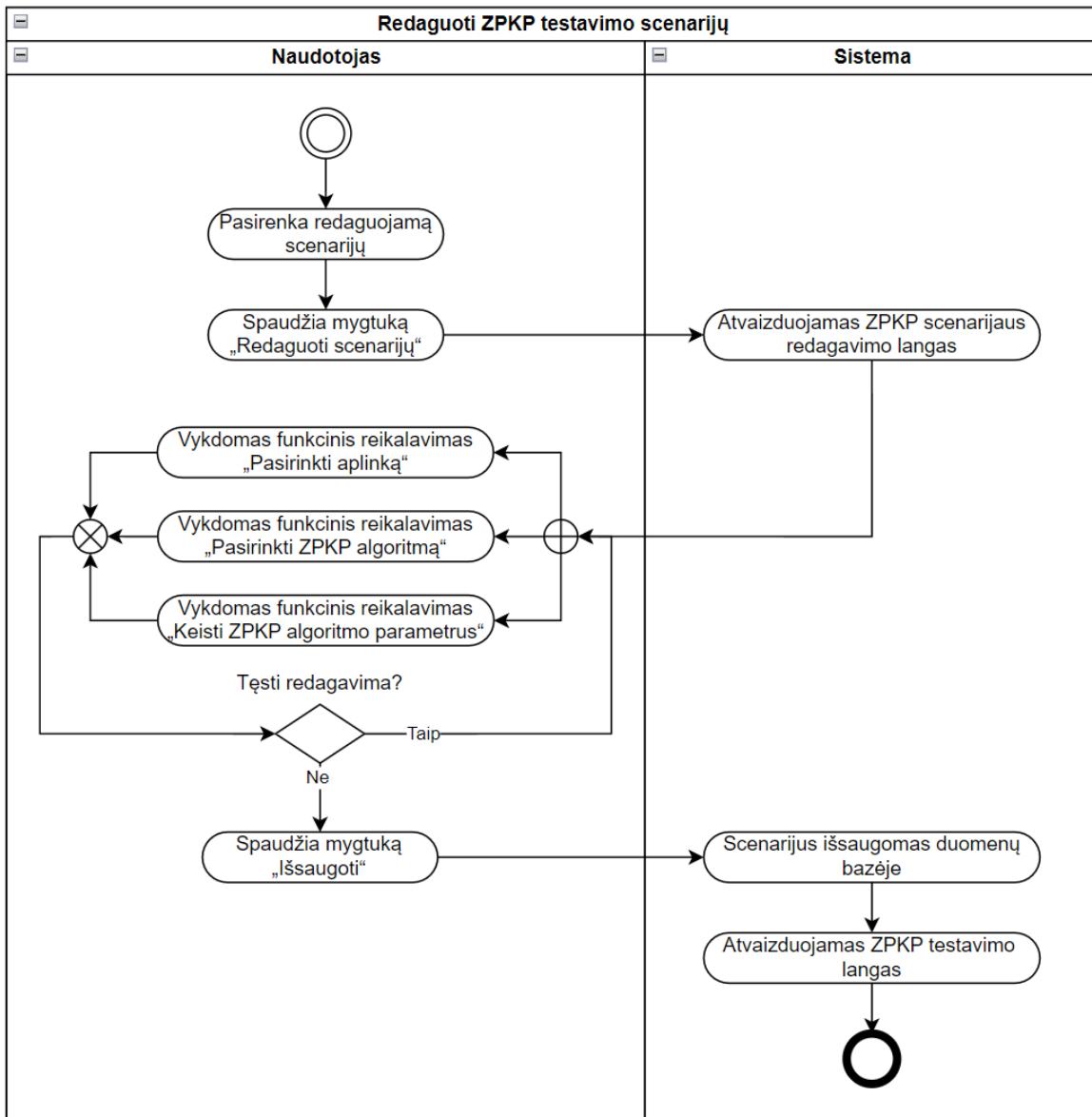
30 pav. Veiklos diagrama funkciniam reikalavimui „Kurti ZPKP testavimo scenarijų“

3.6.1.2.0.2 Funkcinis reikalavimas „Redaguoti ZPKP testavimo scenarijų“

9 lentelė. Funkcinis reikalavimas „Redaguoti ZPKP testavimo scenarijų“

Trumpas aprašymas	Naudotojas gali redaguoti sukurtą testavimo scenarijų.
Scenarijai:	
Pagrindinis	1. Naudotojas redagoja testavimo scenarijų.

	<p>1.1. Naudotojas pasirenka redaguojamą scenarijų, spaudžia mygtuką „Redaguoti scenarijų“.</p> <p>1.2. Sistema atidaro ZPKP scenarijaus redagavimo langą su užkrautu pasirinktu testavimo scenarijumi.</p> <p>1.3. (Neprivaloma) Naudotojas pasirenka išsaugotą aplinką, vykdomas funkcinis reikalavimas „Pasirinkti aplinką“.</p> <p>1.4. (Neprivaloma) Naudotojas pasirenka algoritmą, kurį nori prietaikyti pasirinktai aplinkai, vykdomas funkcinis reikalavimas „Pasirinkti ZPKP algoritmą“.</p> <p>1.5. (Neprivaloma) Naudotojas pakeičia algoritmo parametrus, vykdomas funkcinis reikalavimas „Keisti ZPKP algoritmo parametrus“.</p> <p>1.6. Kartojami 1.3 — 1.5 žingsniai, kol pilnai pakeistas scenarijus.</p> <p>1.7. Naudotojas spaudžia mygtuką „Išsaugoti“.</p> <p>1.8. Sistema išsaugo sukurtą testavimo scenarijų duomenų bazę.</p> <p>1.9. Sistema atvaizduoja ZPKP testavimo langą.</p> <p>Pilnas scenarijus pavaizduotas 31 paveiksle.</p>
Apribojimai:	
Sąlygos prieš	Atidarytas testavimo scenarijų sąrašo langas.
Sąlygos po	Modifikuotas testavimo scenarijus, atidarytas testavimo langas paruoštas testavimo vykdymui.



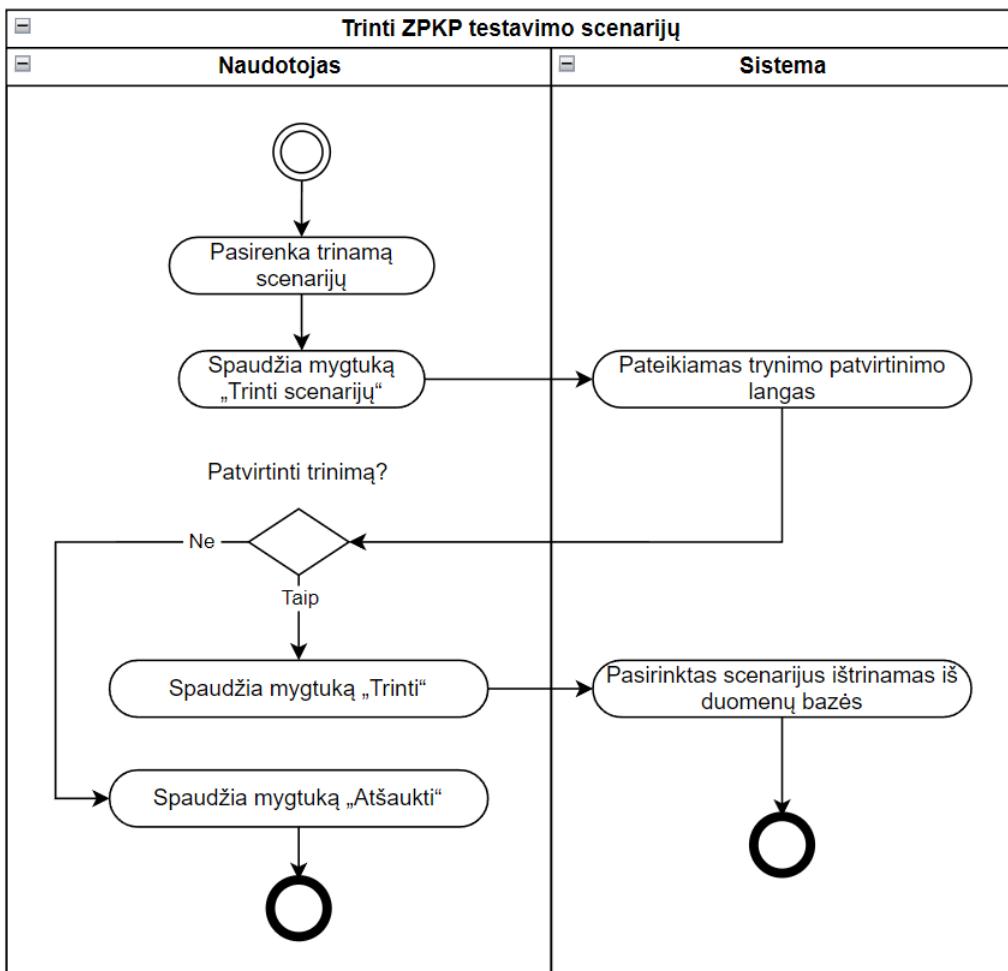
31 pav. Veiklos diagrama funkciniam reikalavimui „Redaguoti ZPKP testavimo scenarijų“

3.6.1..2.0.3 Funkcinis reikalavimas „Trinti ZPKP testavimo scenarijų“

10 lentelė. Funkcinis reikalavimas „Trinti ZPKP testavimo scenarijų“

Trumpas aprašymas	Naudotojas gali ištrinti pasirinktą testavimo scenarijų.
Scenarijai:	
Pagrindinis	1. Naudotojas ištrina testavimo scenarijų.

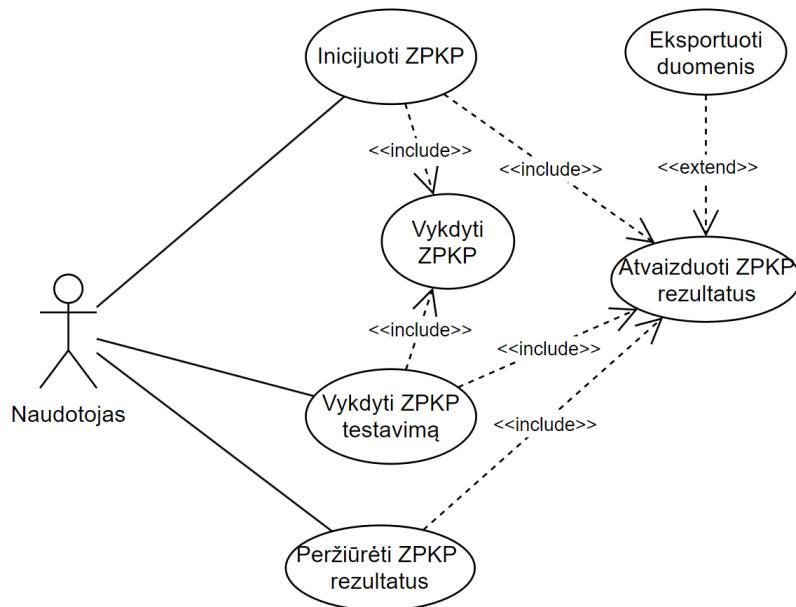
	<p>1.1. Naudotojas pasirenka trinamą scenarijų, spaudžia mygtuką „Trinti scenarijų“.</p> <p>1.2. Sistema pateikia trynimo patvirtinimo langą (R01).</p> <p>1.3. Naudotojas paspaudžia mygtuką „Trinti“.</p> <p>1.4. Sistema ištrina scenarijų iš duomenų bazės.</p> <p>Pilnas scenarijus pavaizduotas 32 paveiksle.</p>
Alternatyva	2. Naudotojas nutraukia trinimo procesą.
	<p>2.1. Vykdomi 1.1, 1.2 žingsniai.</p> <p>2.2. Naudotojas paspaudžia mygtuką „Atšaukti“.</p> <p>2.3. Sistema grįžta į pradinę būseną.</p>
Apribojimai:	
Sąlygos prieš	Atidarytas testavimo scenarijų sąrašo langas.
Sąlygos po	Ištrintas testavimo scenarijus, atidarytas testavimo scenarijų sąrašo langas.
Reikalavimai:	
R01	Trynimo patvirtinimo langas <ul style="list-style-type: none"> • Pateikiamas pranešimas „Ar tikrai norite pašalinti šį testavimo scenarijų?“ • Mygtukas „Trinti“ — pasirinktas scenarijus ištrinamas iš duomenų bazės. • Mygtukas „Atšaukti“ — nutraukiama trynimo procesas.



32 pav. Veiklos diagrama funkciniam reikalavimui „Trinti ZPKP testavimo scenarijų“

3.6.2. ZPKP funkcinių reikalavimų grupė

Tyrimo procese, po aplinkos ir algoritmo sukonfigūravimo, ZPKP funkcinių reikalavimų grupė atsakinga už proceso tęsimą, paruoštus duomenis siunčiant išorinei sistemai, kuri atlieka priskirtus skaičiavimus. Gavus rezultatus jie išsaugomi duomenų bazėje ir atvaizduojami naudotojui, kuris turi galimybę gautus rezultatus eksportuoti į išorinį failą.



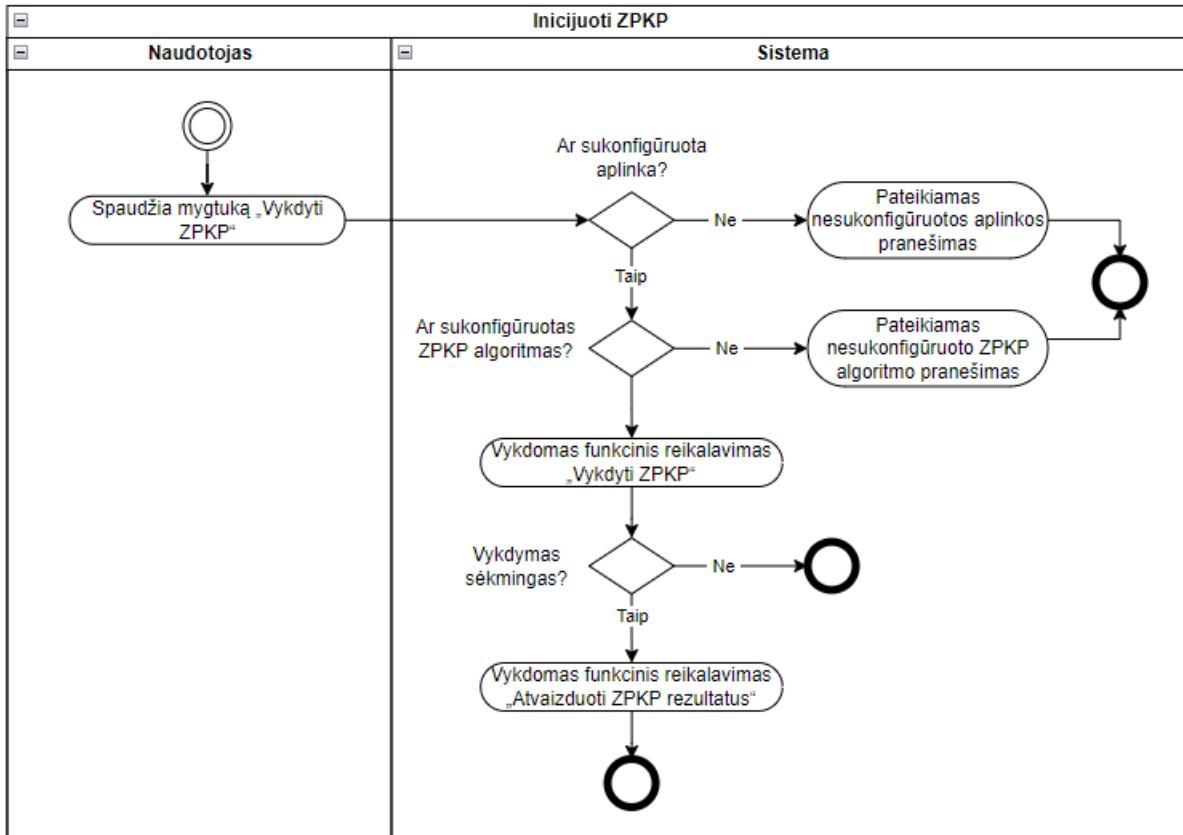
33 pav. ZPKP funkcinių reikalavimų grupė

3.6.2..1 Funkcinis reikalavimas „Inicijuoti ZPKP“

11 lentelė. Funkcinis reikalavimas „Inicijuoti ZPKP“

Trumpas aprašymas	Naudotojas gali inicijuoti, vykdyti ZPKP algoritmą. Tai tiesioginis būdas gauti algoritmo rezultatus aktyviai (darbalaukyje užkrautai) aplinkai ir pasirinktais algoritmo konfigūracijai.
Scenarijai:	
Pagrindinis	<p>1. Naudotojas inicijuoja ZPKP algoritmą.</p> <p>1.1. Naudotojas spausdina mygtuką „Vykduti ZPKP“.</p> <p>1.2. Vydomas funkcinis reikalavimas „Vykduti ZPKP“.</p> <p>1.3. Vydomas funkcinis reikalavimas „Atvaizduoti ZPKP rezultatus“.</p> <p>Pilnas scenarijus pavaizduotas 34 paveiksle.</p>
Išimtis	<p>2. Nesukonfigūruoti duomenys.</p> <p>2.1. Sistema pateikia nesukonfigūruotų duomenų pranešimą (R01).</p> <p>2.2. Sistema grįžta į pradinę būseną.</p>
Apribojimai:	
Sąlygos prieš	Sukonfigūruota aplinka ir algoritmas.
Sąlygos po	Atvaizduoti ZPKP rezultatai.
Reikalavimai:	
R01	Nesukonfigūruotų duomenų pranešimai.

	<p>Pranešimai pateikiami, jeigu nesukonfigūruota:</p> <ul style="list-style-type: none"> • Aplinka. • Algoritmas.
--	---



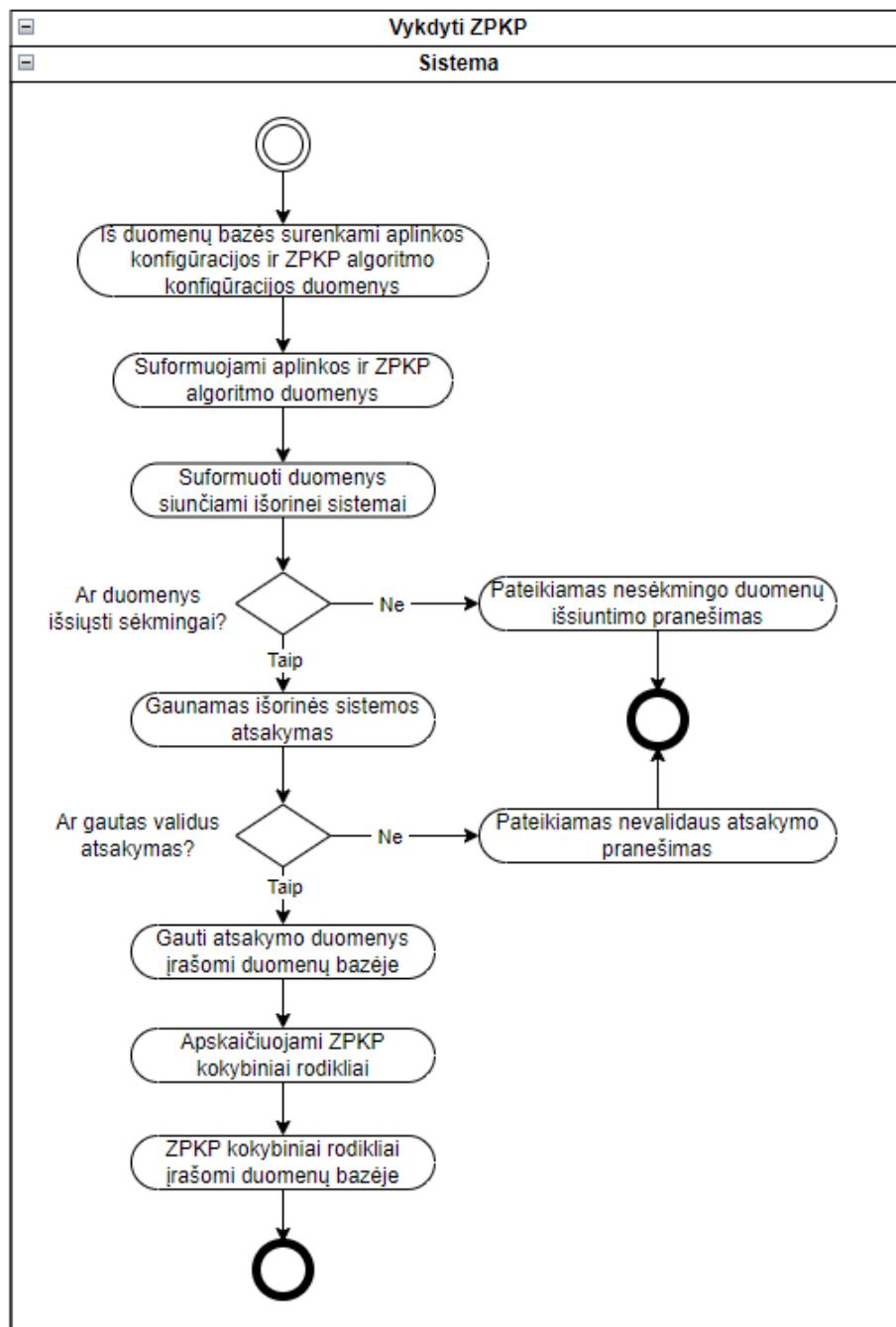
34 pav. Veiklos diagrama funkciniam reikalavimui „Inicijuoti ZPKP“

3.6.2.2 Funkcinis reikalavimas „Vykduti ZPKP“

12 lentelė. Funkcinis reikalavimas „Vykduti ZPKP“

Trumpas aprašymas	Sistema gali siųsti vidinius duomenis į išorines sistemas, kurios atlieka ZPKP algoritmų vykdymą.
Scenarijai:	
Pagrindinis	1. Sistema vykdo ZPKP.

