KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Programų inžinerijos katedra

Mantas Smilingis

Eksperimentavimo sistema funkcinio vėlinimo testo generavimo metodui kurti

Magistro darbas

Darbo vadovas:

Prof. habil. dr. Rimantas Šeinauskas

KAUNAS, 2010 m.

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Programų inžinerijos katedra

Mantas Smilingis

Eksperimentavimo sistema funkcinio vėlinimo testo generavimo metodui kurti

Magistro darbas

Recenzentas:

Dr. Kęstutis Paulikas

2010-05-31

Darbo vadovas:

Prof. habil. dr. Rimantas Šeinauskas

2010-05-31

Atliko

IFM-4/4 gr. stud.

Mantas Smilingis

2010-05-31

KAUNAS, 2010 m.

Experimenting system for developing functional delay test generating method

Summary

As sizes of circuits are shrinking and the clock speed is increasing, functional delay tests are receiving more attention within each day. Based on circuits "black-box" behavioural description, it is possible to generate a high quality functional delay test at initial circuits design stages, which covers more than 99% of all circuits' transition faults. To make more reliable and speed up the process of finding a better functional delay test generating method, an experimenting system was developed. One of the system`s main features is its capability of automatically storing data files to their set destination directories, which lowers the risk of human being error-prone.

This thesis analyses methods used for creating functional delay tests and suggests four more implementations using activity vectors. By using partly automated experimenting system, all four methods were tested, together with the experimenting system. The results reveal how much the functional delay testing process speed up using automated system compared to launching all the testing tasks personally. As experimenting system automatically creates functional delay test method comparison table, it was possible to easily evaluate the most acceptable method that produces tests with high fault coverage and low test size.

Keywords: Functional Delay Test Generation, "Black-Box" Testing, Behavioural Testing, Experimenting System, Automated System, Data Management, Content Management, Transition Fault Coverage

Turinys

Sutarti vardai ir apibrėžimai (terminų žodynas)

1. ATPG (angl. **A**utomatic **T**est **P**attern **G**eneration ir **A**utomatic **T**est **P**attern **G**enerator) – elektroninis automatizuotas modelis ar technologija, naudojama surasti įėjimo signalų rinkinį (arba testą), kurį galima pritaikyti skaitmeninei schemai. Tokiu būdu testuojant nustatoma ar schema veikia gerai, ar klaidingą veikimą išprovokavo atsiradusi klaida.
2. Aktyvumo vektorius – schemos įėjimo signalo simbolių seka, apibūdinanti tam tikro išėjimo signalo pasikeitimą įtakojančius įėjimo signalus.
3. Aktyvios reikšmės – sutartai pažymėtos aktyvumo vektoriaus reikšmės, kurių keitimas įtakoja signalo pasikeitimą analizuojamame išėjime.
4. Pasikliautinumas – tam tikromis taisyklėmis nustatytas žymėjimas aktyvumo vektorių paieškos sėkmingumui apibrėžti.
5. SIT (angl. **S**ingle-**I**nput **T**ransition) testas – simbolių sekų pora, kurioje lygiai vieno įėjimo signalo pasikeitimas sužadina perėjimą *[12]*.
6. MIT (angl. **M**ulti-**I**nput **T**ransition) testas – simbolių sekų pora, kurioje daugiau negu vieno įėjimo signalo pasikeitimas sužadina perėjimą *[12]*.
7. Klaidų simuliacija (angl. fault simulation) – tai procesas, kuriuo siekiama nustatyti loginės grandinės išvestis, pateikus testinį sekų bei klaidų rinkinius. Klaidų simuliacija naudojama, norint nustatyti ar klaidos yra aptinkamos, ar ne.
8. Klaidų padengimas (angl. fault coverage) – tai santykio tarp aptiktų bei visų klaidų procentinė išraiška *[13]*.
9. DRC (angl. **D**esign **R**ule **C**hecking) – patikrinimas ar konkrečios schemos fizinis išdėstymas tenkina rekomenduojamų parametrų sąrašą.
10. Veiklos Procesų Automatizavimas (angl. Business Process Automation) - tai programinės įrangos panaudojimas automatizuojant užduotis, siekiant sumažinti išlaidas (finansines, laiko ar kitas).
11. Automatizavimas arba automatizacija - procesų arba įrenginių transformacija į automatinį veikimo būdą.
12. Turinio tvarkymo sistema (angl. **C**ontent **M**anagement **S**ystem) - įvairūs programiniai įrankiai, supaprastinantys informacinių sistemų turinio (tekstinio ir grafinio) valdymą taip, kad sukuriant bei keičiant turinį ar jo struktūrą nereikėtų jokių specialiųjų (programavimo) žinių.
13. WYSIWYG (angl. **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et) – pasakymas, naudojamas

kalbant apie kompiuterines programas, norint apibūdinti kokią nors programinę įrangą, kurios turinio išvaizda rengimo metu yra labai panaši į tos programinės įrangos pateikiamo galutinio produkto išvaizdą

14. RSS (angl. **R**eally **S**imple **S**yndication) - XML failų formatų šeima internetiniam duomenų rinkimui iš naujienų tinklų ir tinklaraščių.
15. Kešavimas (angl. Caching) – tam tikras optimizavimo procesas, kurio metu dažniau naudojami duomenų elementai daromi lengviau pasiekimais, nei tie, kurie naudojami retai.
16. SSL (angl. **S**ecure **S**ockets **L**ayer) - kriptografinis protokolas, skirtas informacijos, sklindančios internete apsaugojimui šifruojant.
17. Captcha (angl. **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part) – testas, naudojamas kompiuteriuose ir skirtas nustatyti ar naudotojas yra žmogus, ar ne.

1. Įvadas

Sparčiai vystantis technologijoms, mažėjant jų matmenims ir augant greičiams bei taktiniam dažniui, funkcinio vėlinimo testai vis daugiau dėmesio sulaukia iš mikroschemų gamyba užsiimančių įmonių, norinčių pagerinti testų kokybę klaidoms aptikti.