	<p>1.1. Sistema surenka aplinkos ir algoritmo duomenis.</p> <p>1.2. Sistema suformatuoja surinktus duomenis (R01).</p> <p>1.3. Sistema išsiunčia duomenis išorinei sistemai.</p> <p>1.4. Sistema gauna atsakymą iš išorinės sistemos.</p> <p>1.5. Sistema gautus atsakymo duomenis (R02) įrašo į duomenų bazę.</p> <p>1.6. Sistema apskaičiuoja ZPKP kokybinius rodiklius (R03).</p> <p>1.7. Sistema įrašo kokybinius rodiklius į duomenų bazę.</p> <p>Pilnas scenarijus pavaizduotas 35 paveiksle.</p>
Išimtis	2. Nesėkmingas duomenų perdavimas.
	<p>2.1. Vykdant duomenų siuntimą, apdorojant atsakymą gaunama klaida.</p> <p>2.2. Sistema pateikia klaidos pranešimą (R04).</p>
Apribojimai:	
Salygos prieš	Sukonfigūruota aplinka ir algoritmas.
Salygos po	ZPKP rezultatai išsaugoti duomenų bazėje.
Reikalavimai:	
R01	Aplinkos duomenys
	<p>Duomenys suformatuoti JSON formatu:</p> <ul style="list-style-type: none"> • ZPKP algoritmo duomenys. • Aplinkos duomenys.
R02	Atsakymo duomenys
	<p>Gaunamas atsakymas JSON formatu:</p> <ul style="list-style-type: none"> • Maršruto taškai. • Tarpiniai algoritmo skaičiavimai.
R03	Kokybiniai rodikliai
	<p>Kokybinius rodiklius sudaro:</p> <ul style="list-style-type: none"> • Zonos padengimas — matuojama santykine dalimi, kokia dalis zonos buvo padengta. • Maršruto persidengimas — matuojamas santykine dalimi, kokia dalis maršruto persidengia. • Posūkių skaičius — skaičius, kiek kartų pakeičiama judėjimo kryptis.
R04	Klaidos pranešimas
	<p>Naudotojui pateikiami klaidos pranešimai:</p> <ul style="list-style-type: none"> • Nesėkmingo duomenų išsiuntimo. • Nevalidaus atsakymo gavimo.



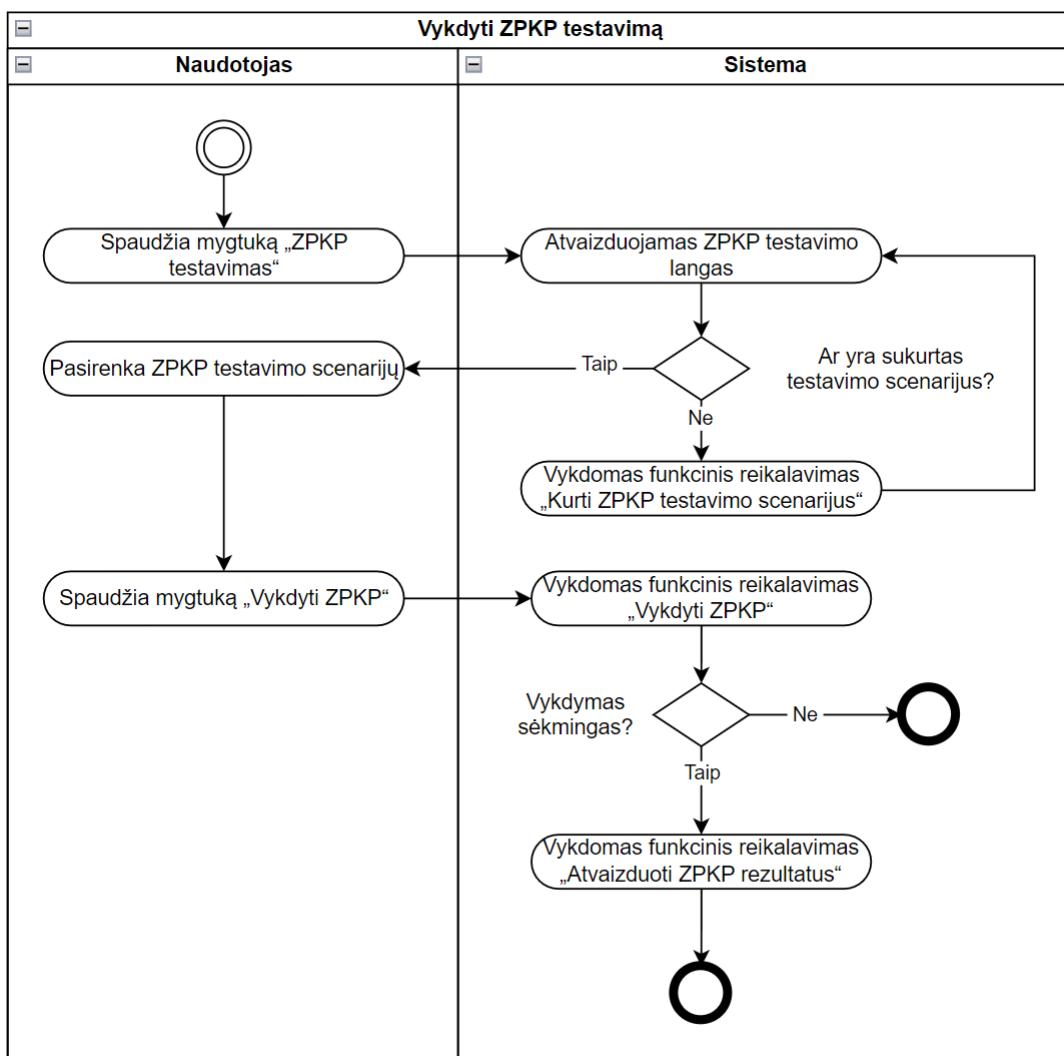
35 pav. Veiklos diagrama funkciniam reikalavimui „Vykdyti ZPKP”

3.6.2.3 Funkcinis reikalavimas „Vykdyti ZPKP testavimą“

13 lentelė. Funkcinis reikalavimas „Vykdyti ZPKP testavimą“

Trumpas aprašymas	Naudotojas gali iniciuoti, vykdyti ZPKP testavimo scenarijų, krejiantis į išorinę sistemą kiekvienam testavimo atvejui ir surenkant gautus duomenis.
Scenarijai:	
Pagrindinis	1. Naudotojas vykdo ZPKP testavimą.

	<p>1.1. Naudotojas spaudžia mygtuką „ZPKP testavimas“.</p> <p>1.2. Sistema atidaro ZPKP testavimo scenarijų sąrašo langą.</p> <p>1.3. Naudotojas pasirenka ZPKP testavimo scenarijų.</p> <p>1.4. Sistema atvaizduoja ZPKP testavimo langą.</p> <p>1.5. Naudotojas paspaudžia mygtuką „Vykdyti ZPKP“.</p> <p>1.6. Kiekvienam testavimo atvejui vykdomas funkcinis reikalavimas „Vykdyti ZPKP“.</p> <p>1.7. Vykdomas funkcinis reikalavimas „Atvaizduoti ZPKP rezultatus“.</p> <p>Pilnas scenarijus pavaizduotas 36 paveiksle.</p>
Apribojimai:	
Sąlygos prieš	Sukurtas ZPKP testavimo scenarijus.
Sąlygos po	Atvaizduoti ZPKP testavimo rezultatai.

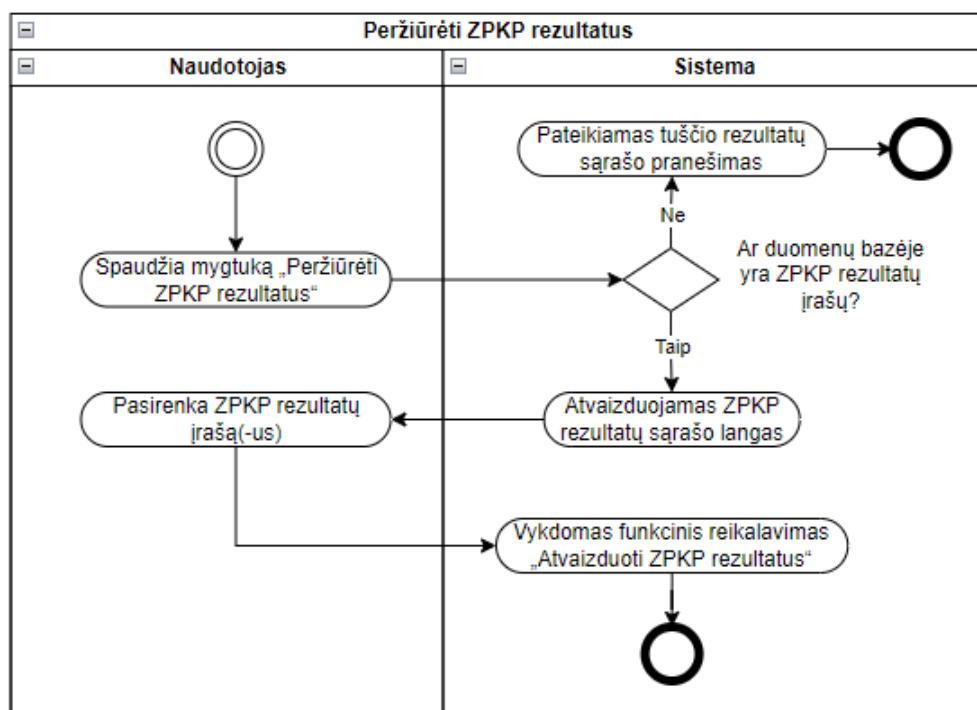


36 pav. Veiklos diagrama funkciniam reikalavimui „Vykdyti ZPKP testavimą”

3.6.2.4 Funkcinis reikalavimas „Peržiūrėti ZPKP rezultatus“

14 lentelė. Funkcinis reikalavimas „Peržiūrėti ZPKP rezultatus“

Trumpas aprašymas	Naudotojas gali peržiūrėti išsaugotus ZPKP rezultatus.
Scenarijai:	
Pagrindinis	1. Naudotojas peržiūri ZPKP rezultatus.
	<p>1.1. Naudotojas spaudžia mygtuką „Peržiūrėti ZPKP rezultatus“.</p> <p>1.2. Sistema atvaizduoja ZPKP rezultatų sąrašo langą.</p> <p>1.3. Naudotojas pasirenka ZPKP rezultatų įrašą(—us).</p> <p>1.4. Vykdomas funkcinis reikalavimas „Atvaizduoti ZPKP rezultatus“</p> <p>Pilnas scenarijus pavaizduotas 37 paveiksle.</p>
Apribojimai:	
Sąlygos prieš	Sistemoje yra išsaugotų ZPKP vykdymo rezultatų.
Sąlygos po	Atvaizduoti pasirinkti ZPKP vykdymo įrašo rezultatai.

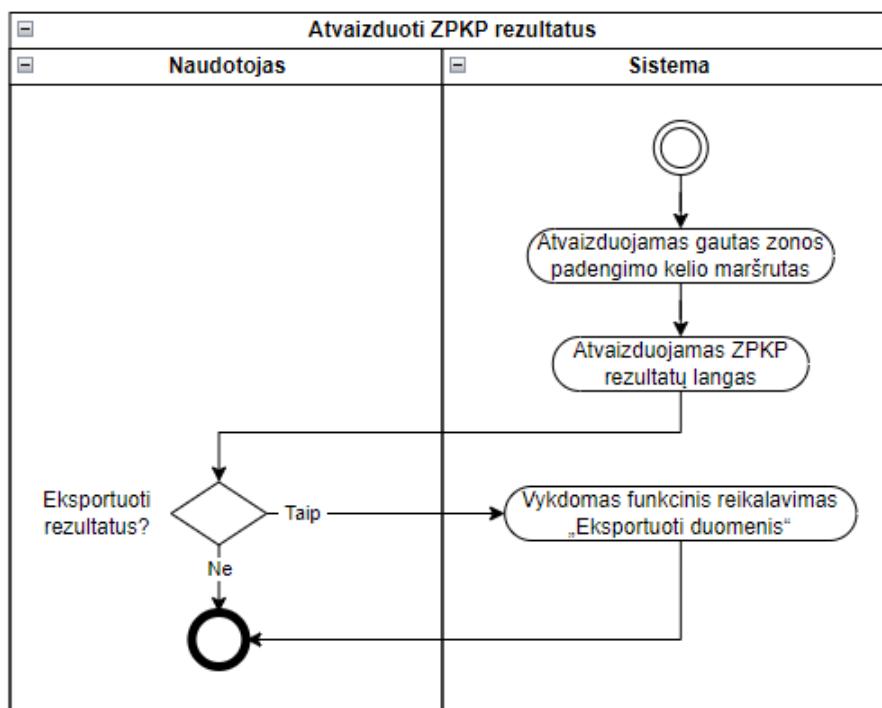


37 pav. Veiklos diagrama funkciniam reikalavimui „Peržiūrėti ZPKP rezultatus“

3.6.2.5 Funkcinis reikalavimas „Atvaizduoti ZPKP rezultatus“

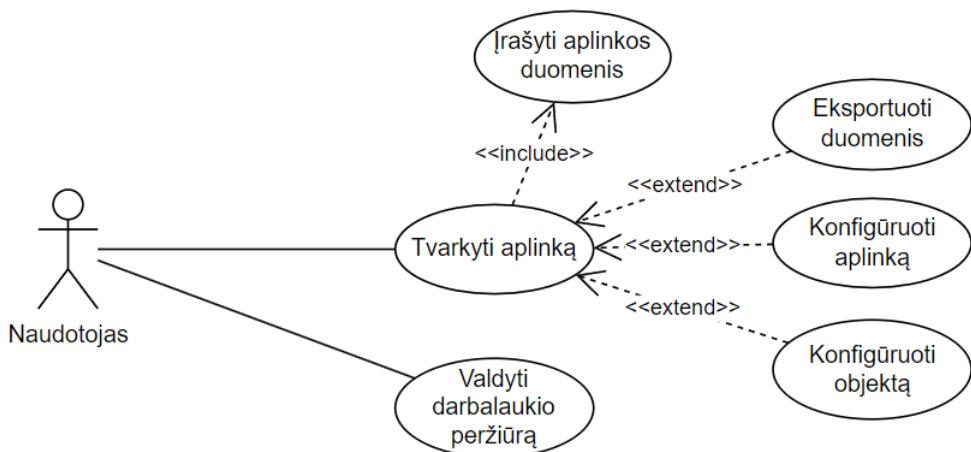
15 lentelė. Funkcinis reikalavimas „Atvaizduoti ZPKP rezultatus“

Trumpas aprašymas	Sistema gali pateikti ZPKP vykdymo rezultatus.
Scenarijai:	
Pagrindinis	1. Sistema pateikia ZPKP rezultatus.
	1.1. Sistema atvaizduoja gautos zonas padengimo maršrutą. 1.2. Sistema atvaizduoja ZPKP rezultatų langą. Pilnas scenarijus pavaizduotas 38 paveiksle.
Alternatyva	2. Naudotojas nori eksportuoti rezultatus.
	2.1. Vykdomi 1.1, 1.2 žingsniai. 2.2. Vykdomas funkcinis reikalavimas „Eksportuoti duomenis“.
Apribojimai:	
Sąlygos prieš	Sistemoje yra išsaugotų ZPKP vykdymo rezultatų.
Sąlygos po	Atvaizduoti pasirinkti ZPKP vykdymo įrašo rezultatai.



38 pav. Veiklos diagrama funkciniam reikalavimui „Atvaizduoti ZPKP rezultatus“

3.6.3. Aplinkos valdymo funkciniai reikalavimai



39 pav. Aplinkos valdymo funkciniai reikalavimai

3.6.3.1 Funkcinis reikalavimas „Įrašyti aplinkos duomenis“

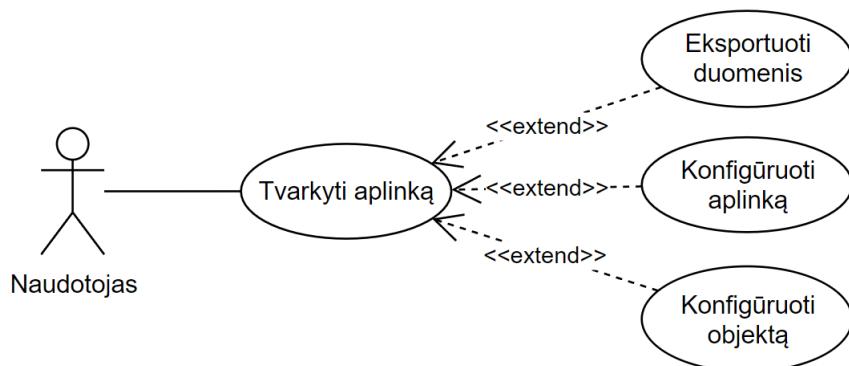
16 lentelė. Funkcinis reikalavimas „Įrašyti aplinkos duomenis“

Trumpas aprašymas	Sistema turi įrašyti aplinkos duomenis į duomenų bazę, atlikus aplinkos pakeitimus.
Scenarijai:	
Pagrindinis	<p>1. Sistema įrašo aplinkos duomenis.</p> <p>1.1. Sistema surenka aplinkos duomenis (R01) iš sistemos. 1.2. Sistema išsaugo duomenis duomenų bazėje.</p>
Išimtis	Įrašant duomenis įvyksta klaida.
	<p>2.1. Sistema pateikia naudotojui klaidos pranešimą.</p>
Apribojimai:	
Sąlygos prieš	Pasikeitusi aplinkos būsena.
Sąlygos po	Aplinkos būsena išsaugota duomenų bazėje.
Reikalavimai:	
R01	Aplinkos duomenys

	<p>Aplinkos duomenis sudaro:</p> <ul style="list-style-type: none"> • ID — unikalus aplinkos identifikacinis numeris. • Pavadinimas — naudojama naudotojo sasajoje ir eksportuojamuo failo pavadinime. • Formatas — daugiakampiai (angl. polygonal) arba tinkleliu gristu (angl. grid-based) formatas. • Tipas — žinomas (angl. off-line), nežinomas (angl. on-line), mišrus. • Globalus tipas — taip arba ne. Taip — objektų tipai ignoruojami, laikomasi bendro aplinkos tipo. Ne — galioja individualūs objektų tipai, jeigu objektui nepriskirtas individualus tipas, objektui priskiriamas aplinkos tipas. • Zonos objektų skaičius. • Zonos objektų duomenys (funkcinis reikalavimas „Sukurti naują objekta“, R02). • Kliūčių objektų skaičius. • Kliūčių objektų duomenys (funkcinis reikalavimas „Sukurti naują objekta“, R02).
--	--

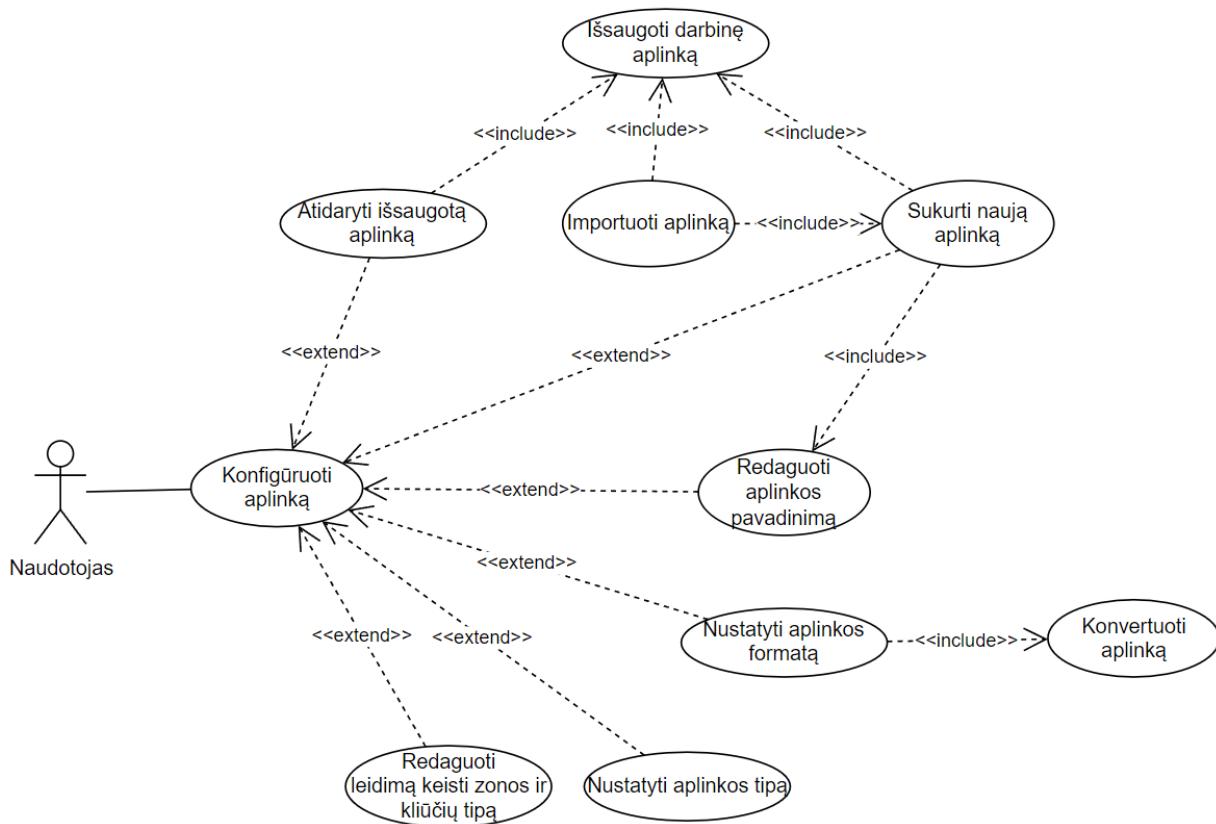
3.6.3.2 Funkcinis reikalavimas „Tvarkyti aplinką“

Aplinka susideda iš objektų (zonų ir kliūčių), todėl jie yra aplinkos dalis. „Tvarkyti aplinką“ grupė skirta valdyti bendrus aplinkos parametrus ir individualius objektų parametrus. Naudotojas sukonfigūruotą aplinką (kartu su visais objektais) gali eksportuoti į išorinį failą.



40 pav. Funkcinis reikalavimas „Tvarkyti aplinką“

3.6.3..2.1 Funkcinis reikalavimas „Konfigūruoti aplinką“



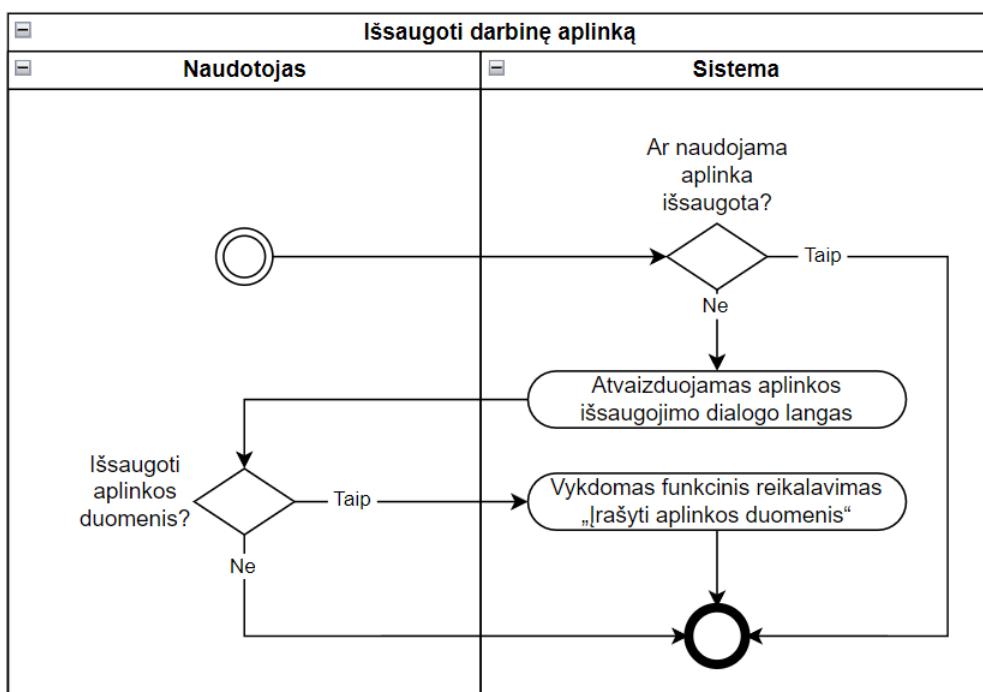
41 pav. Funkcinis reikalavimas „Konfigūruoti aplinką“

3.6.3..2.1.1 Funkcinis reikalavimas „Išsaugoti darbinę aplinką“

17 lentelė. Funkcinis reikalavimas „Išsaugoti darbinę aplinką“

Trumpas aprašymas	Prieš keičiant aplinkas, patikrinama ar dabartinė aplinka, su kuria dirbama, yra išsaugota.
Scenarijai:	
Pagrindinis	<p>1. Naudotojas išsaugo darbinę aplinką.</p> <p>1.1. Sistema atvaizduoja aplinkos išsaugojimo dialogo langą R01. 1.2. Naudotojas pasirenka išsaugoti darbinę aplinką. 1.3. Vykdomas funkcinis reikalavimas „Įrašyti aplinkos duomenis“.</p> <p>Pilnas scenarijus pavaizduotas 42 paveiksle.</p>
Alternatyva	<p>2. Naudotojas neišsaugo darbinės aplinkos.</p> <p>2.1. Vykdomas 1.1 žingsnis. 2.2. Naudotojas pasirenka neišsaugoti darbinės aplinkos. 2.3. Išsaugojimo procesas nutraukiamas.</p>

Išimtis	3. Darbinės aplinka išsaugota.
	3.1. Išsaugojimo procesas nutraukiamas.
Apribojimai:	
Sąlygos prieš	Atidaroma kita aplinka.
Sąlygos po	Darbinė aplinka išsaugota duomenų bazėje.
R01	Aplinkos išsaugojimo dialogo langas
	<ul style="list-style-type: none"> Pateikiamas pranešimas „Išsaugoti aplinką?“ Mygtukas „Taip“ — aplinka išsaugoma duomenų bazėje. Mygtukas „Atšaukti“ — nutraukiamas išsaugojimo procesas.



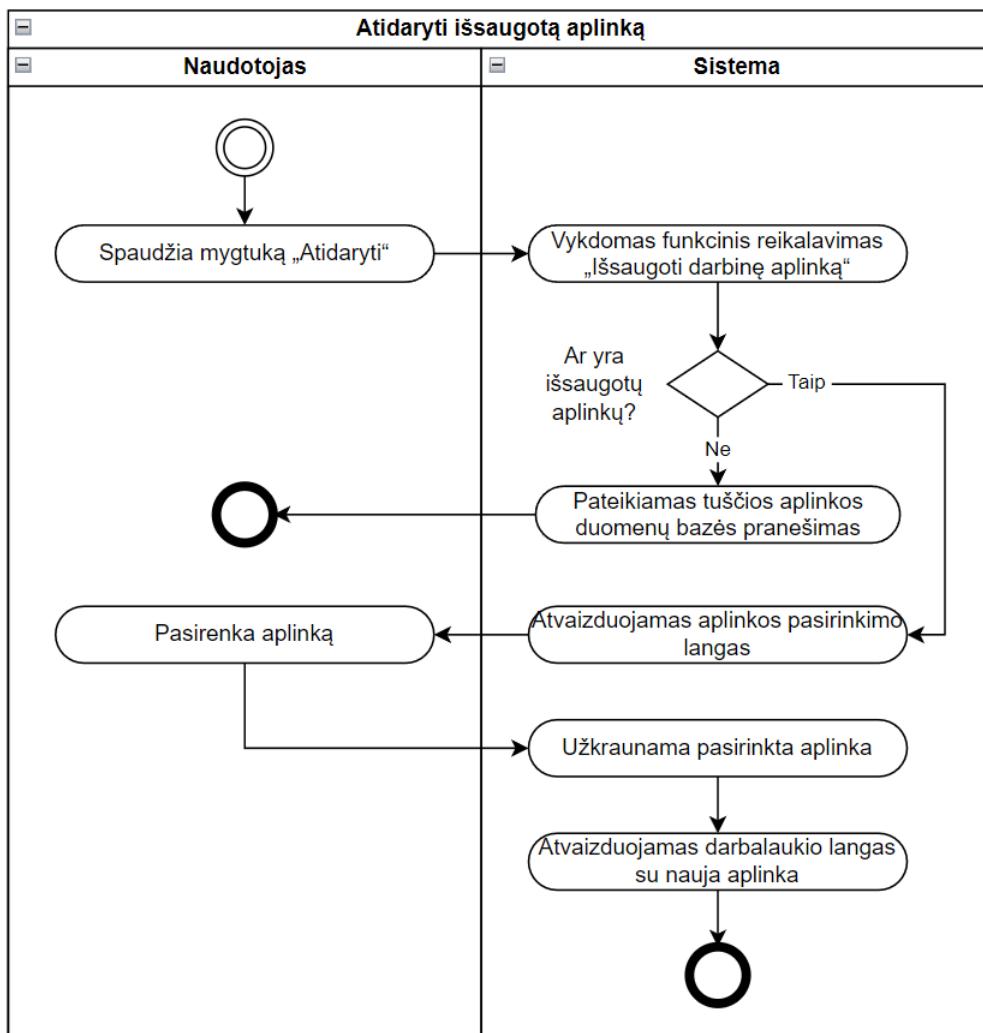
42 pav. Veiklos diagrama funkciniam reikalavimui „Išsaugoti darbinę aplinką“

3.6.3..2.1.2 Funkcinis reikalavimas „Atidaryti išsaugotą aplinką“

18 lentelė. Funkcinis reikalavimas „Atidaryti išsaugotą aplinką“

Trumpas aprašymas	Naudotojas gali atidaryti pasirinktą aplinką, išsaugotą duomenų bazėje.
Scenarijai:	
Pagrindinis	1. Naudotojas atidaro pasirinktą aplinką.

	<p>1.1. Naudotojas spaudžia skirtuko „Failas“ mygtuką „Atidaryti“.</p> <p>1.2. Vykdomas funkcinis reikalavimas „Išsaugoti darbinę aplinką“.</p> <p>1.3. Sistema atvaizduoja aplinkos pasirinkimo langą.</p> <p>1.4. Naudotojas pasirenka norimą aplinką.</p> <p>1.5. Sistema užkrauna pasirinktą aplinką.</p> <p>1.6. Sistema atnaujina darbalaukio vaizdą.</p> <p>Pilnas scenarijus pavaizduotas 43 paveiksle.</p>
Išimtis	2. Nėra išsaugotų aplinkų.
	<p>2.1. Sistema pateikia tuščios aplinkos duomenų bazės pranešimą.</p> <p>2.2. Sistema grįžta į pradinę būseną.</p>
Apribojimai:	
Salygos prieš	Atidaryta sistema.
Salygos po	Užkrauta ir darbalaukyje atvaizduota pasirinkta aplinka.



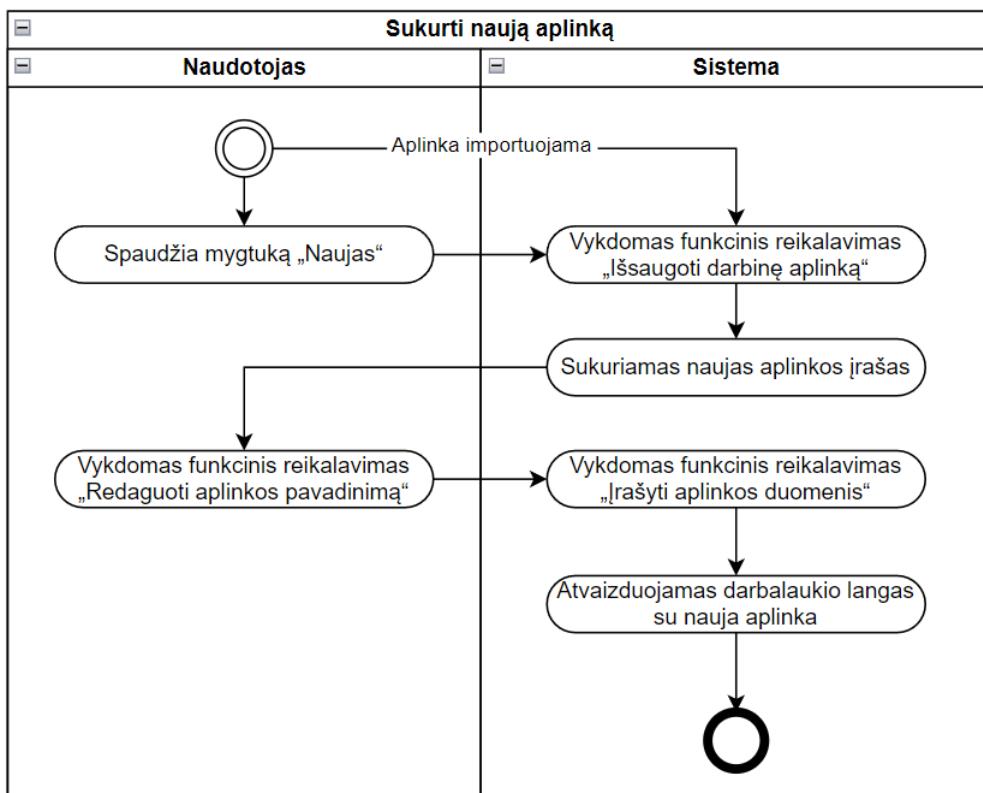
43 pav. Veiklos diagrama funkciniam reikalavimui „Atidaryti išsaugotą aplinką“

3.6.3..2.1.3 Funkcinis reikalavimas „Sukurti naują aplinką“

19 lentelė. Funkcinis reikalavimas „Sukurti naują aplinką“

Trumpas aprašymas	Naudotojas gali sukurti naują aplinką.
Scenarijai:	
Pagrindinis	1. Naudotojas sukuria naują aplinką.

	<p>1.1. Naudotojas paspaudžia skirtuko „Failas“ mygtuką „Naujas“.</p> <p>1.2. Vykdomas funkcinis reikalavimas „Išsaugoti darbinę aplinką“.</p> <p>1.3. Sistema sukuria naują aplinkos įrašą.</p> <p>1.4. Vykdomas funkcinis reikalavimas „Redaguoti aplinkos pavadinimą“.</p> <p>1.5. Vykdomas funkcinis reikalavimas „Įrašyti aplinkos duomenis“.</p> <p>1.6. Sistema darbalaukyje atvaizduoja naują aplinką.</p> <p>Pilnas scenarijus pavaizduotas 44 paveiksle.</p>
Alternatyva	2. Aplinka importuojama.
	<p>2.1. Vykdomas 1.3 žingsnis.</p> <p>2.2. Suformatuojamas aplinkos įrašas iš išorinių duomenų.</p> <p>2.3. Vykdomi 1.5, 1.6 žingsniai.</p>
Apribojimai:	
Sąlygos prieš	Atidaryta sistema.
Sąlygos po	Užkrauta ir darbalaukyje atvaizduota nauja aplinka.

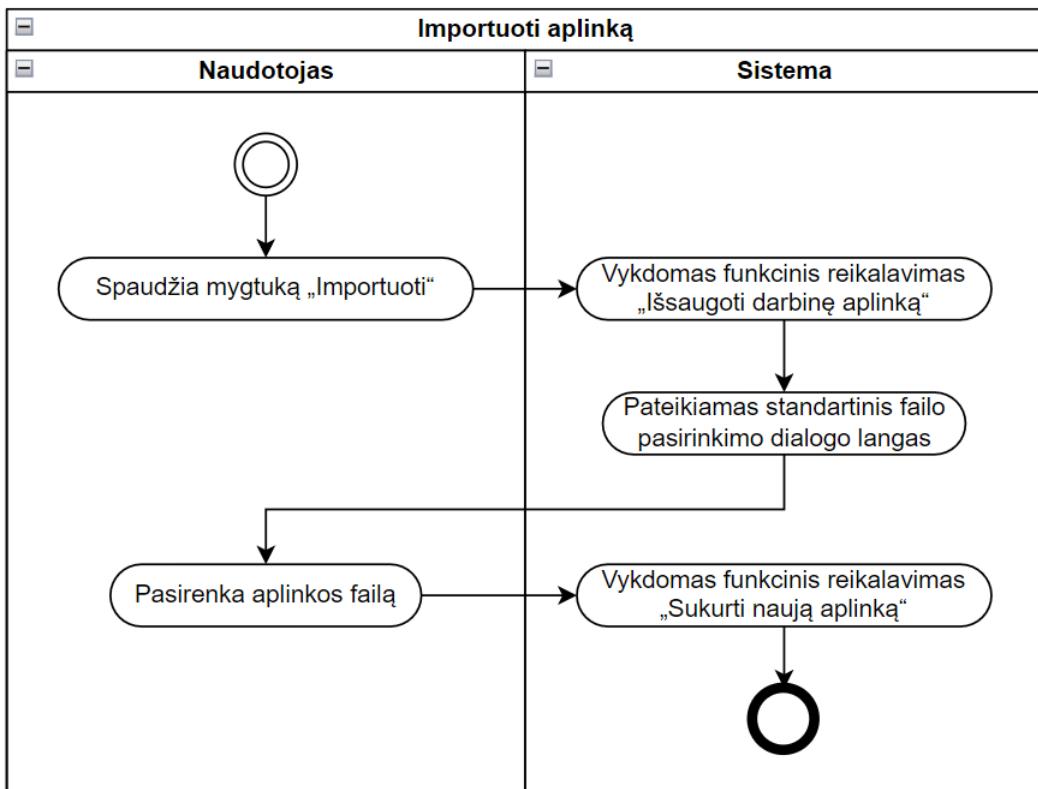


44 pav. Veiklos diagrama funkciniam reikalavimui „Sukurti naują aplinką“

3.6.3..2.1.4 Funkcinis reikalavimas „Importuoti aplinką“

20 lentelė. Funkcinis reikalavimas „Importuoti aplinką“

Trumpas aprašymas	Naudotojas gali importuoti aplinką iš išorinio failo.
Scenarijai:	
Pagrindinis	<p>1. Naudotojas importuoja aplinką.</p> <p>1.1. Naudotojas paspaudžia skirtuko „Failas“ mygtuką „Importuoti“.</p> <p>1.2. Vykdomas funkcinis reikalavimas „Išsaugoti darbinę aplinką“.</p> <p>1.3. Sistema pateikia standartinį failo pasirinkimo dialogo langą.</p> <p>1.4. Naudotojas pasirenka failą (R01).</p> <p>1.5. Vykdomas funkcinis reikalavimas „Sukurti naują aplinką“, alternatyva „Aplinka importuojama“.</p> <p>Pilnas scenarijus pavaizduotas 45 paveiksle.</p>
Apribojimai:	
Sąlygos prieš	Atidaryta sistema.
Sąlygos po	Užkrauta ir darbalaukyje atvaizduota importuota aplinka.
Reikalavimai:	
R01	<p>Importuojamas failas</p> <p>Importuoamo failo duomenys:</p> <ul style="list-style-type: none"> • Žr. funkcinis reikalavimas „Įrašyti aplinkos duomenis“, R01.