Dažniausiai testai generuojami susintezavus įrenginį, tokiu būdu prailginant bendrą įrenginio paruošimo laiką. Tačiau generuojant testus ankstyvoje įrenginio sintezavimo stadijoje, galima sumažinti generavimo laiką po sintezės bei įrenginio eksploatacijos kaštus *[1, 2, 4, 14]*. Testų gamybos rinkoje galima aptikti įvairių programinių sistemų siūlančių sprendimus funkcinio vėlinimo testams generuoti. Pritaikomi sprendimai ir metodai atskleidžia testų generavimo, simuliuojant schemos veikimą, perspektyvas *[2, 14]*. Siūlomais metodais sugeneruoti testai nėra visais atvejais išsamūs ir nepadengia visų testuojamų klaidų *[3]*. Sugeneruoti funkcinio vėlinimo testai vidutiniškai aptinka iki 99% signalo perdavimo klaidų *[14]*. Tai skatina atlikti daugiau tyrimų bei eksperimentų šioje srityje.

Turint programinį įrenginio veikimą aprašantį prototipą, galima verifikuoti ir analizuoti įrenginio elgseną. Pateikus įvesties duomenis prototipo veikimo simuliacijos sprendimams - gaunami apskaičiuoti išeigos rezultatai. Pritaikius gautiems duomenims testų sudarymo algoritmus, sudaromos įvairios testinės sekos. Augant poreikiui sudaryti daugiau klaidų aptinkančius testus, atliekamos prielaidos kuriamiems, kombinuojamiems bei modifikuojamiems testų sudarymo metodams. Norint pateisinti arba paneigti naujai sudaryto metodo gerumą, reikia atlikti daug eksperimentų, kurių metu naudojami tarpiniai duomenys sudaro galimybę įvelti žmogiškų klaidų, nulemiančių galutinius rezultatus. Sudarius veikiančią automatizuotą sistemą, kuri padėtų atlikti eksperimentus, būtų galima šių klaidų išvengti, greičiau suformuoti testų rezultatus, stebėti testavimo procesą, o su gaunamais rezultatais įvertinti metodų gerumą.

Magistrinio darbo tikslas yra pagerinti funkcinio vėlinimo testų generavimo metodą, kuris remiasi programiniu schemos modeliu. Norint pasiekti tikslių rezultatų per trumpą laiką, reikia pagreitinti funkcinio vėlinimo testų generavimo bei testavimo eksperimentų atlikimo procesą, sukuriant sistemą, leidžiančią intuityviai peržiūrėti eksperimentų rezultatus bei įvertinti sudarytų ir kombinuojamų metodų gerumą, kuris matuojamas signalo perdavimo klaidų padengimu, išreikštu procentais.

2. Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti analizė

2.1. Analizės tikslas

Informacinėje visuomenėje, esant intensyviems sprendimų įgyvendinimams, galima rasti keletą siūlomų sistemų funkcinio vėlinimo testams generuoti bei atvaizduoti. Tačiau testų kūrimas dažniausiai apsiriboja ties vienu testuojamo objekto aprašu. Esant poreikiui atlikti daug testų susijusiu su kiekvienu schemos modeliu, periodiškai atsiranda laiko praradimas, sueikvojamas kiekvieno veiklos etapo pasiruošimui.

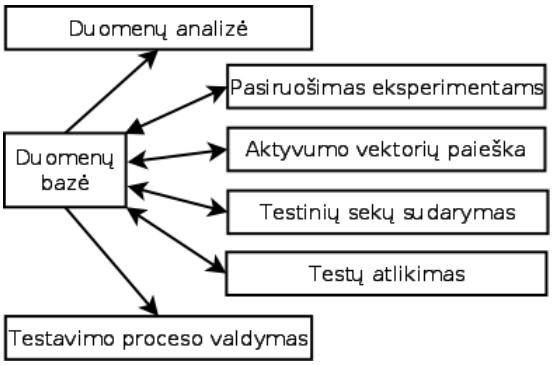
Analizės tikslas yra apžvelgti bei įvertinti esamus funkcinio vėlinimo testų generavimo metodus ir dinamines turinio, informacijos atvaizdavimo sistemas, ir jų sprendimų panaudojimo galimybę kuriamoje sistemoje, siekiant greitesnio ir patogesnio eksperimentų proceso atlikimo bei intuityvios sąsajos generuojamiems rezultatams analizuoti.

2.2. Tyrimo sritis, objektas ir problema

2.2.1. Tyrimo sritis

Eksperimentavimo sistemos kuriamos kitų sistemų, tokių kaip: turinio, failų arba duomenų valdymo, portalų pagrindu, sukuriant palaikomus komponentus, pritaikant reikalingas savybes turinčius elementus ar kodo atviro kodo fragmentus. Pasirinkimas, kokią sistemą laikyti tėvine, vystant eksperimentavimo sistemą yra labai platus ir priklauso nuo užsakovo poreikių bei reikalavimų sistemai. Kartais gali tėvinės sistemos neturėti reikiamo funkcionalumo, dėl to gali tekti arba taisyti esamą sistemą, arba pasirašyti savo.

Eksperimentavimo sistema leistų centralizuoti su testavimu susijusių procesų valdymą: pasiruošimo eksperimentui etapą, aktyvumo vektorių paiešką, testinių sekų sudarymą, testų simuliaciją ir rezultatų analizę bei atvaizdavimą (1 pav. Eksperimentavimo sistemos koncepcija).



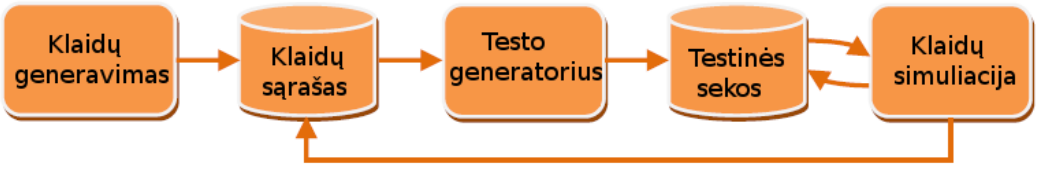
1 pav. Eksperimentavimo sistemos koncepcija

Eksperimentavimo proceso funkcinio vėlinimo testams atlikti automatizavimas apima dvi sritis:

- Procesų automatizavimą. Siunčiama užklausa atlikti norimą veiklos etapą ir pagal gautą komandą automatiškai įvykdomi su veiklos etapu susijusių integruojamų procesų iškvietimai. Siekiama, kad veiklos etapus būtų galima kontroliuoti, o integruotų procesų rezultatai būtų automatiškai tvarkomi ir keliami į jiems skirtus katalogus.
- Testavimą. Paduodama įėjimo poveikių seka ir pagal reakcijas žiūrima ar objektas neturi klaidų arba gedimų. Tam tikslui reikia žinoti teisingas reakcijas. Siekiama, kad signalo perdavimo klaidų padengimas būtų kuo didesnis, o įėjimo poveikių seka būtų kiek galima trumpesnė.

2.2.2. Tyrimo objektas

Pažvelgus į įprastą automatinio testinių sekų generatoriaus (ATPG) veiklą, kuri susideda iš klaidų sąrašo generavimo, testinių sekų sudarymo ir simuliacijos (2 pav. Įprasta ATPG veiklos seka), matome, jog numatoma eksperimentavimo sistema dalinai persidengia su ATPG veiklos sekomis, apimdama testų generavimą bei klaidų simuliacijos etapus. Testo generavimo etapas yra sudėtinis, t.y. apima aktyvumo vektorių paiešką ir testinių sekų sudarymą. Taip pat yra žinoma, jog klaidų padengimo simuliacija bus atliekama plačiai paplitusiu Synposys TetraMAX paketu.



2 pav. Įprasta ATPG veiklos seka *[7]*

Norima automatizuoti testų generavimo ir testavimo procesą, stebėti šio proceso eigos rezultatus, ir juos tvarkingai talpinti saugykloje. Tyrimo objektai yra skaitmeninių įtaisų funkcinio vėlinimo testų generavimo metodai, kurie naudoja programinį schemos modelį bei dinaminio turinio ir informacijos atvaizdavimo sistemos.

Testuojamu objektu gali būti programa arba aparatūra. Pirmuoju atveju ieškoma sistemoje klaidų, o antruoju atveju eksploatacijos arba gamybos gedimų. Nagrinėjamas antras atvejis. Pagrindinis apribojimas, jog negalime matuoti vidinių aparatūros reikšmių, galima matuoti tik išėjimus.

2.2.3. Tyrimo problemos bei numatomi sprendimai

Išskiriamos šios problemos, susijusios su eksperimentavimo sistema funkcinio vėlinimo testo generavimo metodui kurti:

Esami funkcinio vėlinimo testo generavimo metodai nėra pakankamai kokybiški, todėl panaudojus šiuos metodus, sukurti testai nepakankamai aptinka signalo perdavimo klaidų *[3]*. Užsakovas pageidauja rasti geresnį metodą, kuriuo remiantis, būtų galima sudaryti funkcinio vėlinimo testus, padengiančius didesnį kiekį signalo perdavimo klaidų.

Atliekant daug eksperimentų gaunama daug rezultatų failų, kurių kiekvieną atskirai reikia užsakovui analizuoti ir pagal kokią nors susigalvotą sistemą pačiam sudėlioti į katalogus. Norima patogesnės sąsajos, kurios dėka būtų galima iš karto peržiūrėti norimo eksperimento schemų rezultatų ir tarpinių duomenų failus, grupuojamus pagal kokią nors sistemą.

Kadangi testuojamų ISCAS'85 schemų pavadinimai labai panašūs, skiriasi tik raide ir skaitmenimis, nurodant su šiais pavadinimais susijusius duomenų failus, yra tikimybė nurodyti netinkamą rezultatams laikyti duomenų failą arba netinkamą duomenų failą. Automatizavus dalį procesų, būtų galima išvengti galimų žmogiškųjų klaidų *[9, 10, 11]*, pavyzdžiui: suklydus perrašyti vieną failą kitu ir gauti klaidingus rezultatus.

Neautomatizuotos veiklos atveju, kiekvienai testuojamai schemai nurodant reikalingus duomenų failus prarandama laiko. Kad surasti geresnį funkcinio vėlinimo generavimo metodą, reikia atlikti daug eksperimentų, kurių metu galima įvelti klaidų. Atsiranda tikimybė, jog nurodžius neteisingą duomenų failą, gausime klaidingą vieno

iš eksperimentų rezultatą, dėl ko turėsime pakartoti eksperimentą. Automatizavus bent dalį procesų, būtų galima išvengti laiko nuostolio ir galbūt net sutaupyti laiko *[10, 11]*.

Aukščiau išvardintoms problemoms numatomi sprendimai:

- Planuojama išanalizuoti funkcinio vėlinimo testų kūrimo metodus. ir pasirinktais metodais atlikti eksperimentus. Atlikus eksperimentus, įvertinti metodų gerumą.
- Sukurti vidinės įmonės portalo veikimo principu paremtą eksperimentavimo sistemą funkcinio vėlinimo testo generavimo metodui kurti, kurios pagalba būtų galima dalinai automatizuoti eksperimentų atlikimą.
- Kad išvengti dalies eksperimento metu galimų įvelti klaidų dirbant su duomenų failais, įdiegta sistemos funkcija leistų gaunamus rezultatus bei tarpinius duomenų failus automatiškai pagal pasirinktą vieningą sistemą suskirstytų į katalogus bei vieno mygtuko paspaudimu pašalinti kurio nors eksperimento schemos norimus tarpinius duomenis ar rezultatus.
- Taip pat sistema turėtų patikrinti galutinį testavimo proceso rezultatų failą, t.y. nustatyti testo dydį bei nustatyti gautą signalų perdavimo klaidų padengimą, išskiriant geriausius rezultatus, kuriais remiantis būtų galima padaryti išvadas apie panaudotų metodų gerumą.

2.3. Tyrimo tikslai ir uždaviniai

2.3.1. Tyrimo tikslai

Vienas iš tyrimo tikslų yra išnagrinėti funkcinio vėlinimo testų sudarymo pagal programinį schemos prototipą metodus. Bendradarbiaujant su srities specialistais suformuluoti daugiau signalo perdavimo klaidų padengiančių testų sudarymo metodą, kurį eksperimento metu galėtume pritaikyti.

Eksperimentavimo etapai dalinai automatizuojami apjungiant už jų procesus atsakingas programines dalis. Apjungti galima su parinkta arba naujai sukurta sistemą. Kadangi biudžetas ribotas, iškyla dar vienas tyrimo tikslas: ištirti portalų, dinaminio turinio ir informacijos tvarkymo sistemų galimybes bei parinkti eksperimentavimo proceso integracijai tinkamą sistemą.

Tuo atveju, jei nepavyktų atrasti tinkamos sistemos – patiems suprojektuoti sistemos modelį, kurį realizavus būtų galima, paspartinti veiklos proceso, t.y.

kokybiškesnio funkcinio vėlinimo testų generavimo metodo nustatymo procesą, stebėti ir analizuoti vėlinimo testų generavimo procesą, proceso rezultatus, ir rezultatų kokybę.