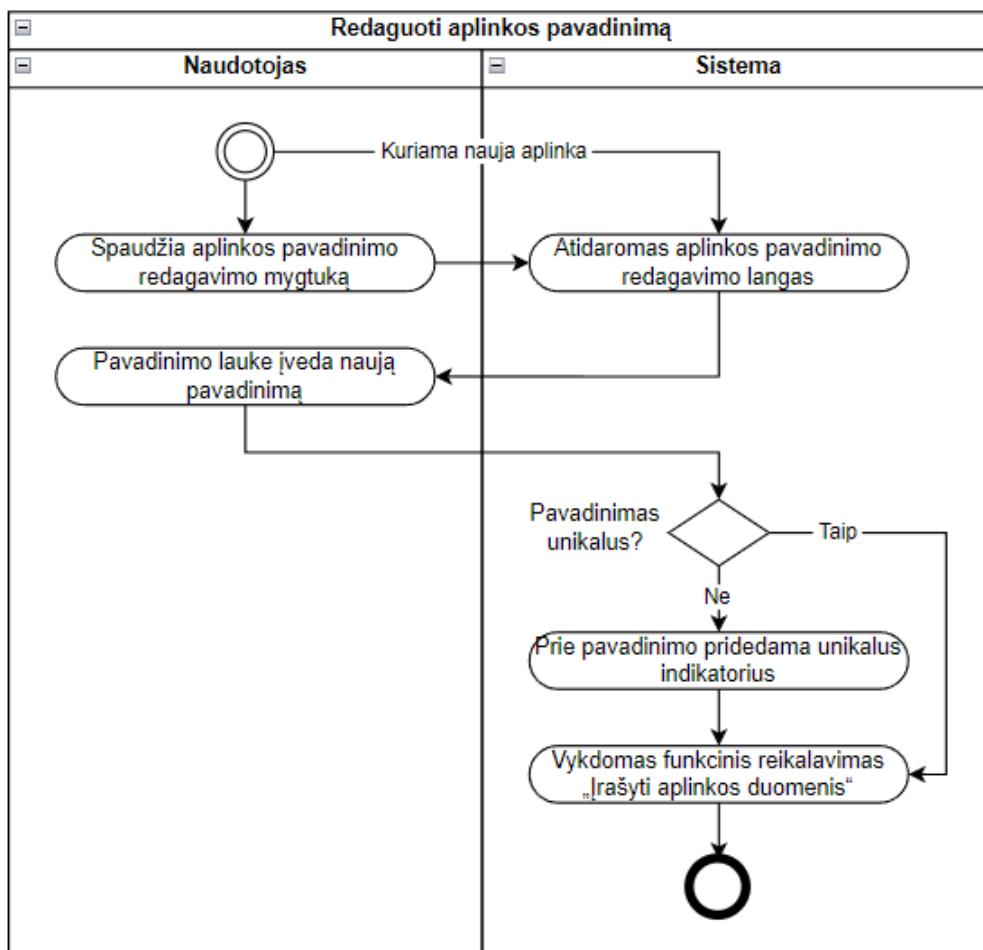
45 pav. Veiklos diagrama funkciniam reikalavimui „Importuoti aplinką“

3.6.3.2.1.5 Funkcinis reikalavimas „Redaguoti aplinkos pavadinimą“

21 lentelė. Funkcinis reikalavimas „Redaguoti aplinkos pavadinimą“

Trumpas aprašymas	Naudotojas gali redaguoti aplinkos pavadinimą.
Scenarijai:	
Pagrindinis	1. Naudotojas redaguoja aplinkos pavadinimą. <ul style="list-style-type: none"> 1.1. Naudotojas spausdžia aplinkos pavadinimo redagavimo mygtuką. 1.2. Sistema atidaro aplinkos pavadinimo redagavimo langą. 1.3. Naudotojas pavadinimo lauke įveda naują pavadinimą. 1.4. Vykdomas funkcinis reikalavimas „Įrašyti aplinkos duomenis“. Pilnas scenarijus pavaizduotas 46 paveiksle.
Išimtis	2. Aplinkos pavadinimas nėra unikalus. <ul style="list-style-type: none"> 2.1. Vykdomi 1.1 — 1.3 žingsniai, įvestas pavadinimas nėra unikalus. 2.2. Sistema praneša, kad pavadinimas nėra unikalus. 2.3. Sistema prideda unikalų indikatorių (R01), prie aplinkos pavadinimo.

Alternatyva	3. Sistema inicijuoja pavadinimo pakeitimą.
	3.1. Vykdomi 1.2 — 1.4 žingsniai.
Apribojimai:	
Sąlygos prieš	Užkrauta aplinka.
Sąlygos po	Pakeistas aplinkos pavadinimas, pakeitimai įrašyti duomenų bazėje.
Reikalavimai:	
R01	Unikalus indikatorius
	<ul style="list-style-type: none"> Indikatorius pridedamas prie pavadinimo pabaigos. Indikatoriaus formatas — „<skaičius>“, pavyzdžiu „Pavadinimas (1)“. <skaičius> – sveikasis skaičius, pradedamas nuo 1. <skaičius> nustatomas randant didžiausią skaičių, jau naudojamą su tuo pačiu aplinkos pavadinimu, ir jį padidinant vienetu.

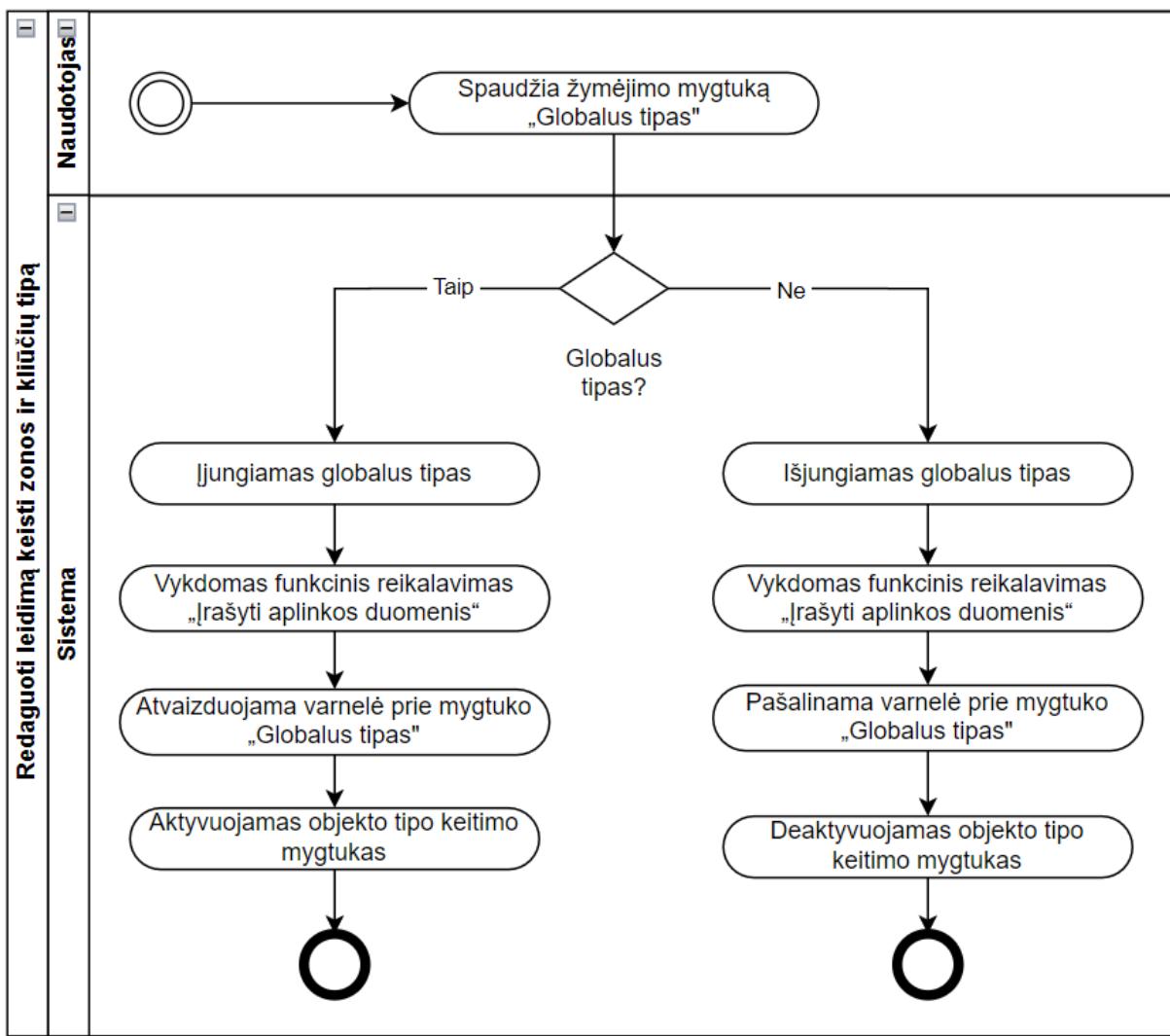


46 pav. Veiklos diagrama funkciniams reikalavimui „Redaguoti aplinkos pavadinimą“

3.6.3..2.1.6 Funkcinis reikalavimas „Redaguoti leidimą keisti zonas ir kliūčių tipą“

22 lentelė. Funkcinis reikalavimas „Redaguoti leidimą keisti zonas ir kliūčių tipą“

Trumpas aprašymas	Naudotojas gali keisti leidimą redaguoti individualių objektų tipą.
Scenarijai:	
Pagrindinis	<p>1. Naudotojas pakeičia leidimo paramетro reikšmę.</p> <p>1.1. Naudotojas paspaudžia leidimo (globalaus tipo) parametro žymėjimo mygtuką (angl. checkbox) (R01).</p> <p>1.2. Sistema pakeičia globalaus tipo parametro reikšmę iš „tiesa“ į „melas“ arba iš „melas“ į „tiesa“.</p> <p>1.3. Sistema įvykdo funkcinį reikalavimą „Irašyti aplinkos duomenis“.</p> <p>1.4. Sistema, naudotojo sasajoje aktyvuoja (jeigu globalus tipas — „tiesa“) arba deaktyvuoja (jeigu globalus tipas — „melas“) objekto tipo keitimo mygtuką objekto parametru skiltyje (vykdant funkcinį reikalavimą „Pasirinkti objektą“).</p> <p>Pilnas scenarijus pavaizduotas 47 paveiksle.</p>
Apribojimai:	
Sąlygos prieš	Naudotojas nėra pasirinkęs jokio objekto.
Sąlygos po	Leidimo parametras pakeistas duomenų bazėje, naudotojo sasaja atvaizduoja atnaujintą būseną.
Reikalavimai:	
R01	<p>„Globalus tipas“ žymėjimo mygtukas</p> <p>Vartotojo sasajos žymėjimo mygtuko būsena:</p> <ul style="list-style-type: none"> Kai parametro reikšmė yra „tiesa“ (angl. true), šalia mygtuko rodoma varnelė. Kai parametro reikšmė yra „melas“ (angl. false), varnelė nėra rodoma.



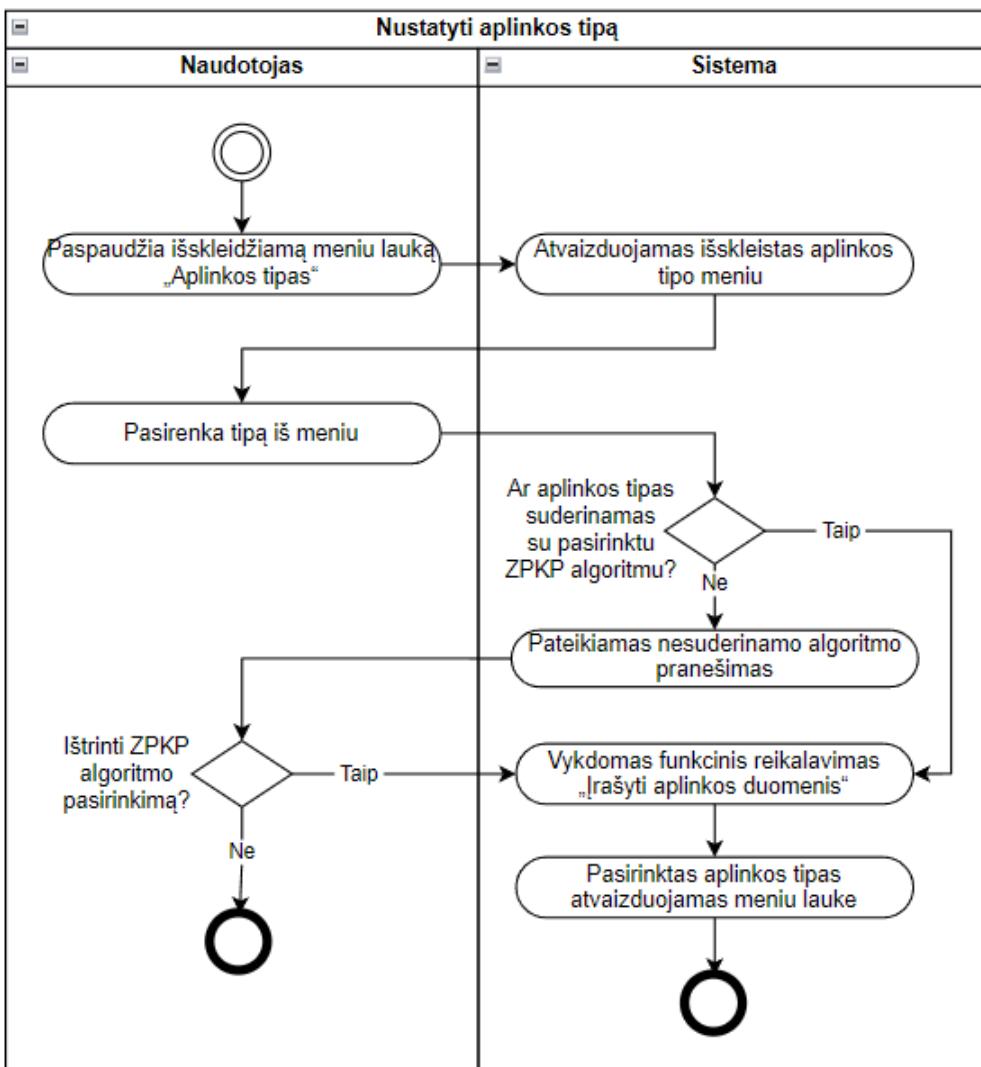
47 pav. Veiklos diagrama funkciniam reikalavimui „Redaguoti leidimą keisti zonas ir kliūčių tipą“

3.6.3..2.1.7 Funkcinis reikalavimas „Nustatyti aplinkos tipą“

23 lentelė. Funkcinis reikalavimas „Nustatyti aplinkos tipą“

Trumpas aprašymas	Naudotojas gali nustatyti aplinkos tipą.
Scenarijai:	
Pagrindinis	<p>1. Naudotojas nustato aplinkos tipą.</p> <p>1.1. Naudotojas spaudžia meniu lauką „Aplinkos tipas“. 1.2. Sistema atvaizduoja išskleistą meniu lauką. 1.3. Naudotojas pasirenka tipą iš meniu. 1.4. Vykdomas funkcinis reikalavimas „Irašyti aplinkos duomenis“.</p> <p>Pilnas scenarijus pavaizduotas 48 paveiksle.</p>
Išimtis	2. Pasirinktas algoritmas nesuderinamas su nauju aplinkos tipu.

	2.1. Vykdomi 1.1 — 1.3 žingsniai. 2.2. Sistema pateikia nesuderinamo algoritmo pranešimą (R01).
1 Alternatyva	3. Naudotojas nurodo pašalinti algoritmo pasirinkimą.
	3.1. Vykdomi 2.1, 2.2 žingsniai. 3.2. Naudotojas paspaudžia mygtuką „Tęsti“. 3.3. Sistema pašalina algoritmo pasirinkimą. 3.4. Tęsiamas „Pagrindinis“ scenarijus.
2 Alternatyva	4. Naudotojas nutraukia aplinkos tipo keitimo procedūrą.
Apribojimai:	
Sąlygos prieš	Naudotojas nėra pasirinkęs jokio objekto.
Sąlygos po	Aplinkos tipo parametras pakeistas duomenų bazėje, naudotojo sąsaja atvainduoja atnaujintą būseną.
Reikalavimai:	
R01	Nesuderinamo algoritmo pranešimo langas
	<p>Lango struktūra:</p> <ul style="list-style-type: none"> • Pateikiamas pranešimas „Pasirinktas ZPKP algoritmas nėra sunderinamas su pasirinktu aplinkos tipu. Tęsti ir panaikinti pasirinktą algoritmą?“ • Mygtukas „Tęsti“ — pašalinamas algoritmo pasirinkimas iš duomenų bazės, atnaujinamas algoritmo pasirinkimo laukas. • Mygtukas „Atšaukti“ — atšaukiamas aplinkos tipo keitimo procesas.