2.3.2. Tyrimo uždaviniai

Išskiriami šie tyrimo uždaviniai:

- Išsiaiškinti:
  - kaip gauti aktyvumo vektorius iš programinio įrenginio modelio;
  - kaip gauti funkcinio vėlinimo testus iš aktyvumo vektorių;
  - kas turėtų sudaryti sistemą funkcinio vėlinimo testo generavimo metodui kurti.
- Išanalizuoti:
  - jau esamus funkcinio vėlinimo testų generavimo metodus;
  - atviro kodo turiniui bei informacijai atvaizduoti sistemas ir jų funkcionalumą.
- Nustatyti, kaip galime panaudoti atrastą informaciją.
- Įvertinti galimas automatizuoti sistemos dalis.
- Suformuluoti daugiau signalo perdavimo klaidų padengiančio funkcinio vėlinimo testų generavimo metodą.

Pagrindinis sprendžiamas uždavinys – testinės sekos, tikrinančios visus gedimus, suradimas. Gedimai gali būti dviejų tipų – statiniai ir dinaminiai. Nagrinėjami bus dinaminiai gedimai. Dinaminių gedimų tikrinimui naudojamos įėjimo poveikių poros, kur pirmas poveikis nustato signalo reikšmes, o antras poveikis bando kai kurias iš jų keisti ir iš karto po antro poveikio matuojami išėjimai siekiant nustatyti ar išėjimai pasikeitė taip kaip turėjo.

Testinės sekos gali būti skaičiuojamos pagal susintezuotos schemos modelį arba pagal programinio prototipo modelį. Pirmuoju atveju gaunama optimizuota testinė seka konkrečiai schemai, bet testinės sekos generavimo laikas didina projektavimo trukmę ir paruošimo naudojimui laiką. Testinės sekos suradimo laikas turi būti kiek galima trumpesnis. Antruoju atveju testinė seka skaičiuojama nežinant konkrečios realizacijos, ji gaunama ilgesnė, sunkiai užtikrina visų gedimų patikrinimą, bet gali būti skaičiuojama lygiagrečiai su schemos projektavimo – sintezavimo veiksmais neilginant paruošimo naudojimui laiko. Tyrimai vystomi pagal antro atvejo kryptį.

2.4. Analizės metodai

Pasitelkus interneto paieškos sistemomis ir bendraujant su testavimo srities specialistais, atliekama siūlomų funkcinio vėlinimo testų sudarymo metodų paieška ir analizė. Surasti metodų rezultatai patalpinami į palyginimų lentelę, kurioje lyginame esamų metodų suformuotų testų rezultatus, atsižvelgdami į perdavimo klaidų padengimo koeficiento reikšmę bei sugeneruotų testų dydį. Taip pat palygimo lentelė sudaroma surastų atviro kodo turinio ir informacijos atvaizdavimo sistemų funkcionalumui palyginti. Siekiama surasti sistemą, kurią būtų galima pritaikyti proceso etapų automatizavimui.

2.5. Funkcinio vėlinimo testo generavimo analizė

2.5.1. Funkcinio vėlinimo klaidų analizė

Iš pirmo žvilgsnio schema gali veikti nepriekaištingai, tačiau kintant jos greičiui, jos veikla gali sutrikti. Tokio tipo klaidos vadinamos vėlinimo klaidomis. Kad patikrinti ar schema veikia teisingai norimame greityje, generuojami vėlinimo testai. Konstantinių gedimų testai gali aptikti dalį vėlinimo klaidų, tačiau schemų gamybos industrijoje vien tik šių testų jau neužtenka, dėl to išauga poreikis naudoti ir vėlinimo testavimą.

Yra išvystyti trys klasikiniai klaidų modeliai, atstovaujantys vėlinimo klaidas: signalo perdavimo klaidos (angl. „transition fault“), ventilių vėlinimo (angl. „gate delay fault“) bei kelio vėlinimo klaidos (angl. „path delay fault“). Be šių klasikinių yra dar ir segmento vėlinimo klaidos. Visi modeliai turi savo trūkumų ir pranašumų. Didelio delsimo schemos klaidoms aptikti signalo perdavimo klaidų modelis laikomas labiau efektyviu už kitus, o kelio vėlinimo klaidų modelis – trumpo vėlinimo klaidoms *[4]*.

E. Bareišos siūlyto „kojų-poros“ (angl. „PIN-pair“ - PP) klaidų modelio perspektyvas galima įžvelgti analizuojant funkcinio vėlinimo testų generavimą, tikrinantį konstantinius (angl. „stuck-at“) gedimus *[2]*, kuomet neaptiktų klaidų vidurkis neviršijo 0.5% testuojamų ISCAS’85 schemų modelių. Kadangi siūlomame modelyje neanalizuojama schemos sudėtis žemame lygmenyje, ir testai būna pagrįsti tiktai įėjimo signalų rinkiniais bei išėjimų į juos reakcijomis – remsimės šiuo klaidų modeliu, analizuojant vėlinimo testų sudarymo metodus.

Funkcinio vėlinimo klaida žymima (*I*, *O*, *t I*, *t O*), kur simbolis I reiškia schemos įėjimą, O – schemos išėjimą, t – kylantis (r) arba krentantis (f) signalas įėjime arba išėjime. Remdamiesi PP klaidų modeliu sugeneruotais funkcinio vėlinimo testais tikrinsime signalo perdavimo klaidų padengimą.

Pasak analizuojamų straipsnių autorių, norint aptikti vėlinimo klaidą, reikia dviejų testinių sekų <u, v>. Vėlinimo testo įėjimo signalų sekas gali sudaryti vieno arba kelių įėjimų pasikeitimai, tačiau klaidą gali aptikti tik tie pasikeitimai įėjimuose, kurie lemia pasikeitimą išėjimuose *[4, 14]*. Kadangi testiniai vektoriai sudaromi remiantis „juodos dėžės“ modeliu, pirmiausiai jį ir analizuosime.

2.5.2. „Juodosios dėžės“ modelio analizė

Elgsenos modelis arba „juodoji dėžė“, tai sistemos arba įrenginio prototipas, kurio vidinė struktūrą yra nežinoma. Tokiame prototipe apibrėžtas schemos elgesys, atsižvelgus į reikšmes, kurias pateikiame įėjimams. Iš programinio schemos prototipo galime nustatyti tik įėjimų bei išėjimų sąryšį, t.y. kokie įėjimai turi sąryšį su išėjimais bei nuo kokių įėjimų priklauso išėjimai *[2, 4, 14]*. Remiantis šia informacija, analizuojamos įėjimo signalų sekos.

2.5.3. „Kojelių-poros“ klaidų modelio analizė

Tarkime turime „juodos dėžės“ modelį, kurios įėjimų aibė $X = \{x_1, x_2, ..., x_i, ... ,x_n\}$ ir išėjimų aibę $Z = \{z_1, z_2, ... ,z_j, ... ,z_m\}$. Pagal PP modelį, konstantinės 0 bei 1 klaidos įvyksta modelio kraštuose ir turi nežymią koreliaciją su schemos fiziniais gedimais. Įėjimų konstantinius 1 ir 0 gedimus žymėsime x_i^1 ir x_i^0 iš išėjimo – z_j^1 ir z_j^0. Iš viso yra 2*n + 2*m galimų kojelių gedimų. Įėjimo – išėjimo kojelių poros konstantinis gedimas (x_i^t, z_j^k), t=0,1, k=0,1 yra vadinamas „kojelių-poros“ gedimu *[2, 4, 15]*. Daugiausiai schemoje PP klaidų gali būti 4*n*m, kur n – įėjimo signalų skaičius ir m – išėjimo signalų skaičius.

Funkcinio vėlinimo testui sudaryti iš PP klaidų testo yra taikomos kelios taisyklės, kurios pateiktos žemiau esančioje lentelėje (1 lentelė. Taisyklės funkcinio vėlinimo testui iš PP klaidų testo sudarymui).

1 lentelė. Taisyklės funkcinio vėlinimo testui iš PP klaidų testo sudarymui **[4, 15]**

Taisyklės nr.	**Taisyklė**
1.	Jei įėjimo signalų seka q aptinka PP klaidą (x_i^t, z_j^k), t=0,1, k=0,1, tai įėjimo signalo sekų pora **<p, q>** aptinka funkcinio vėlinimo klaidą (x_i, z_j, **r(f) x_i**, tr

	z_j), tr = r,f. Signalo x_i reikšmė sekoje q yra 1 (0), o x_i signalo reikšmė sekoje p yra 0 (1).
2.	Jei įėjimo signalų seka q aptinka PP klaidą (x_i^t, z_j^k), t=0,1, k=0,1, tai įėjimo signalo sekų pora **<q, p>** aptinka funkcinio vėlinimo klaidą (x_i, z_j, **f(r) x_i**, tr z_j), tr = r,f. Signalo x_i reikšmė sekoje q yra 1 (0), o x_i signalo reikšmė sekoje p yra 0 (1).
3.	Jei įėjimo signalų seka q aptinka PP klaidą (x_i^t, z_j^k), ..., (x_g^t, z_d^k), ..., (x_h^t, z_f^k), t=0,1, k=0,1, tai įėjimo signalo sekų pora **<p, q>** gali aptikti šias funkcinio vėlinimo klaidas: klaidą (x_i, z_j, **r(f) x_i**, tr z_j), ..., (x_g, z_d, **r(f) x_g**, tr z_d), ..., (x_h, z_f, **r(f) x_h**, tr z_f), tr = r,f. Signalų $x_{i, \dots, } x_g$, ..., x_h reikšmės sekoje q yra 1 (0), o $x_{i, \dots, } x_g$, ..., x_h signalų reikšmė sekoje p yra 0 (1).
4.	Jei įėjimo signalų seka q aptinka PP klaidą (x_i^t, z_j^k), ..., (x_g^t, z_d^k), ..., (x_h^t, z_f^k), t=0,1, k=0,1, tai įėjimo signalo sekų pora **<q, p>** gali aptikti šias funkcinio vėlinimo klaidas: klaidą (x_i, z_j, **f(r) x_i**, tr z_j), ..., (x_g, z_d, **f(r) x_g**, tr z_d), ..., (x_h, z_f, **f(r) x_h**, tr z_f), tr = r,f. Signalų $x_{i, \dots, } x_g$, ..., x_h reikšmės sekoje q yra 1 (0), o $x_{i, \dots, } x_g$, ..., x_h signalų reikšmė sekoje p yra 0 (1).

Pirma ir antra taisyklės naudojamos sudaryti SIT testus, tuo tarpu trečia ir ketvirta – MIT.

2.5.4. Funkcinio vėlinimo klaidų testų iš PP modelio sudarymo metodų analizė

Funkcinio vėlinimo testai išvedami iš PP klaidų testų, pagal keturias lentelėje aprašytas taisykles (1 lentelė. Taisyklės funkcinio vėlinimo testui iš PP klaidų testo sudarymui). Sudaryti testai aptinka signalo perdavimo klaidas ventilių lygmenyje.

Testai, kurių dėka kiekvieną klaidą f aptinka n skirtingų įėjimo signalų rinkinių arba daugiausiai galimų įėjimo signalo rinkinių (jei klaida f turi mažiau, negu n skirtingų įėjimo signalo rinkinių, galinčių ją aptikti), vadinami „n – aptikimo“ testais. O tie testai, kurių dėka visi gedimai aptinkami vieną kartą, vadinami „1 – aptikimo“.

Antroje lentelėje („1 - aptikimo“ funkcinio vėlinimo kūrimo metodai) pateikti „1 – aptikimo“ funkcinio vėlinimo kūrimo metodai, o 2 lentelėje („n - aptikimo“ funkcinio vėlinimo kūrimo metodai) pateikti „n - aptikimo“ funkcinio vėlinimo kūrimo metodai, sudaryti pagal ankščiau lentelėje (1 lentelė. Taisyklės funkcinio vėlinimo testui iš PP klaidų testo sudarymui **[4, 15]**) aprašytas taisykles.