48 pav. Veiklos diagrama funkciniam reikalavimui „Nustatyti aplinkos tipą“

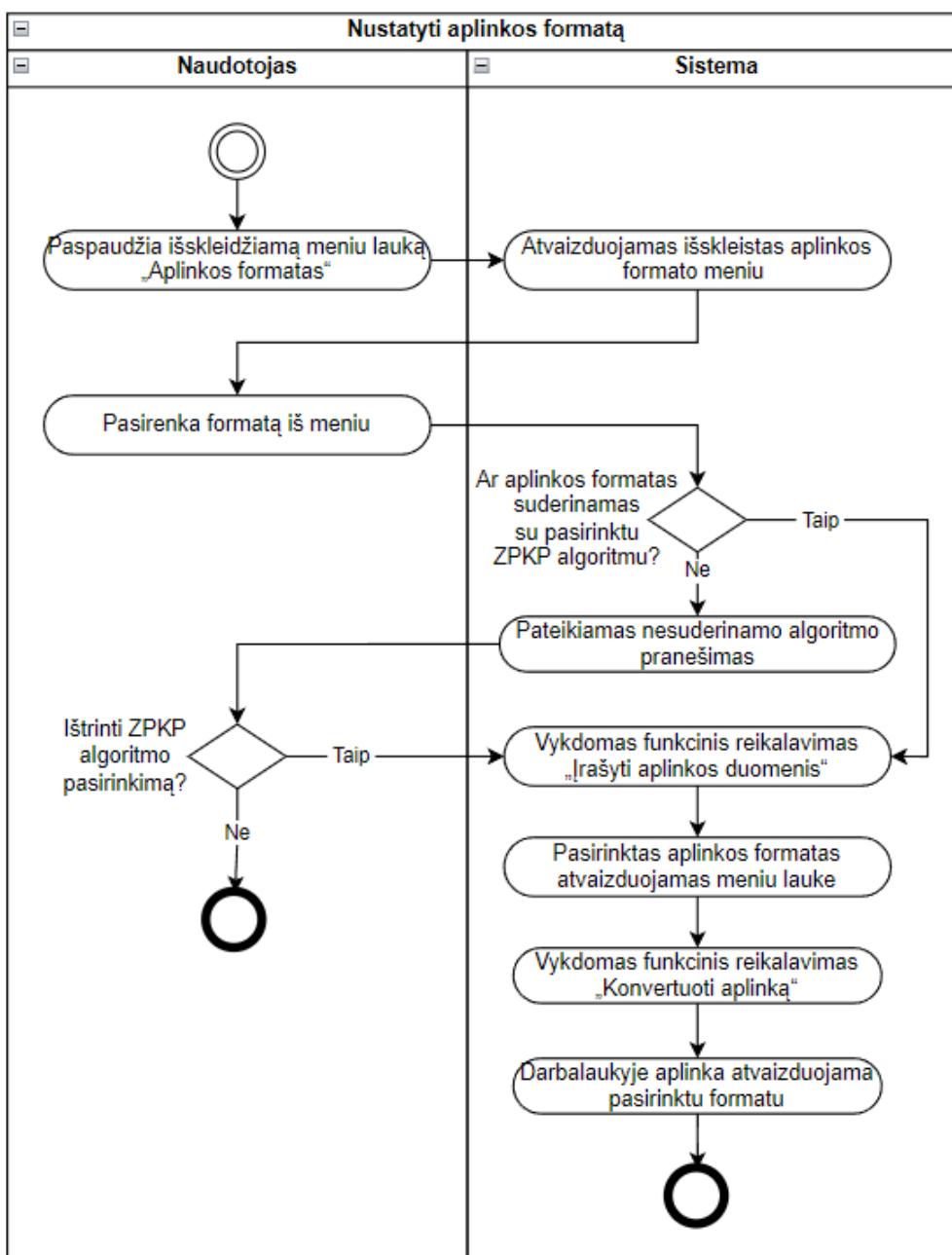
3.6.3..2.1.8 Funkcinis reikalavimas „Nustatyti aplinkos formatą“

24 lentelė. Funkcinis reikalavimas „Nustatyti aplinkos formatą“

Trumpas aprašymas	Naudotojas gali pakeisti aplinkos formatą.
Scenarijai:	
Pagrindinis	1. Naudotojas pakeičia apklinkos formatą.

	<p>1.1. Naudotojas paspaudžia išskleidžiamą meniu lauką „Aplinkos formatas“.</p> <p>1.2. Sistema atvaizduoja išskleistą aplinkos formato meniu lauką.</p> <p>1.3. Naudotojas pasirenka formatą iš meniu.</p> <p>1.4. Vykdomas funkcinis reikalavimas „Išsaugoti aplinkos duomenis“.</p> <p>1.5. Sistema atvaizduoja pasirinktą aplinkos formatą meniu laukę.</p> <p>1.6. Vykdomas funkcinis reikalavimas „Konvertuoti aplinką“.</p> <p>1.7. Sistema darbalaukyje atvaizduoja aplinką pasirinktu formatu.</p> <p>Pilnas scenarijus pavaizduotas 49 paveiksle.</p>
Išimtis	2. Pasirinktas algoritmas nesuderinamas su nauju aplinkos formatu.
	<p>2.1. Vykdomi 1.1 — 1.3 žingsniai.</p> <p>2.2. Sistema pateikia nesuderinamo algoritmo pranešimą (R01).</p>
1 Alternatyva	3. Naudotojas nurodo pašalinti algoritmo pasirinkimą.
	<p>3.1. Vykdomi 2.1, 2.2 žingsniai.</p> <p>3.2. Naudotojas paspaudžia mygtuką „Tęsti“.</p> <p>3.3. Sistema pašalina algoritmo pasirinkimą.</p> <p>3.4. Tęsiamas „Pagrindinis“ scenarijus.</p>
2 Alternatyva	4. Naudotojas nutraukia aplinkos formato keitimo procedūrą.
	<p>4.1. Vykdomi 2.1, 2.2 žingsniai.</p> <p>4.2. Naudotojas paspaudžia mygtuką „Atšaukti“.</p> <p>4.3. Sistema grįžta į pradinę būseną.</p>
Apribojimai:	
Sąlygos prieš	Naudotojas nėra pasirinkęs jokio objekto.
Sąlygos po	Aplinkos formato parametras pakeistas duomenų bazėje, naudojtojo sąsaja atvaizduoja atnaujintą būseną.
Reikalavimai:	
R01	Nesuderinamo algoritmo pranešimo langas

	<p>Lango struktūra:</p> <ul style="list-style-type: none"> Pateikiamas pranešimas „Pasirinktas ZPKP algoritmas nėra suderinamas su pasirinktu aplinkos formatu. Tęsti ir panaikinti pasirinktą algoritmą?“ Mygtukas „Tęsti“ — pašalinamas algoritmo pasirinkimas iš duomenų bazės, atnaujinamas algoritmo pasirinkimo laukas. Mygtukas „Atšaukti“ — atšaukiamas aplinkos formato keitimo procesas.
--	---



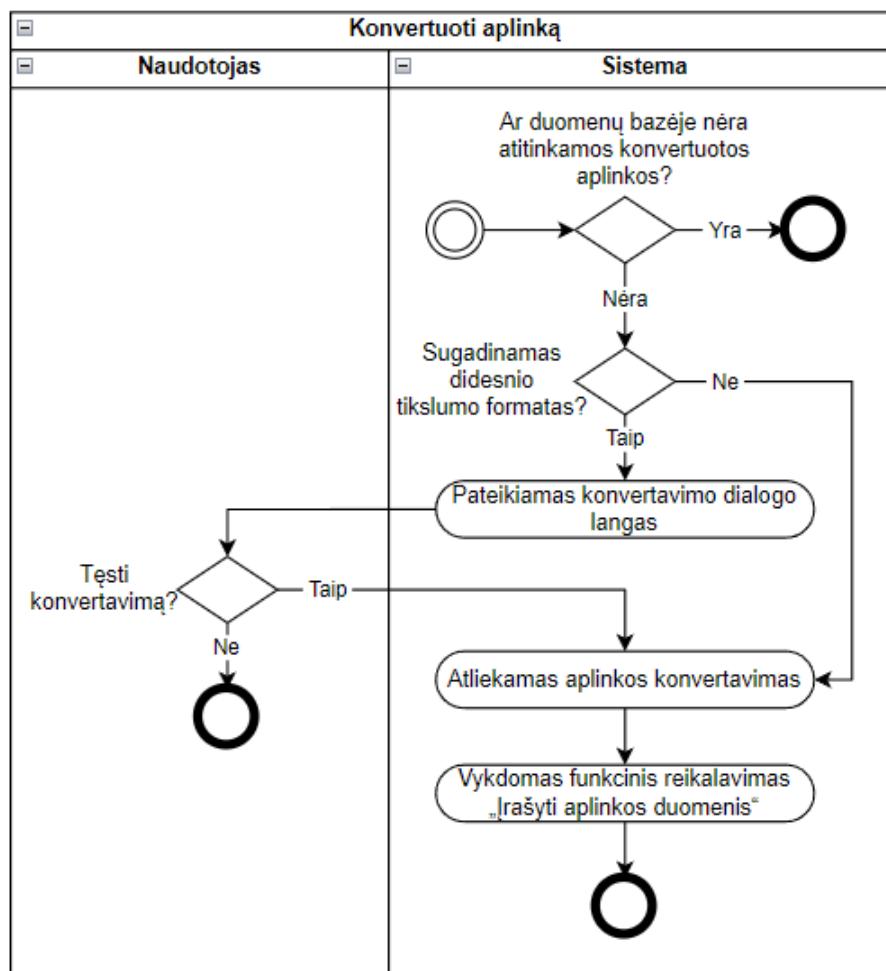
49 pav. Veiklos diagrama funkciniam reikalavimui „Nustatyti aplinkos formatą“

3.6.3..2.1.9 Funkcinis reikalavimas „Konvertuoti aplinką“

25 lentelė. Funkcinis reikalavimas „Konvertuoti aplinką“

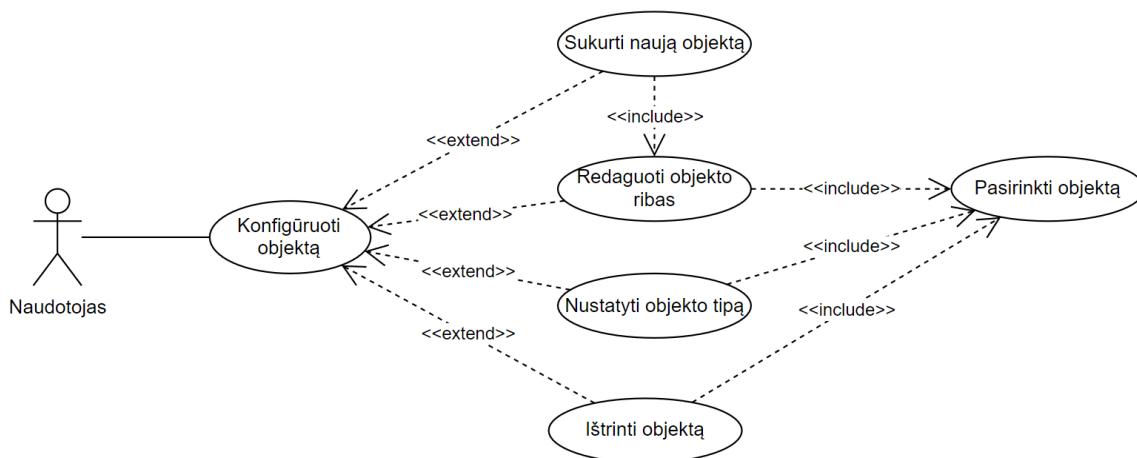
Trumpas aprašymas	Sistema gali konvertuoti vieną aplinkos formatą į kitą.
Scenarijai:	
Pagrindinis	<p>1. Sistema atlieka aplinkos konvertavimą.</p> <p>1.1. Sistema atlieka aplinkos konvertavimą (R01).</p> <p>1.2. Vykdomas funkcinis reikalavimas „Išsaugoti aplinkos duomenis“.</p> <p>Pilnas scenarijus pavaizduotas 50 paveiksle.</p>
1 Išimtis	<p>2. Duomenų bazėje jau yra konvertuota aplinka, atitinkanti esamą aplinkos būseną.</p> <p>2.1. Sistema nutraukia konvertavimo procesą.</p>
2 Išimtis	<p>3. Sugadinamas didesnio tikslumo formatas.</p> <p>3.1. Sistemos būsena aprašyta R02.</p> <p>3.2. Pateikiamas konvertavimo dialogo langas (R03).</p>
2.1 Alternatyva	<p>4. Tęsiamas konvertavimo procesas.</p> <p>4.1. Naudotojas paspaudžia mygtuką „Tęsti“.</p> <p>4.2. Tęsiamas „Pagrindinis“ scenarijus.</p>
2.2 Alternatyva	<p>5. Nutraukiama konvertavimo procesas.</p> <p>5.1. Naudotojas paspaudžia mygtuką „Atšaukti“.</p> <p>5.2. Sistema nutraukia konvertavimo procesą.</p>
Apribojimai:	
Sąlygos prieš	Pakeistas aplinkos formato parametras duomenų bazėje.
Sąlygos po	Sukurtas aplinkos formatas pagal naują aplinkos formato parametro reikšmę.
Reikalavimai:	
R01	Konvertavimo režimai
	<p>Galimi konvertavimo režimai:</p> <ul style="list-style-type: none"> • Daugiakampis formatas į tinklelio formatą. • Tinklelio formatas į daugiakampio formatą.
R02	Aukštesnės kokybės formato sugadinimas

	Naudotojas dirba su aukštesnės kokybės formatu, perjungia formatą į žemesnės kokybės, jį redaguoja ir tuomet perjungia į aukštesnės kokybės formatą. Sistema negali išlaikyti aukštesnės kokybės formato, nes bandoma jį sukurti naudojant žemesnės kokybės formatą, todėl abiejų formatų kokybė susilygina, prarandant aukštesnės kokybės formato tikslumą.
R03	Konvertavimo dialogo langas



50 pav. Veiklos diagrama funkciniam reikalavimui „Konvertuoti aplinką“

3.6.3..2.2 Funkcinis reikalavimas „Konfigūruoti objektą“



51 pav. Funkcinis reikalavimas „Konfigūruoti objektą“

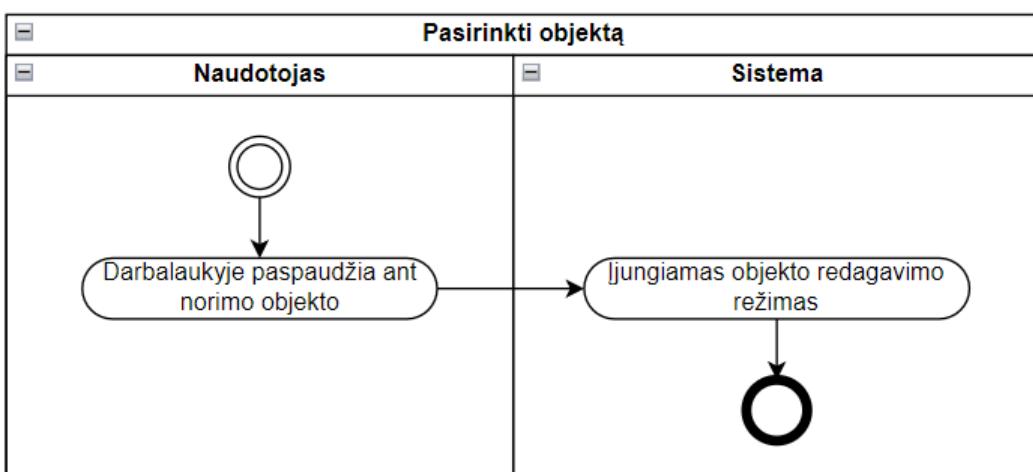
3.6.3..2.2.1 Funkcinis reikalavimas „Pasirinkti objektą“

26 lentelė. Funkcinis reikalavimas „Pasirinkti objektą“

Trumpas aprašymas	Naudotojas gali pasirinkti objektą, kuriam norima atlikti pakeitimus.
Scenarijai:	
Pagrindinis	1. Naudotojas pasirenka objektą
	1.1. Naudotojas darbalaukyje paspaudžia ant norimo objekto (R01). 1.2. Sistema įjungia objekto redagavimo režimą (R02). Pilnas scenarijus pavaizduotas 52 paveiksle.
1 Alternatyva	2. Naudotojas pasirenka kelis objektus
	2.1. Naudotojas darbalaukyje paspaudžia ant norimo objekto laikydamas CTRL mygtuką. 2.2. Sistema atnaujina objekto redagavimo režimą. 2.3. Kartojami 2.1 — 2.2 žingsniai, kol pasirinkti visi norimi objektais.
2 Alternatyva	3. Naudotojas atžymi pasirinktą objektą.
	3.1. Naudotojas darbalaukyje paspaudžia ant pažymėto objekto laikydamas CTRL mygtuką. 3.2. Sistema atžymi pasirinktą objektą, atnaujinamas objekto redagavimo režimas.
Apribojimai:	

Sąlygos prieš	Darbalaukis su sukurtais objektais.
Sąlygos po	Ijungtas pasirinkto objekto redagavimo režimas.
Reikalavimai:	
R01	Objektai
	<p>Sistemoje naudojamos 2 objektų kategorijos:</p> <ul style="list-style-type: none"> • Zonas — teritorija, kurią siekiama padengti. • Kliūties — sritis esanti zonoje, kurios reikia išvengti. <p>Bendri objekto reikalavimai:</p> <ul style="list-style-type: none"> • Objektas turi savo kategoriją (zona arba kliūtis). • Objektas turi savo tipą (žinomas arba nežinomas). • Objektas aprašomas viršunių masyvu. • Viršūnės aprašomos XY koordinačių sistemoje. <p>Zonos reikalavimai:</p> <ul style="list-style-type: none"> • Zonos viršūnės aprašomos laikrodžio rodyklės kryptimi. <p>Kliūties reikalavimai:</p> <ul style="list-style-type: none"> • Kliūties viršūnės aprašomos prieš laikrodžio rodyklės kryptį.
R02	Objekto redagavimo režimas

	<p>Ijungus objekto redagavimo režimą:</p> <ul style="list-style-type: none"> • Pakeičiamas objekto atvaizdavimo stilius. • Atvaizduojami objekto parametrai, kuriuos leidžiama keisti. • Aktyvinami viršūnių pridėjimo ir trynimo mygtukai. • Aktyvinamas objekto trynimo mygtukas. • Leidžiama pasirinkti individualias objekto viršunes. <p>Objekto redagavimo stilius:</p> <ul style="list-style-type: none"> • Paryškintos viršūnės. • Punktyrinės linijos briaunos. • Šrafuotas (angl. hatch) vidinis užpildas. <p>Objekto parametrai (atvaizduojami pasirinkus vieną objektą):</p> <ul style="list-style-type: none"> • Tipas: statinis (objekto pozicija iš anksto žinoma), dinaminis (objekto pozicija prieš vykdant ZPKP nežinoma). • Viršūnių skaičius. • Plotas. • Naudingas plotas — zonas plotas atėmus joje esančių kliūčių plotą. <p>Objektų parametrai (atvaizduojami pasirinkus kelis objektus):</p> <ul style="list-style-type: none"> • Bendras plotas. • Bendras naudingas plotas — zonas plotas atėmus joje esančių kliūčių plotą. <p>Valdymo reikalavimai:</p> <ul style="list-style-type: none"> • Redagavimo režimas išjungiamas paspaudus „ESC“ mygtuką arba paspaudus ant tuščios darbalaukio vietos.
--	--



52 pav. Veiklos diagrama funkciniam reikalavimui „Pasirinkti objektą“

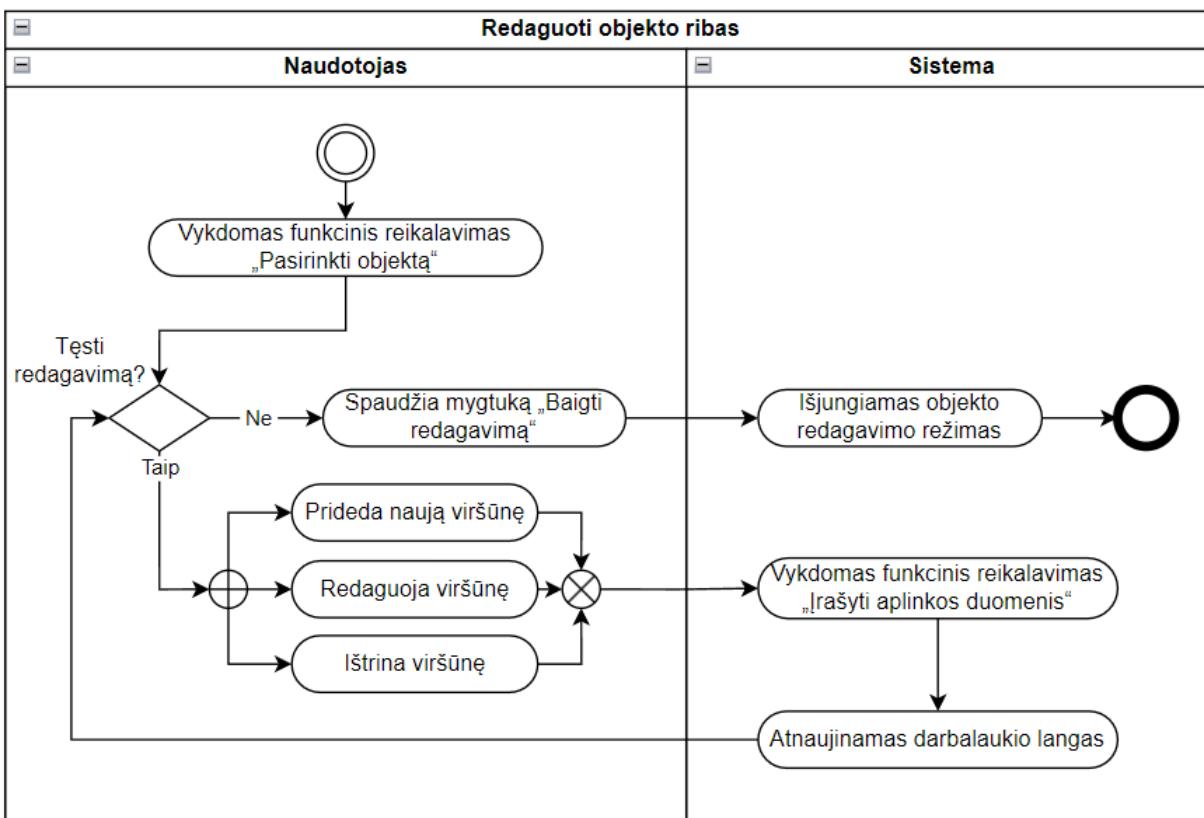
3.6.3..2.2.2 Funkcinis reikalavimas „Redaguoti objekto ribas“