2 lentelė. „1 - aptikimo“ funkcinio vėlinimo kūrimo metodai **[4, 14]**

Metodo nr.	Metodo aprašymas
M1.	PP testai yra transformuojami į funkcinio vėlinimo testus, pagal ankščiau lentelėje (1 lentelė. Taisyklės funkcinio vėlinimo testui iš PP klaidų testo sudarymui **[4, 15]**) pateiktą pirmą taisyklę. Sudarytų testinių porų rinkiniai, atspindi vienos reikšmės pasikeitimą viename įėjime, todėl jie vadinami SIT testais
M2.	PP testai yra transformuojami į funkcinio vėlinimo testus, pagal ankščiau lentelėje (1 lentelė. Taisyklės funkcinio vėlinimo testui iš PP klaidų testo sudarymui **[4, 15]**) pateiktą trečią taisyklę. Šiuo metodu sudarytų testinių porų rinkiniai atspindi signalų pasikeitimą keliuose įėjimuose, ir yra vadinami MIT testais. Remiantis šiuo metodu, galima neaptikti keleto funkcinio vėlinimo klaidų, nes signalo sklidimui iš numatomų įėjimų į išėjimus gali reikėti papildomų aktyvuojančių sąlygų. Dėl šios priežasties testas gali neaptikti 100% analizuojamų klaidų.
M3.	Funkcinio vėlinimo testas transformuojamas tokiu pačiu būdu, kaip ir aprašytu antro metodo atveju. Skirtumas yra tik toks, kad sudaryto testo porų sekų eiliškumas yra sukeistas vietomis, t.y. jei antrame metode tam tikrai PP testo sekai buvo sugeneruota pora <u, v>, tai šiuo metodu sugeneruota testinė pora yra <v, u>.

3 lentelė. „n - aptikimo“ funkcinio vėlinimo kūrimo metodai **[4, 14]**

Metodo nr.	Metodo aprašymas
M4.	Du atskiri PP testai sugeneruoti ir transformuoti pagal 1 taisyklę. Apjungus, gautas „2 aptikimų“ funkcinio vėlinimo SIT testas.
M5.	Viena PP klaida vidutiniškai atitinka vieną funkcinio vėlinimo klaidą. Jei testinė pora <u, v> yra SIT testas ir aptinka funkcinio vėlinimo klaidą (I, O, t I, t O), kur t I ir t O – kylantys (krentantys) signalai įėjime I ir išėjime O. Tuomet testinė pora <v, u> aptinka funkcinio vėlinimo klaidą (I, O, t I‘, t O‘), kur t I‘ ir t O‘ – krentantys (kylantys) signalai įėjime I ir išėjime O. Pritaikius pirmą ir antrą taisykles pateiktas pirmoje lentelėje (1 lentelė. Taisyklės funkcinio vėlinimo testui iš PP klaidų testo sudarymui **[4, 15]**), sugeneruojami funkcinio vėlinimo testai iš PP testų. Turint įėjimo signalų rinkinį w, kuris aptinka q kiekį PP klaidų atstovaujančiame funkcinio vėlinimo teste, buvo sudaryta 2*q įėjimo signalų rinkinio porų. Sudarytas funkcinio vėlinimo testas yra „2-aptikimų“, o

	sudarytos poros – SIT testai.
M6.	Du atskiri PP testai sugeneruoti ir transformuoti pagal trečią taisyklę, kuri pateikta pirmoje lentelėje (1 lentelė. Taisyklės funkcinio vėlinimo testui iš PP klaidų testo sudarymui **[4, 15]**) ir sujungti į vieną 2-aptikimų funkcinio vėlinimo testą. Sudarytas testas yra MIT testas.
M7.	Pagal antrą ir trečią taisykles, pateiktas pirmoje lentelėje (1 lentelė. Taisyklės funkcinio vėlinimo testui iš PP klaidų testo sudarymui **[4, 15]**), suformuoti funkcinio vėlinimo testai ir apjungti į 2-aptikimų testą. Sudarytos poros yra MIT testai.
M8.	Pagal pirmą, antrą ir trečią taisykles, apteiktas pirmoje lentelėje (1 lentelė. Taisyklės funkcinio vėlinimo testui iš PP klaidų testo sudarymui **[4, 15]**), suformuoti funkcinio vėlinimo testai ir apjungti į 3-aptikimų testą. Viena sudarytų testinių sekų porų yra SIT testai, kita dalis – MIT testai.
M9.	Du atskiri PP testai transformuoti pagal M5 metodą ir gauti 2-aptikimų funkcinio vėlinimo testai sujungti į vieną 4-aptikimų SIT testą.

Atitinkamai 4, 5 ir 6 lentelėse pateikti išanalizuoti šių metodų, generuojančių funkcinio vėlinimo testus rezultatai. „1 - aptikimo“ testo rezultatai gauti panaudotus vienodą PP testą. Paryškinti rezultatai atspindi didžiausius perdavimo klaidų padengimus. Kaip matyti iš rezultatų, M1 SIT metodas vidutiniškai aptiko daugiau nei 95% perdavimo klaidų. Tuo tarpu nors ir metodais M2 ir M3 gauti testai buvo vidutiniškai 3,8 karto trumpesni, tai neatsveria daugiau nei 12% prastesnio perdavimo klaidų padengimo. Geriausias signalų perdavimo padengimas pasiektas M8 metodu, kur testai suformuoti pagal pirmą, antrą ir trečią taisykles. Sudaryti SIT ir MIT testai apjungti į vieną testą.

4 lentelė. M1-M3 – „1 - aptikimo“ funkcinio vėlinimo testų rezultatai **[4]**

Schema	M1		M2		M3		Perdav. Klaidos	Funk. vėlin. (PP) klaidos
	Padeng. (%)	Testo dydis	Padeng. (%)	Testo dydis	Padeng. (%)	Testo dydis		
C432	**95.56**	348	87.28	117	52.98	117	1412	540
C499	**94.40**	5180	91.23	1077	94.29	1077	3430	5184
C880	**98.91**	1001	90.90	381	83.56	381	2396	1326

C1355	**97.13**	5162	92.36	1011	92.09	1011	3350	5184
C1908	**95.24**	2359	81.58	620	69.08	620	4848	3004
C2670	**96.51**	1820	90.44	448	67.50	448	5646	3320
C3540	83.08	1457	**88.28**	515	80.47	515	8960	2588
C5315	**98.41**	4950	97.94	1169	89.24	1169	13816	10540
C6288	**99.75**	1065	98.72	268	98.70	268	14422	3068
C7552	**99.21**	5801	94.21	2115	98.90	2115	19160	12188
Vidurkis	**95.82**	2914	91.29	772	82.68	772	7744	4694

5 lentelė. M4-M9 – „n – aptikimo“ funkcinio vėlinimo testų rezultatai (klaidų padengimas) **[4]**

Schema	Perdavimo klaidų padengimas (%)					
	M4	M5	M6	M7	M8	M9
C432	98.11	97.67	88.59	93.02	98.48	**99.42**
C499	94.40	94.40	91.6	**99.83**	**99.83**	94.40
C880	99.17	99.21	90.9	91.78	**100.00**	99.29
C1355	97.13	97.13	92.6	98.69	**98.99**	97.13
C1908	95.61	97.48	83.13	83.48	96.74	**97.85**
C2670	98.35	98.65	90.75	91.73	**99.56**	99.11
C3540	89.03	88.71	90.01	90.77	**96.85**	93.79
C5315	99.55	99.58	97.96	98.09	**99.95**	99.86
C6288	99.88	99.95	98.72	98.72	99.94	**99.99**
C7552	99.52	99.66	94.57	99.11	99.62	**99.74**
Vidurkis	97.08	97.24	91.88	94.52	**99.00**	98.06

Taip pat žemiau esančioje lentelėje (6 lentelė. M4-M9 – „n – aptikimo“ funkcinio vėlinimo testų rezultatai (testo dydis) **[4]**) matyti, jog M8 metodu, kuris

padengia daugiausiai signalo perdavimo klaidų, gauti funkcinio vėlinimo testai yra apie 2.6 karto trumpesni, negu M9 metodu gauti testai.

6 lentelė. M4-M9 – „n – aptikimo" funkcinio vėlinimo testų rezultatai (testo dydis) **[4]**

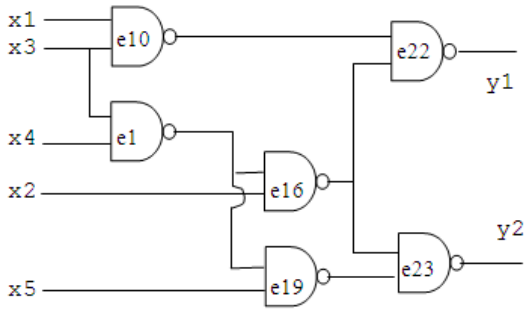
Schema	Testų dydžiai					
	M4	M5	M6	M7	M8	M9
C432	697	696	247	234	582	1394
C499	10351	10360	2136	2154	7334	20702
C880	1985	2002	791	762	1763	3970
C1355	10302	10324	2059	2022	7184	20604
C1908	4656	4718	1231	1240	3599	9312
C2670	3716	3640	1272	896	2716	7432
C3540	2962	2914	1098	1030	2487	5924
C5315	9905	9900	2315	2338	7288	19810
C6288	2219	2130	522	536	1601	4438
C7552	11744	11602	6273	4230	10031	23488
Vidurkis	5854	5829	1794	1544	4459	11707

Lyginant SIT ir MIT „1-aptikimo" testus, matome, jog SIT testai aptiko apie 13% daugiau klaidų negu MIT, dėl to yra labiau priimtini. Kita vertus didžiausias perdavimo klaidų padengimas gaunamas maišant SIT ir MIT testus ir suformavus „n-aptikimo" testus.

2.5.5. SIT ir MIT funkcinio vėlinimo testų sudarymo metodų panaudojant aktyvumo vektorius analizė

Sužymėjus aktyvius įėjimus, nuo kurių pasikeitimo priklauso analizuojamo išėjimo signalo pasikeitimas, sužymėtą seką vadiname aktyvumo vektoriumi, o aktyvius simbolius – aktyvumo reikšmėmis. Tarkime yra programinis schemos modelio prototipas, turintis *n* įėjimų ir *m* išėjimų. Pažymėjime įėjimo simbolius $P=<p_1, p_2, \ldots, p_i, \ldots, p_n>$, kur $p_i= \{0, 1\}$, $i=1, 2, \ldots, n$. Aktyvumo vektorius $P^j=<p^j_1, p^j_2, \ldots, p^j_i, \ldots, p^j_n>$ yra susietas su išėjimu *j*. Kiekviena aktyvumo vektoriaus sudedamoji dalis gali įgyti šias reikšmes: 0, 1, N, V. Reikšmė N rodo, jog įėjime *i* padavus signalą lygų 0,

gausime priešingą reikšmę išėjime *j*. Reikšmė V rodo, jog įėjime *i* padavus signalą lygų 1, gausime priešingą reikšmę išėjime *j*. $P0^j$ žymėsime, jei *j* išėjime aktyvumo vektorius nustato 0, ir $P1^j$ – jei 1. V ir N ir yra aktyvumo vektoriaus aktyvios reikšmės. Aktyvumo vektorius apibendrina n+1 įeinančių signalų, kurie tarpusavyje skiriasi tik vienu simboliu. Pavyzdžiui, priskiriame įeinantiems signalams reikšmes $<X_1, X_2, X_3, X_4, X_5>$ = <01011> testuojamam C17 schemos modeliui (3 pav. Testuojamos C17 schemos modelis).



3 pav. Testuojamos C17 schemos modelis *[15]*.

Paduodant šį signalo reikšmių rinkinį, išėjime y1 gauname 1. Invertuojant kiekvieną signalų rinkinio reikšmę, išvedame aktyvumo vektorių <0*VN*11>, kuris apibendrina signalų rinkinius, pateiktus 7 lentelėje „Įėjimo signalų rinkiniai“. Keičiant kiekvieno aktyvumo vektoriaus po vieną aktyvią reikšmę į priešingą, galime gauti SIT testą. O pritaikius įvairius algoritmus, galima sudaryti ir MIT funkcinio vėlinimo testus.

7 lentelė. Įėjimo signalų rinkiniai

Signalų rinkinio numeris	X1	X2	X3	X4	X5
1.	0	1	0	1	1
2.	1	1	0	1	1
3.	0	0	0	1	1
4.	0	1	1	1	1
5.	0	1	0	0	1
6.	0	1	0	1	0

Iš viso kiekvienam išėjimui galima sugeneruoti po aktyvumo vektorių, kuris išėjime nustato $P1^j$ arba $P0^j$ signalus. Jei aktyvumo vektorius padengia P_a padengia

vektorių P_b, žymėsime $P_a > P_b$. Kad aktyvumo vektorius P_a dengtų P_b vektorių, turi būti išpildytos šios sąlygos *[15]*:

1. Aktyvumo vektorius P_b turi aktyvias reikšmes tik tuose pačiuose įėjimuose, kaip ir vektorius P_a.
2. Aktyvumo vektoriaus P_b reikšmės sutampa su aktyvumo vektoriaus P_a reikšmėmis tuose pačiuose įėjimuose.

Žemiau pateiktoje lentelėje (8 lentelė. Padengimo sąlygų tenkinimas) pateiktas pavyzdys, įgyvendinantis šias sąlygas, norint padengti P_b, vektoriumi P_a. Lentelėje paryškintos aktyvios reikšmės, kurios tuose pačiuose įėjimuose sutampa.