27 lentelė. Funkcinis reikalavimas „Redaguoti objekto ribas“

Trumpas aprašymas	Naudotojas pasirinktam objektui gali redaguoti jo ribas, redagujant viršunes.
Scenarijai:	
1 Scenarijus	1. Pridedama nauja viršūnė
1.1 Alternatyva	1.1. Viršūnė pridedama į masyvo galą
	<ul style="list-style-type: none"> 1.1.1. Naudotojas paspaudžia viršunių pridėjimo mygtuką. 1.1.2. Sistema įjungia viršunių redagavimo režimą (R01). 1.1.3. Naudotojas paspaudžia ant darbalaukio ten, kur nori pridėti viršūnę. 1.1.4. Sistema prideda naują objekto viršūnę, viršūnė įrašoma objekto viršunių masyvo gale. 1.1.5. Kartojami 1.1.3, 1.1.4 žingsniai, kol naudotojas nori pridėti objekto viršunes.
1.2 Alternatyva	1.2. Viršūnė pridedama tarp 2 briaunos viršunių
	<ul style="list-style-type: none"> 1.2.1. Naudotojas spaudžia ant objekto briaunos. 1.2.2. Sistema prideda naują viršūnę spaudimo vietoje, naują viršūnę sujungiant su briaunos viršunėmis. Panaikinama sena briauna, sukuriant naujas 2. 1.2.3. Sistema įjungia viršūnės fokusavimo režimą (R02). 1.2.4. Naudotojas laikydamas pelės mygtuką tempia viršūnę į normą vietą. 1.2.5. Naudotojas atleidžia pelės mygtuką. 1.2.6. Sistema įrašo naują viršūnę objekto masyve tarp 2 senos briaunos viršunių.
2 Scenarijus	2. Redaguojama viršūnė
2.1 Alternatyva	2.1. Grafinis redagavimas
	<ul style="list-style-type: none"> 2.1.1. Naudotojas spaudžia ant objekto viršūnės. 2.1.2. Sistema įjungia viršūnės fokusavimo režimą (R02). 2.1.3. Naudotojas laikydamas pelės mygtuką tempia viršūnę į normą vietą, pakeitimai atvaizduojami realiu laiku. 2.1.4. Naudotojas atleidžia pelės mygtuką. 2.1.5. Sistema įrašo naują viršūnės poziciją objekto viršunių masyve.
2.2 Alternatyva	2.2. Parametrinis redagavimas

	<p>2.2.1. Naudotojas paspaudžia ant objekto viršūnės.</p> <p>2.2.2. Sistema įjungia viršūnės fokusavimo režimą (R02).</p> <p>2.2.3. Naudotojas pakeičia X ašies lauko ir / arba Y ašies lauko vertes.</p> <p>2.2.4. Sistema įrašo viršūnės lokacijos pakeitimus objekto viršūnių masyve, atvaizduoja viršūnės pakeitimus.</p>
3 Scenarijus	3. Ištrinama viršūnė
3.1 Alternatyva	3.1. Serijinis trynimas
	<p>3.1.1. Naudotojas paspaudžia viršūnių trynimo mygtuką.</p> <p>3.1.2. Sistema įjungia viršūnių redagavimo režimą (R01).</p> <p>3.1.3. Naudotojas paspaudžia ant viršūnės, kurią nori pašalinti.</p> <p>3.1.4. Sistema pašalina objekto viršūnę, viršūnė panaikinama iš objekto viršūnių masyvo, atnaujinamos atvaizduojamos objekto briaunos.</p> <p>3.1.5. Kartojami 3.1.3, 3.1.4 žingsniai, kol naudotojas nori trinti objekto viršunes.</p>
3.2 Alternatyva	3.2. Individualus trynimas
	<p>3.2.1. Naudotojas paspaudžia ant objekto viršūnės.</p> <p>3.2.2. Įjungiamas viršūnės fokusavimo režimas (R02).</p> <p>3.2.3. Naudotojas paspaudžia „DEL“ arba trynimo mygtuką.</p> <p>3.2.4. Sistema pašalina objekto viršūnę, viršūnė panaikinama iš objekto viršūnių masyvo, atnaujinamos atvaizduojamos objekto briaunos.</p>
Išimtis	4. Nutrauktas grafinis redagavimas / pridėjimas
	<p>4.1. Naudotojas neatleidęs pelės mygtuko ir paspaudęs „ESC“ mygtuką, nutraukia viršūnės redagavimą / pridėjimą.</p> <p>4.2. Sistema grįžta į ankstesnę būseną prieš pelės mygtuko paspaudimą.</p>
	Pilnas funkcinio reikalavimo scenarijus pavaizduotas 53 paveiksle.
Apribojimai:	
Sąlygos prieš	Įvykdytas funkcinis reikalavimas „Pasirinkti objektą“.
Sąlygos po	Objekto pakeitimai išsaugoti duomenų bazėje, įvykdytas funkcinis reikalavimas „Irašyti aplinkos duomenis“.
Reikalavimai:	
R01	Viršūnių redagavimo režimas

	<p>Sistema pakeičia darbinės aplinkos stilių:</p> <ul style="list-style-type: none"> • Viršunių pridėjimo mygtukas paryškinamas. • Kiti mygtukai atjungiami. • Darbalaukio lango apvadas pakeičia spalvą. • Režimas išjungiamas naudotojui atliekant kitą veiksmą arba spaudus „ESC“ mygtuką.
R02	Viršūnės fokusavimo režimas



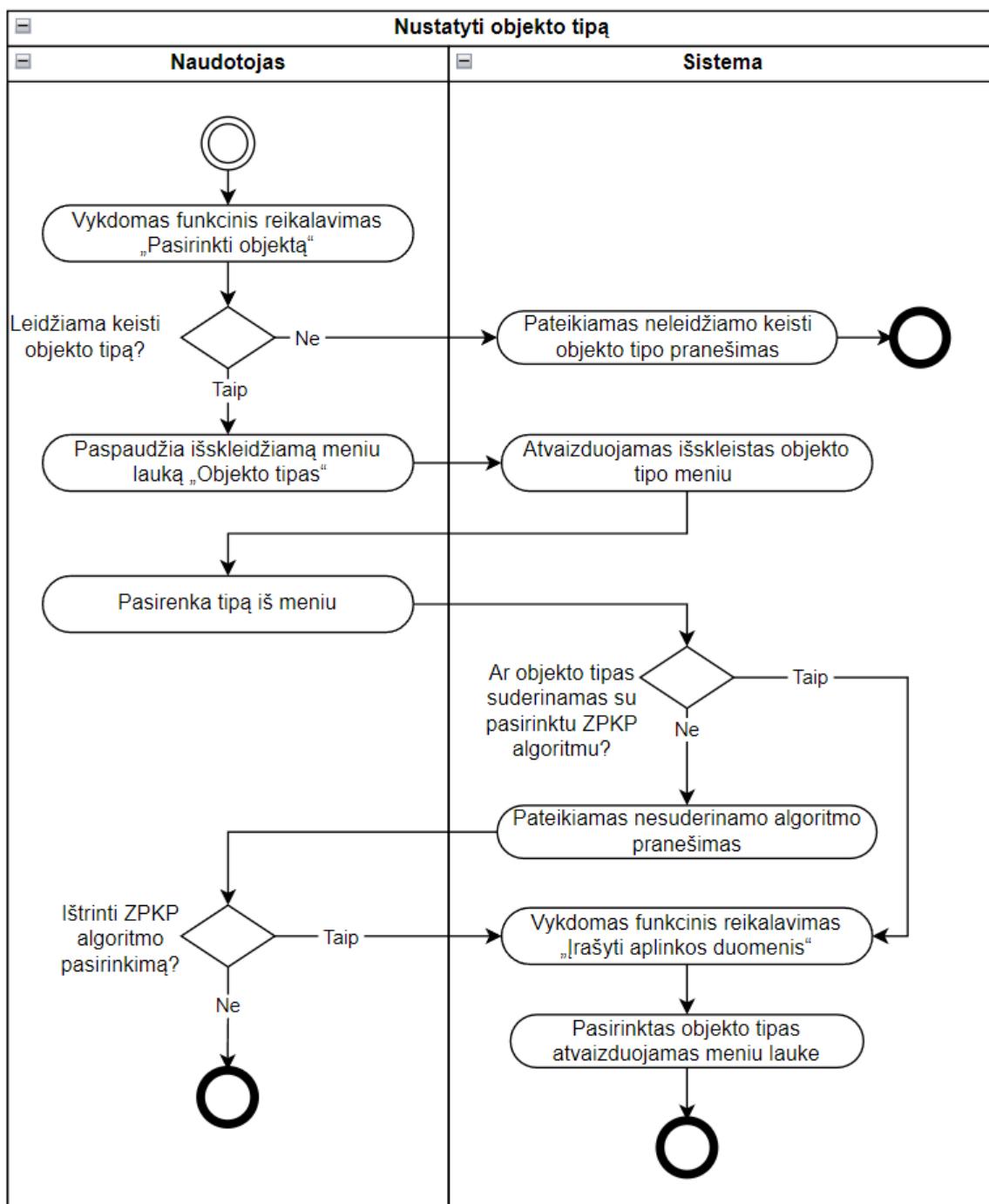
53 pav. Veiklos diagrama funkciniam reikalavimui „Redaguoti objekto ribas“

3.6.3.2.2.3 Funkcinis reikalavimas „Nustatyti objekto tipą“

28 lentelė. Funkcinis reikalavimas „Nustatyti objekto tipą“

Trumpas aprašymas	Naudotojas gali keisti objekto tipą.
Scenarijai:	
Pagrindinis	1. Pakeičiamas objekto tipas

	<p>1.1. Naudotojas spaudžia meniu lauką „Objekto tipas“.</p> <p>1.2. Sistema atvaizduoja meniu su galimais pasirinkti objekto tipais (R01).</p> <p>1.3. Naudotojas pasirenka tipą iš meniu.</p> <p>1.4. Sistema išsaugo naujają objekto tipą duomenų bazėje.</p> <p>1.5. Sistema atvaizduoja nauja objekto tipo reikšmę meniu lauke.</p> <p>Pilnas scenarijus pavaizduotas 54 paveiksle.</p>
Išimtis	2. Neleidžiama keisti objekto tipo
	<p>2.1. Vykdomas 1.1 žingsnis.</p> <p>2.2. Sistema neleidžia keisti objekto tipo, dėl aplinkos nustatymų (žr. funkcinis reikalavimas „Redaguoti leidimą keisti zonas ir kliūčių tipą“).</p>
1 Alternatyva	3. Objekto tipas nesuderinamas su pasirinktu algoritmu (pašalinamas algoritmo pasirinkimas)
	<p>3.1. Vykdomi 1.1 — 1.3 žingsniai.</p> <p>3.2. Sistema pateikia nesuderinamo algoritmo pranešimą, nes algoritmo negalima vykdyti su pasirinkto tipo objektu.</p> <p>3.3. Naudotojas nurodo pašalinti algoritmo pasirinkimą.</p> <p>3.4. Sistema atnaujina algoritmo nustatymus duomenų bazėje.</p> <p>3.5. Vykdomi 1.4 — 1.5 žingsniai.</p>
2 Alternatyva	4. Objekto tipas nesuderinamas su pasirinktu algoritmu (nutraukiama objekto tipo keitimo procesas)
	<p>4.1. Vykdomi 3.1, 3.2 žingsniai.</p> <p>4.2. Naudotojas nurodo nutraukti objekto tipo keitimo procesą.</p> <p>4.3. Sistema grįžta į pradinę būseną (žr. sąlygos prieš).</p>
Apribojimai:	
Sąlygos prieš	Jvykdytas funkcinis reikalavimas „Pasirinkti objektą“.
Sąlygos po	Objekto pakeitimai išsaugoti duomenų bazėje, jvykdytas funkcinis reikalavimas „Irašyti aplinkos duomenis“.
Reikalavimai:	
R01	Objekto tipai
	<p>Meniu lauke „Objekto tipas“ pateikiami pasirinkimai:</p> <ul style="list-style-type: none"> • Statitnis (angl. Off-line) — objektas, kurio lokacija žinoma prieš generuojant ZPK (zonos padengimo kelią). • Dinaminis (angl. On-line) — objektas, kurio lokacija nežinoma prieš generuojant ZPK.



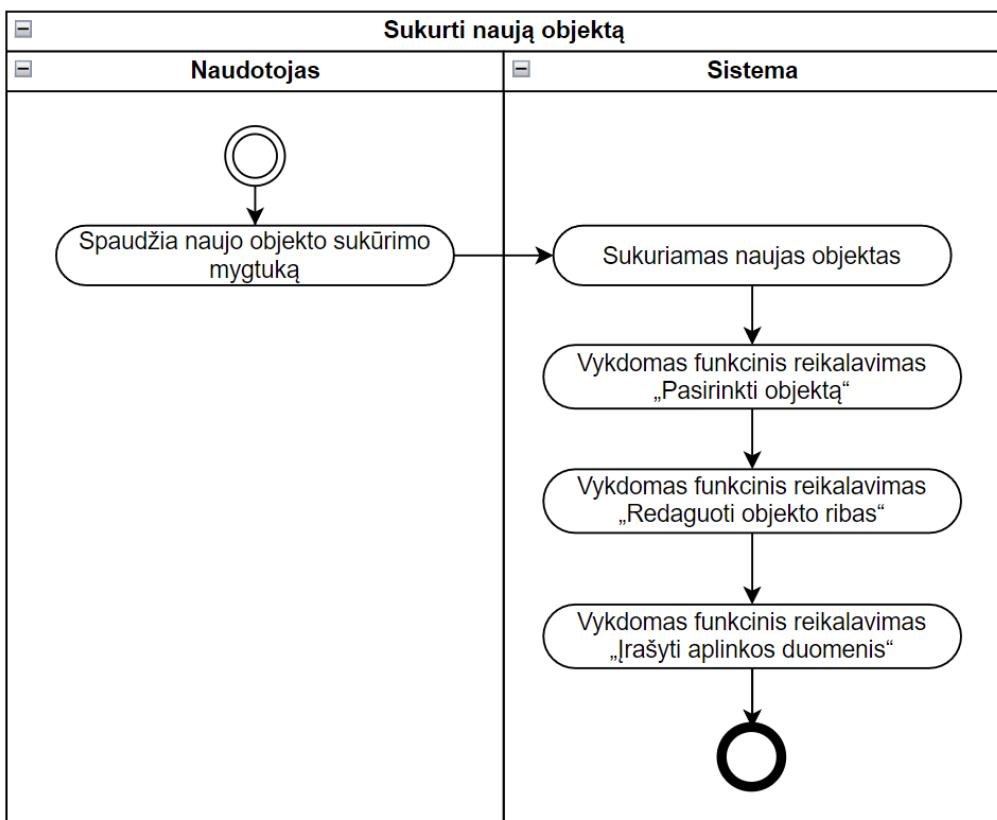
54 pav. Veiklos diagrama funkciniam reikalavimui „Nustatyti objekto tipą“

3.6.3..2.2.4 Funkcinis reikalavimas „Sukurti naują objektą“

29 lentelė. Funkcinis reikalavimas „Sukurti naują objektą“

Trumpas aprašymas	Naudotojas gali pridėti naują objektą.
Scenarijai:	
Pagrindinis	1. Naudotojas sukuria naują objektą.

	<p>1.1. Naudotojas paspaudžia objekto pridėjimo mygtuką (R01).</p> <p>1.2. Sistema sukuria naują objektą.</p> <p>1.3. Vykdomas funkcinis reikalavimas „Pasirinkti objektą“, 1.2 žingsnis.</p> <p>1.4. Vykdomas funkcinis reikalavimas „Redaguoti objekto ribas“, vykdomi 1.1.2 — 1.1.5 žingsniai.</p> <p>Pilnas scenarijus pavaizduotas 55 paveiksle.</p>
Apribojimai:	
Sąlygos prieš	Užkrauta aplinka.
Sąlygos po	Objekto pakeitimai išsaugoti duomenų bazėje, įvykdytas funkcinis reikalavimas „Įrašyti aplinkos duomenis“.
Reikalavimai:	
R01	Objekto pridėjimo mygtukas
	<p>Naudojami kelių objektų tipų mygtukai:</p> <ul style="list-style-type: none"> • Zonas — sukuriamas naujas zonas objektas. • Kliūties — sukuriamas naujas kliūties objektas.
R02	Objekto duomenys
	<p>Objekto duomenis sudaro:</p> <ul style="list-style-type: none"> • ID - unikalus objekto numeris. • Kategorija — zona arba kliūtis. • Tipas — žinomas (angl. Off-line) arba nežinomas (angl. Online). • Viršunių masyvas. Jeigu aplinkos formatas yra daugiakampis (angl. polygonal), zonas viršunės įrašomos palei laikrodžio rodykle, kliūties — prieš.



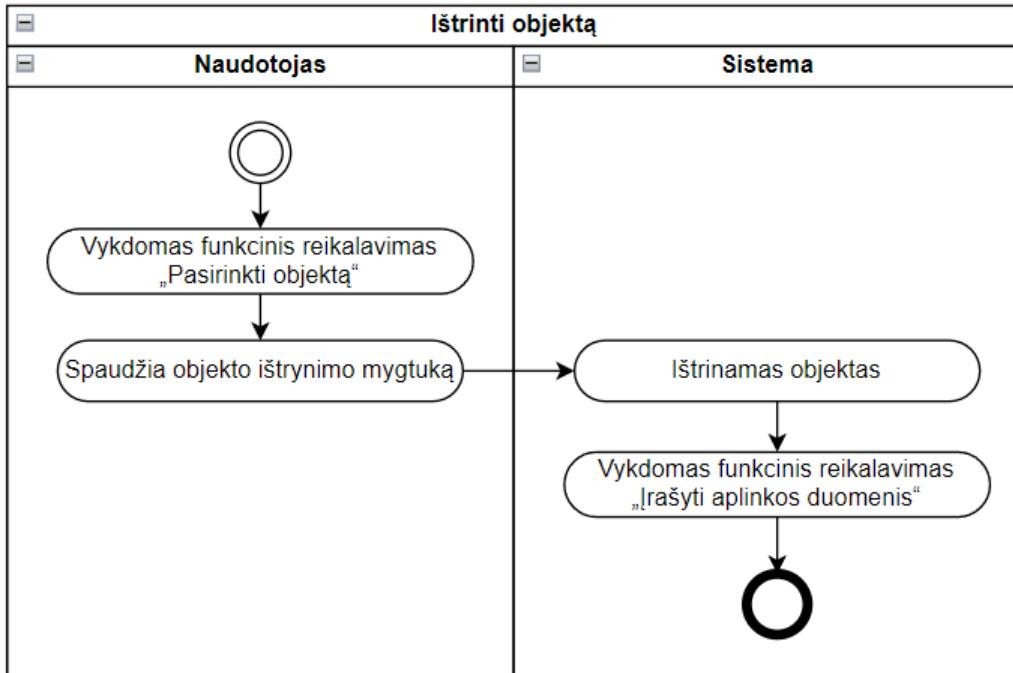
55 pav. Veiklos diagrama funkciniam reikalavimui „Sukurti naują objektą“

3.6.3..2.2.5 Funkcinis reikalavimas „Ištrinti objektą“

30 lentelė. Funkcinis reikalavimas „Ištrinti objektą“

Trumpas aprašymas	Naudotojas gali pašalinti pasirinktą objektą.
Scenarijai:	
Pagrindinis	1. Naudotojas ištrina objektą. 1.1. Vykdomas funkcinis reikalavimas „Pasirinkti objektą“. 1.2. Naudotojas spaudžia objekto ištrynimo mygtuką (R01). 1.3. Objektas ištrinamas, atnaujinama su objektu susijusi informacija. 1.4. Vykdomas funkcinis reikalavimas „Įrašyti aplinkos duomenis“. 1.5. Sistema atvaizduoja pasikeitusią aplinkos būseną. Pilnas scenarijus pavaizduotas 56 paveiksle.
Apribojimai:	
Sąlygos prieš	Sukonfigūruota aplinka su bent 1 objektu.
Sąlygos po	Objektas ištrintas iš duomenų bazės, susiję duomenys atnaujinti, darbalaukio vaizdas atnaujintas.
Reikalavimai:	

R01	Objekto trynimas
	<p>Objekto ištrynimo veiksmą galima iniciuoti 2 būdais:</p> <ul style="list-style-type: none"> „Trinti“ vartotojo sąsajos mygtuku. „DELETE“ klaviatūros mygtuku.



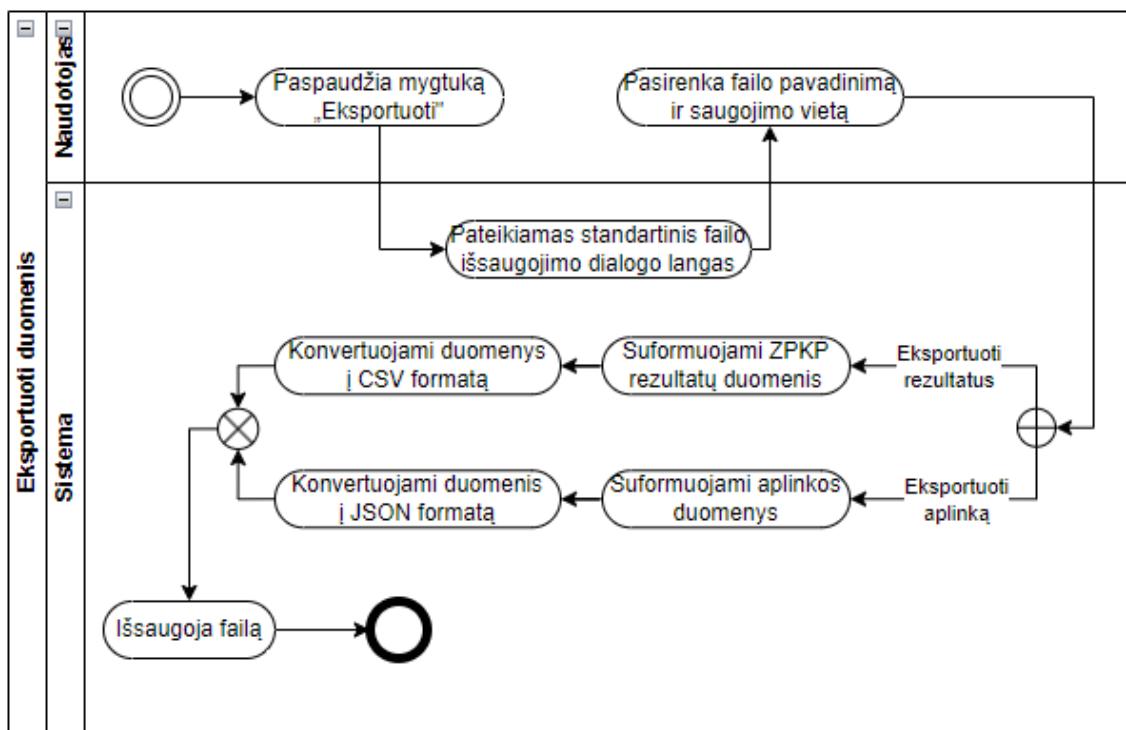
56 pav. Veiklos diagrama funkciniam reikalavimui „Ištrinti objektą“

3.6.4. Funkcinis reikalavimas „Eksportuoti duomenis“

31 lentelė. Funkcinis reikalavimas „Eksportuoti duomenis“

Trumpas aprašymas	Naudotojas gali eksportuoti ZPKP rezultatus ir aplinkos duomenis.
Scenarijai:	
1 Scenarijus	1. Naudotojas eksportuoja aplinkos failą

	<p>1.1. Naudotojas spaudžia mygtuką „Eksportuoti“ esantį „Failas“ skirtuke.</p> <p>1.2. Sistema atidaro standartinj failo išsaugojimo dialogo langą.</p> <p>1.3. Naudotojas pasirenka saugojimo vietą ir, jei reikia, jveda fai-lo pavadinimą. Duomenis turi būti galima eksportuoti JSON formatu.</p> <p>1.4. Sistema surenka aplinkos duomenis iš sistemos duomenų bazės.</p> <p>1.5. Sistema suformuoja grąžinamą failą, kuriame saugomi R01 apibrėžti duomenys.</p> <p>Pilnas scenarijus pavaizduotas 57 paveiksle.</p>
2 Scenarijus	2. Naudotojas eksportuoja ZPKP rezultatų failą
	<p>2.1. Naudotojas spaudžia mygtuką „Eksportuoti“ esantį ZPKP rezultatų lange.</p> <p>2.2. Sistema atidaro standartinj failo išsaugojimo dialogo langą.</p> <p>2.3. Naudotojas pasirenka saugojimo vietą ir, jei reikia, jveda fai-lo pavadinimą. Duomenis turi būti galima eksportuoti CSV, JSON formatu.</p> <p>2.4. Sistema surenka rezultatų duomenis iš sistemos duomenų bazės.</p> <p>2.5. Sistema suformuoja grąžinamą failą, kuriame saugomi R02 apibrėžti duomenys.</p> <p>Pilnas scenarijus pavaizduotas 57 paveiksle.</p>
Apribojimai:	
Sąlygos prieš	Vykstant 1 Scenarijų aplinka néra tuščia ir yra išsaugota Vykdant 2 Scenarijų naudotojas įvykdė funkcinj reikalavimą „Iniciuoti ZPKP“ arba „Vykdyti ZPKP testavimą“
Sąlygos po	Grąžinamas failas pasirinktu formatu pagal pasirinktus eksportavimo duomenis.
Reikalavimai:	
R01	Aplinkos failo duomenų struktūra
	Eksportuojami duomenys atitinka duomenų bazės struktūrą: • Žr. funkcinis reikalavimas „Išsaugoti aplinkos duomenis“, R01.
R02	ZPKP rezultatų failo duomenų struktūra
	Eksportuojami duomenys atitinka duomenų bazės struktūrą: • Žr. funkcinis reikalavimas „Vykdyti ZPKP“, R03.



57 pav. Veiklos diagrama funkciniam reikalavimui „Eksportuoti duomenis”

3.6.5. Funkcinis reikalavimas „Valdyti darbalaukio peržiūrą”

32 lentelė. Funkcinis reikalavimas „Valdyti darbalaukio peržiūrą”

Trumpas aprašymas	Naudotojas gali valdyti kaip aplinka atvaizduojama darbalaukio lauke.
Scenarijai:	
1 Scenarijus	1. Naršyti tarp darbalaukio <ol style="list-style-type: none"> 1.1. Naudotojas darbalaukyje, paspaudęs pelės ratuko mygtuką, stumia pelę per ekraną. 1.2. Sistema atnaujina vaizdą tempimo metu ir keičia atvaizduojamą aplinkos langą, pelės tempimo kryptimi.
1 Alternatyva	1. Naršymas horizontaliai <ol style="list-style-type: none"> 1.1. Naudotojas darbalaukyje, paspaudęs klaviatūros SHIFT mygtuką, suka pelės ratuką. 1.2. Sistema atnaujina vaizdą sukimo metu ir keičia atvaizduojamą aplinkos langą, pagal R01 slenkant vaizdą horizontaliai.
2 Alternatyva	2. Naršymas vertikaliai <ol style="list-style-type: none"> 2.1. Naudotojas darbalaukyje, suka pelės ratuką. 2.2. Sistema atnaujina vaizdą sukimo metu ir keičia atvaizduojamą aplinkos langą, pagal R02 slenkant vaizdą vertikaliai.

2 Scenarijus	2. Keisti vaizdo mastelį
	<p>2.1. Naudotojas darbalaukyje, paspaudęs klaviatūros CTRL mygtuką, suka pelės ratuką.</p> <p>2.2. Sistema atnaujina vaizdą sukimo metu ir keičia atvaizduojamą aplinkos langą, pagal R03 didinant ar mažinant atvaizduojamą aplinkos plotą.</p>
3 Scenarijus	3. Atstatyti pradinį vaizdą
	<p>3.1. Naudotojas paspaudžia mygtuką „Atstatyti vaizdą“ esantį „Rodymas“ skirtuke.</p> <p>3.2. Sistema atstato vaizdą į pradinę poziciją, atstato pradinį mastelį.</p>
4 Scenarijus	4. Perjungti pagalbinį tinklelį
	<p>4.1. Naudotojas paspaudžia mygtuką „Tinklelis“ esantį „Rodymas“ skirtuke.</p> <p>4.2. Sistema atnaujina statuso varnelę esančią šalia „Tinklelis“ mygtuko.</p> <p>4.3. Sistema atnaujina darbalaukio tinklelj.</p>
Apribojimai:	
Sąlygos prieš	Aplinkos duomenys yra išsaugoti duomenų bazėje
Sąlygos po	Atnaujintas darbalaukio atvaizdavimas. Atvaizdavimo parametrai išsaugoti duomenų bazėje, surišti su naudojama aplinka.
Reikalavimai:	
R01	Pelės valdymas naršant horizontaliai
	<p>Pelės ratukas sukamas į viršų:</p> <ul style="list-style-type: none"> • darbalaukio vaizdas stumiamas į kairę pusę. <p>Pelės ratukas sukamas į apačią:</p> <ul style="list-style-type: none"> • darbalaukio vaizdas stumiamas į dešinę pusę.
R02	Pelės valdymas naršant vertikaliai
	<p>Pelės ratukas sukamas į viršų:</p> <ul style="list-style-type: none"> • darbalaukio vaizdas stumiamas į viršų. <p>Pelės ratukas sukamas į apačią:</p> <ul style="list-style-type: none"> • darbalaukio vaizdas stumiamas į apačią.
R03	Pelės valdymas keičiant mastelį
	<p>Pelės ratukas sukamas į viršų:</p> <ul style="list-style-type: none"> • darbalaukio vaizdas priartinamas. <p>Pelės ratukas sukamas į apačią:</p> <ul style="list-style-type: none"> • darbalaukio vaizdas atitolinamas.

3.7. Vartotojo sasajos schema

Šiame skyriuje pateikiama vartotojo sasajos išdėstymo schemas. Pagrindinio lango — darbalaukio — schema pateikta 58 paveiksle. Pagrindinis langas sudarytas iš: sistemos valdymo juostos, įrankių juostos, nustatymo skydelio, centrinės darbo srities ir informacinio, rezultatų skydelio. Viršutinėje sasajos dalyje pateikiama pagrindinė sistemos valdymo juosta. Ją sudaro:

- Logotipas — sistemos logotipas.
- Aplinkos pavadinimas — aktyvios aplinkos pavadinimas, redaguojamas 2 kartus paspaudus ant pavadinimo arba paspaudus skirtuką „Failas“ ir atsidariusiame meniu „Redaguoti aplinkos pavadinimą“.
- Meniu juosta, kurią sudaro skirtukai:
 - „Failas“ — aplinkos kūrimas, redagavimas, atidarymas ir eksportavimas.
 - „Peržiūra“ – vaizdavimo ir sasajos nustatymai, matavimo vienetų nustatymai.

Po sistemos valdymo juostos pateikiama įrankių juosta, leidžianti vykdyti sistemos funkcijas:

- Mastelio keitimo įrankiai.
- Mygtukas „Pridėti naują zoną“ — inicijuojama nauja, tuščia zona ir įjungiamas objekto redagavimo režimas.
- Mygtukas „Pridėti naują kliūtį“ — inicijuojama nauja, tuščia kliūtis ir įjungiamas objekto redagavimo režimas.
- Mygtukas „Trinti“ — įjungiamas / išjungiamas trynimo režimas.
- Mygtukas „Inicijuoti ZPKP“ — vykdomas zonas padengimo kelio planavimo algoritmas.
- Mygtukas „Išvalyti aplinkos rezultatus“ – pašalina sugeneruotus maršrutus ir rezultatus.
- Mygtukas „Atidaryti testavimo sąrašą“ — atidaromas testavimo scenarijų sąrašo langas.
- Mygtukas „Atidaryti testavimo rezultatų sąrašą“ — atidaromas jvykdytų testavimų scenarijų rezultatų langas.

Kairėje pusėje, nustatymų skydelyje pateikiami visi aplinkos ir algoritmo konfigūracijos parametrai:

- Skyrelis „Aplinka“:
 - Tipas.
 - Formatas.
 - Globalus tipas.

- Skyrėlis „Algoritmas“:
 - Algoritmas – pasirenkamas naudojamas ZPKP metodas.
 - Kelio plotis.
 - Kelio persidengimas.
 - „API“ galutinis taškas – serverio adresas.
 - „API“ raktas – naudojama prisijungimui prie išorinės sistemos.

Centrinėje dalyje naudotojas apibrėžia objektus, naršo po aplinką, peržiūri algoritmo rezultatus, maršrutą:

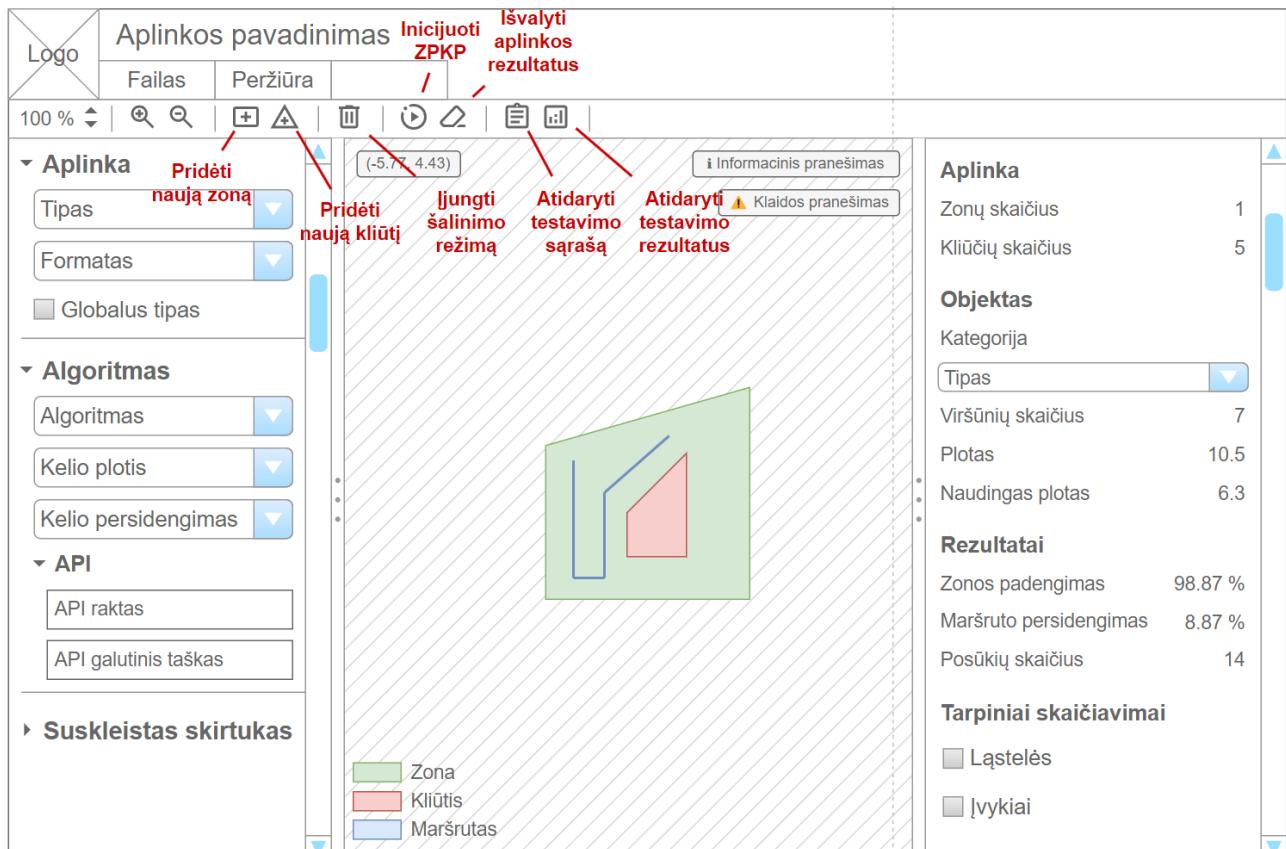
- Zona – žalia spalva pažymėta sritis.
- Kliūtis – raudona spalva pažymėta sritis.
- Maršrutas – mėlyna linija rodanti sugeneruotą kelią.
- Viršuje, kairėje pateikiamos žymeklio koordinatės.
- Viršuje, dešinėje pateikiami informaciniai ir klaidos pranešimai.
- Apačioje, kairėje pateikiama legenda, kurią galima išjungti „Peržiūros“ skirtuke.
- Tinklelio fonas padeda nustatyti objektų padėtį, orientuotis aplinkoje.

Dešinėje pusėje, informaciame, rezultatų skydelyje pateikiami aplinkos ir objektų duomenys, kokybinės ZPKP vykdymo metrikos:

- Aplinka:
 - Zonų skaičius.
 - Kliūčių skaičius.
- Objektas:
 - Kategorija.
 - Tipas, leidžiama keisti.
 - Viršunių skaičius.
 - Plotas.
 - Naudingas plotas, rodoma zonas objektams.
- Rezultatai, kokybinės metrikos:
 - Zonos padengimas (%).
 - Maršruto persidengimas (%).

- Posūkių skaičius.
- Tarpiniai skaičiavimai (kontroliuojamas jų atvaizdavimas):
 - Ląstelės.
 - Įvykiai.

Naudotojas šiame lange sukuria arba pakoreguoja aplinką, parenka algoritmo parametrus ir paleidžia ZPKP algoritmą, o sistema sugeneruoja maršrutą ir atvaizduoja tiek vizualinius, tiek skaitinius rezultatus.



58 pav. Darbalaukio lango išdėstymas

3.8. Loginė duomenų bazės struktūros schema

Šiame skyriuje pateikiama sistemos loginė duomenų bazės schema (žr. 59 pav.). Sistemos duomenų struktūra sudaryta iš trijų pagrindinių objektų: aplinkos, algoritmo, testavimo scenarijaus ir sisteminių parametru.

Sisteminių parametrai: matavimo vienetai (sistemos lygmeniu naudojami matavimo vienetai), rezultatų archyvavimas (taip/ne, nustatoma ar norima archyvuoti rezultatus duomenų bazéje, kad juos vėliau būtų galima peržiūrėti ir eksportuoti).

Kiekviena aplinka turi savo duomenis: ID, pavadinimas (naudotojo įvestą), formatas (daugia-kampus/tinklelis), tipas (žinoma/nežinoma), globalus tipas (taip/ne), zonas objektų skaičius ir kliūčių

objektų skaičius (objektų skaičiai dažnai atvaizduojamos reikšmės, todėl mažinant duomenų bazės operacijų skaičių reikšmių apskaičiavimui, reikšmės išsaugomos duomenų bazėje).

Aplinka yra sudaryta iš daugelio objektų. Kiekvienas objektas turi savo duomenis: ID, aplinkos ID (kuriai aplinkai objeketas priklauso), kategorija (zona/kliūtis), tipas (žinoma/nežinoma), viršunių skaičius ir plotas (viršunių skaičius ir plotas, taip pat dažnai atvaizduojami, todėl jrašomi į duomenų bazę).

Objektai sudaryti iš daugelio viršunių. Kiekviена viršūnė turi savo duomenis: ID, objekto ID (kuriam objektui priklauso viršūnė), sekančios viršūnės ID (skirta sekti ir keisti viršunių sekai), X koordinatė, Y koordinatė.

Kiekvienna aplinka turi savo atvaizdavimo duomenis: centrinis taškas (taškas esantis aplinkos lango viduryje), mastelis (procentinė išraiška, nurodo padidinto vaizdo lygį), tinklelis (taip/ne, kontroliuoja pagalbinio tinklelio atvaizdavimą), matavimo vienetai (pagal nutylėjimą naudojami sisteminiai matavimo vienetai, bet galima naudoti individualius matavimo vienetus pasirinkta aplinkai atskirai).

Kiekvienas algoritmas turi savo duomenis: ID, pavadinimas, „API“ galinis taškas, „API“ raktas. „API“ duomenys skirti sistemą prijungti prie išorinių sistemų, kurios atliktu skaičiavimus.

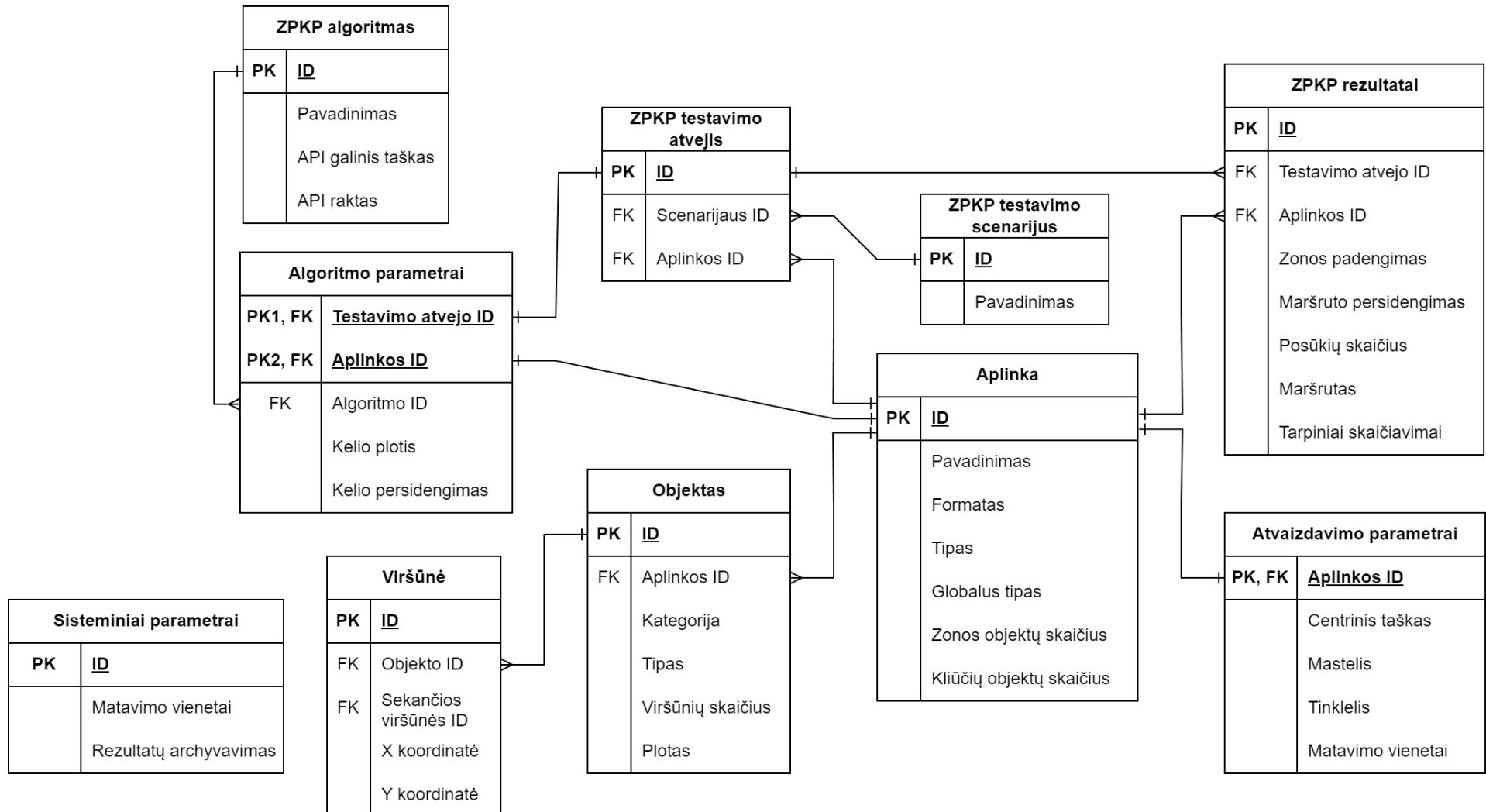
Algoritmo parametrai: kelio plotis ir kelio persidengimas, saugomi atskirai nuo algoritmo, kad galėtų būti individualiai parenkami aplinkai arba testavimo atvejui. Algoritmų parametrų lentelė veikia kaip aplinkos ir testavimo atvejo lentelių praplėtimas. Vienu metu algoritmų parametrų jrašas gali turėti ryšį tik su aplinka arba tik su testavimo atveju. Aplinkai priskirtas algoritmas ir parametrai naudojami, kai naudotojas vykdė funkcinį reikalavimą „Inicijuoti ZPKP“.

ZPKP testavimo scenarijai sudaryti iš daugelio testavimo atvejų. Scenarijaus duomenys: ID ir pavadinimas. Scenarijaus tikslas — apjungti testavimo atvejus, kurie galėtų būti vykdomi visi kartu.

ZPKP testavimo atvejis turi savo duomenis: ID, scenarijaus ID (kuriam scenarijui atvejis priklauso), aplinkos ID (kuri aplinka naudojama vykdant testavimo atvejį). Per algoritmų parametrų lentelę, testavimo atvejis sujungiamas su ZPKP algoritmu.

ZPKP rezultatai turi savo duomenis: ID, testavimo atvejo ID (kurį testavimo atvejį vykdant gauti rezultatai), aplinkos ID (su kuria aplinka naudotojo inicijuotas ZPKP), zonas padengimas, maršruto persidengimas, posūkių skaičius, maršrutas (saugomas kaip vientisas sąrašas, nes jis negali būti modifikuojamas), tarpiniai skaičiavimai (informacija saugoma kaip vientisas objekto, nes negali būti modifikuojamas).

Rezultatų archyvavimo metu duomenų bazėje išsaugomos ZPKP rezultatų, aplinkos (įskaitant objektų su jų viršūnėmis duomenis), testavimo scenarijus (su testavimo atvejais), algoritmo duomenys. Padaromos jų kopijos su naujais ID (archyvavimo numeriu serija), kad eigoje keičiant duomenis, nebūtų prarasti tuo metu naudoti duomenys.



59 pav. Loginė duomenų bazės schema

4. Igyvendinimo dalis

Šiame skyriuje pateikiamas sukurtas sistemos prototipas, kurio naudotojo sąsaja pateikta 60 paveiksle. Prototipas tenkina minimalius sistemos funkcionalumo reikalavimus, tokius kaip: aplinkos konfigūravimas (zonos ir kliūčių apibrėžimas), ZPKP inicijavimas (naudojant vieną sistemoje apibrėžtą algoritmą), algoritmo parametrų keitimas, dalių informacijos pateikimas, atvaizdavimo kontrolė, aplinkos išsaugojimas, užkrovimas.

4.1. Realizacija

Sistema remiasi kliento — serverio architektūra. Serverio pusė parašyta „C“ kalba, naudojama „Mongoose“ „HTTP“ serverio biblioteka — ji aptarnauja „HTTP“ užklausas, apdoroja „JSON“ duomenis ir vykdo Bustrofedono dekompozicijos algoritmą.

Kliento pusė yra standartinė „JavaScript“ aplikacija su „HTML5 Canvas“ elementu, leidžiančiu interaktyviai braižyti aplinkos ribas ir kliūtis. Komunikacija tarp kliento ir serverio vyksta per „REST“ tipo „API“ — aplinkos duomenys siunčiami „JSON“ formatu, serveris juos apdoroja ir grąžina trajektorijos koordinates su tarpiniais skaičiavimais, kuriuos galima atvaizduoti.

Aplinkos duomenys saugomi kaip „JSON“ objektai. Išsaugoti aplinkos failai laikomi serverio failų sistemoje „JSON“ formatu.

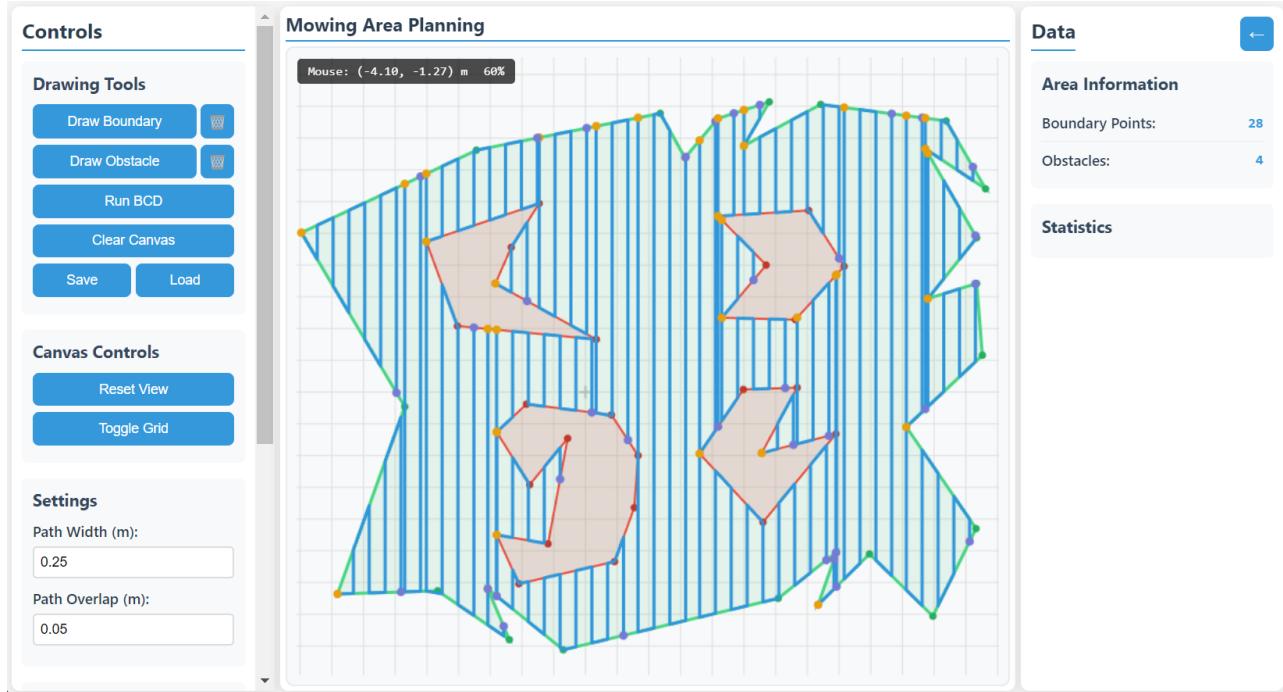
Kodas pasiekiamas: [14].

4.2. Diegimas

Įrankis paleidžiamas naudojantis viena „Make“ komanda, po kurios sistema lokaliai paleidžiama naudotojo kompiuteryje. Dėl pasirinktos žiniatinklio platformos nereikia atlikti diegimo procedūrų, paleidimas sklandus ir greitas.

4.3. Testavimas

Prototipas buvo ištestuotas, išbandytas su keliomis skirtingomis aplinkomis ir vienu algoritmu. Bandymai naudotis sistema parodė, kad sistema yra naudinga dirbant su ZPKP algoritmais ir jos pilnas įgyvendinimas yra prasmingas ir įmanomas.



60 pav. Sistemos prototipas

5. Rezultatai ir apibendrinimas

Mokomosios praktikos metu buvo susipažinta su zonas padengimo kelio planavimo algoritmais, jų principais, privalumais ir trūkumais. Atlikta esančių atvirojo kodo ZPKP sprendimų analizė parodė, kad nors ir egzistuoja, kai kurios bibliotekos, jos yra sudėtingai naudojamos, reikalauja specifinių techninių žinių ir dažnai apsiriboja siauromis taikymo sritimis.

Praktikos metu buvo identifikuoti funkciniai ir nefunkciniai reikalavimai internetinei simuliacijos įrangos sąsajai. Suprojektuota sistema, kuri leistų tyrėjams interaktyviai kurti aplinkas su zonomis ir kliūtimis, konfigūruoti algoritmų parametrus, vykdyti tiek pavienius, tiek pakartotinius ZPKP testavimo scenarijus bei analizuoti gautus rezultatus. Parengta detaliai aprašyta techninė specifikacija, apimanti 24 funkcinius reikalavimus, vartotojo sąsajos schemą ir loginę duomenų bazės struktūrą.

Įgyvendinimo etape buvo sukurtas veikiantis sistemos prototipas. Prototipas sėkmingai išbandytas su keliomis skirtingomis aplinkomis ir vienu algoritmu, patvirtinant sistemos funkcionalumą ir naudingumą.

Praktikos metu įgyta patirtis projektavimo ir programavimo srityse. Plėtotos analitinės kompetencijos, atliekant algoritmų palyginimą ir identifikuojant sistemai keliamus reikalavimus.

ZPKP tema toliau bus atliekama profesinė praktika, siekiant realizuoti įrankį pilnai tenkinantį techninę specifikaciją ir pradėti realizuoti kitus ZPKP algoritmus.

Literatūra

- [1] Z. L. Cao, Y. Huang, E. L. Hall. „Region filling operations with random obstacle avoidance for mobile robotics“. Iš: *Journal of Robotic Systems* 5.2 (1988), puslapiai 87–102.
- [2] Y. Chen, Z.-M. Lu, J.-L. Cui, H. Luo, Y.-M. Zheng. „A Complete Coverage Path Planning Algorithm for Lawn Mowing Robots Based on Deep Reinforcement Learning“. Iš: *Sensors* 25.2 (2025). ISSN: 1424-8220. <https://doi.org/10.3390/s25020416>. URL: <https://www.mdpi.com/1424-8220/25/2/416>.
- [3] H. Choset. „Coverage for robotics—a survey of recent results“. Iš: *Annals of Mathematics and Artificial Intelligence* 31.1 (2001), puslapiai 113–126. URL: <https://link.springer.com/content/pdf/10.1023/A:1016639210559.pdf>.
- [4] H. Choset, P. Pignon. „Coverage path planning: The boustrophedon cellular decomposition“. Iš: *Field and Service Robotics*. Springer, 1998, puslapiai 203–209. URL: https://www.ri.cmu.edu/pub_files/pub4/choset_howie_1997_3/choset_howie_1997_3.pdf.
- [5] J. Colegrave, A. Branch. „A Case Study of Autonomous Household Vacuum Cleaner“. Iš: *AIAA/NASA CIRFFSS*. 1994.
- [6] E. Galceran, M. Carreras. „A survey on coverage path planning for robotics“. Iš: *Robotics and Autonomous Systems* 61.12 (2013), puslapiai 1258–1276. URL: <https://doi.org/10.1016/j.robot.2013.09.004>.
- [7] M. Höffmann, S. Patel, C. Büskens. „Optimal guidance track generation for precision agriculture: A review of coverage path planning techniques“. Iš: *Journal of Field Robotics* 41.3 (2024), puslapiai 823–844. URL: <https://doi.org/10.1002/rob.22286>.
- [8] A. Khan, I. Noreen, H. Ryu, N. L. Doh, Z. Habib. „Online complete coverage path planning using two-way proximity search“. Iš: *Intelligent Service Robotics* 10.3 (2017), puslapiai 229–240. URL: <https://link.springer.com/article/10.1007/s11370-017-0223-z>.
- [9] J. C. Latombe. *Robot Motion Planning*. Tomas 124. Springer Science & Business Media, 2012. URL: https://books.google.lt/books?hl=en&lr=&id=nQ7aBwAAQBAJ&oi=fnd&pg=PR9&dq=Robot+Motion+Planning+Jean&redir_esc=y.
- [10] Y. Li, H. Chen, M. J. Er, X. Wang. „Coverage path planning for UAVs based on enhanced exact cellular decomposition method“. Iš: *Mechatronics* 21.5 (2011), puslapiai 876–885. URL: <https://doi.org/10.1016/j.mechatronics.2010.10.009>.
- [11] G. Mier. *Demo image*. URL: https://github.com/Fields2Cover/Fields2Cover/blob/main/docs/figures/demo_image.png.
- [12] G. Mier, J. Valente, S. de Bruin. „Fields2Cover: An Open-Source Coverage Path Planning Library for Unmanned Agricultural Vehicles“. Iš: *IEEE Robotics and Automation Letters* 8.4 (2023), puslapiai 2166–2172. <https://doi.org/10.1109/LRA.2023.3248439>.

- [13] L. D. Nielsen, I. Sung, P. Nielsen. „Convex Decomposition for a Coverage Path Planning for Autonomous Vehicles: Interior Extension of Edges“. Iš: *Sensors* 19.19 (2019), puslapis 4165. URL: <https://doi.org/10.3390/s19194165>.
- [14] J. Remeikis. *ZPKP internetinės sasajos prototipas*. 2025. URL: <https://github.com/jRmx0/BladeOfGrass-coveragePlanning> (žiūrėta 2025-11-05).
- [15] N. Rottmann, R. Denz, R. Bruder, E. Rueckert. „A probabilistic approach for complete coverage path planning with low-cost systems“. Iš: *Proc. 2021 European Conf. Mobile Robots (ECMR)*. 2021, puslapiai 1–8. URL: <https://ieeexplore.ieee.org/document/9568847>.
- [16] N. Shah, U. Dey, K. Nishimiya. *End-to-End Framework for Robot Lawnmower Coverage Path Planning using Cellular Decomposition*. arXiv preprint arXiv:2506.06028. 2025. URL: <https://arxiv.org/pdf/2506.06028.pdf> (žiūrėta 2026-01-04).
- [17] Z. Shen, J. P. Wilson, S. Gupta. *C*: A Coverage Path Planning Algorithm for Unknown Environments using Rapidly Covering Graphs*. 2025. URL: <https://arxiv.org/abs/2505.13782>.
- [18] C. S. Tan, R. Mohd-Mokhtar, M. R. Arshad. „A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms“. Iš: *IEEE Access* 9 (2021), puslapiai 119310–119342. URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9523743>.
- [19] C. M. Witkowski. „A parallel processor algorithm for robot route planning“. Iš: *Proc. Eighth Int. Joint Conf. on Artificial Intelligence (IJCAI)*. Tomas 2. 1983, puslapiai 827–829. URL: <https://www.ijcai.org/Proceedings/83-2/Papers/055.pdf>.