8 lentelė. Padengimo sąlygų tenkinimas

P_a	N	V	N	V	N	V
P_b	N	V	1	0	0	1

Aktyvumo vektorius laikomas esminiu, kuomet jo nedengia kitas aktyvumo vektorius, t.y. aktyvios reikšmės skiriasi nuo kitų išėjimo aktyvumo vektorių aktyvių reikšmių. Aktyvumo vektorius generuojamas atsitiktinai sugeneravus schemos įėjimo reikšmes ir simuliuojant schemos veikimą, remiantis jos funkciją aprašančios „juodos dėžės“ modeliu *[15]*.

Paanalizuokime testuojamos C17 schemos modelio (3 pav. Testuojamos C17 schemos modelis) signalų rinkinį $P = <X_1, X_2, X_3, X_4, X_5> = <01110>$, kuris nustato išėjimuose $<y1, y2> = <00>$ reikšmes. Keičiant po vieną įėjimo signalų rinkinio reikšmę, gaunam išėjimo rezultatus, kurie patalpinti žemiau esančioje lentelėje (9 lentelė. Įėjimo signalų „<01110>“ inversijos įtaka išėjimų rezultatams). Paryškinti tie įėjimo signalų pasikeitimai, kurie keitė reikšmę išėjimo signaluose. Remiantis lentelėje pateiktais rezultatais, gauname šiuos aktyvumo vektorius: $P0^{y1} = <N1VV0>$, $P0^{y2} = <01VV0>$. Lygiai taip pat gauname įėjimo signalų rinkiniui $P = <00110>$, kurio signalų inversijų įtaką išėjimų rezultatams pateikti lentelėje (10 lentelė. Įėjimo signalų „<00110>“ inversijos įtaka išėjimų rezultatams), aktyvumo vektorius $P0^{y1} = <N0110>$, $P0^{y2} = <00110>$. Pastarieji nėra esminiai, nes juos perdengia ankstesni vektoriai.

9 lentelė. Įėjimo signalų „<01110>“ inversijos įtaka išėjimų rezultatams

p_1	p_2	p_3	p_4	p_5	y_1	y_2
0	1	1	1	0	0	0

1	1	1	1	0	1	0
0	0	1	1	0	0	0
0	1	**0**	1	0	1	1
0	1	1	**0**	0	1	1
0	1	1	1	1	0	0

10 lentelė. Įėjimo signalų „<00110>“ inversijos įtaka išėjimų rezultatams

p_1	p_2	p_3	p_4	p_5	y_1	y_2
0	1	1	1	0	0	0
1	1	1	1	0	1	0
0	0	1	1	0	0	0
0	1	**0**	1	0	1	1
0	1	1	**0**	0	1	1
0	1	1	1	1	0	0

Aktyvumo vektorius galima generuoti atsitiktinai parenkant kiekvienam įėjimui signalus. Sugeneravus vieną vektorių, paieškos procesą įmanoma paspartinti keičiant aktyvias reikmes, kurios analizuojamame išėjime nustatys priešingą reikšmę.

Kai kuriuos signalo perdavimo gedimus padengti gali tik MIT testai, o lengviausias būdas iš aktyvumo vektorių gauti MIT testą yra, kuomet antros testinės poros signalų sekos visos aktyvios reikšmės yra pakeičiamos priešingomis negu pirmoje sekoje *[15]*.

Bendraujant drauge su prof. habil. dr. R. Šeinausku, buvo sudaryti ir kiti galimi MIT testų konstravimo metodai, grįsti vektorių esančių aibėse $A0^j$ ir $A1^j$ analize. Analizuojamus aktyvumo vektorius kiekvienam išėjimui, galima suskirstyti į aibes $A1^j$ bei $A0^j$. Aibėje $A1^j$ esantys aktyvumo vektoriai, išėjime j nustato reikšmę 0, o aibėje $A0^j$ - 1. Kuomet sudaromos testų sekų poros aktyvios V ir N reikšmės pakeičiamos 1 ir 0. Su paršuotu testu galima aptikti signalo perdavimo klaidas, jei poros sudaromos iš priešingas išėjimo reikšmes gražinusių aibėse esančių aktyvumo vektorių. Priešingu atveju taip pat galima aptikti signalo perdavimo klaidas, tačiau signalo perdavimo klaidą galime tik stebėti. Pasiūlyti keturi funkcinio vėlinimo testų sudarymo metodai. Testai gaunami apjungiant aktyvumo vektorius į poras. Žemiau esančioje lentelėje

pateikiami šių metodų aprašai (11 lentelė. MIT funkcinio vėlinimo testų iš aktyvumo vektorių sudarymas).

11 lentelė. MIT funkcinio vėlinimo testų iš aktyvumo vektorių sudarymas

Metodo nr.	Metodo aprašymas
V1.	Kiekvienas iš aibės $A0^j$ ($A1^j$) aktyvumo vektorius laikomas antru testinės poros vektoriumi. Tuo tarpu iš aibės $A1^j$ ($A0^j$) parenkamas pirmasis sudaromos testinės poros vektorius, turintis daugiausiai priešingų reikšmių antro vektoriaus aktyvioms reikšmėms.
V2.	Kiekvienas iš aibės $A0^j$ ($A1^j$) aktyvumo vektorius laikomas antru testinės poros vektoriumi. Tuo tarpu iš aibės $A0^j$ ($A1^j$) parenkamas pirmasis sudaromos testinės poros vektorius, turintis daugiausiai priešingų reikšmių antro vektoriaus aktyvioms reikšmėms.
V3.	Kiekvienas iš aibės $A0^j$ ($A1^j$) aktyvumo vektorius laikomas antru testinės poros vektoriumi. Tuo tarpu iš aibės $A1^j$ ($A0^j$) parenkamas pirmasis sudaromos testinės poros vektorius, turintis mažiausiai priešingų reikšmių antro vektoriaus aktyvioms reikšmėms, tačiau ne mažiau už 1.
V4.	Kiekvienas iš aibės $A0^j$ ($A1^j$) aktyvumo vektorius laikomas antru testinės poros vektoriumi. Tuo tarpu iš aibės $A0^j$ ($A1^j$) parenkamas pirmasis sudaromos testinės poros vektorius, turintis mažiausiai priešingų reikšmių antro vektoriaus aktyvioms reikšmėms, tačiau ne mažiau už 1.

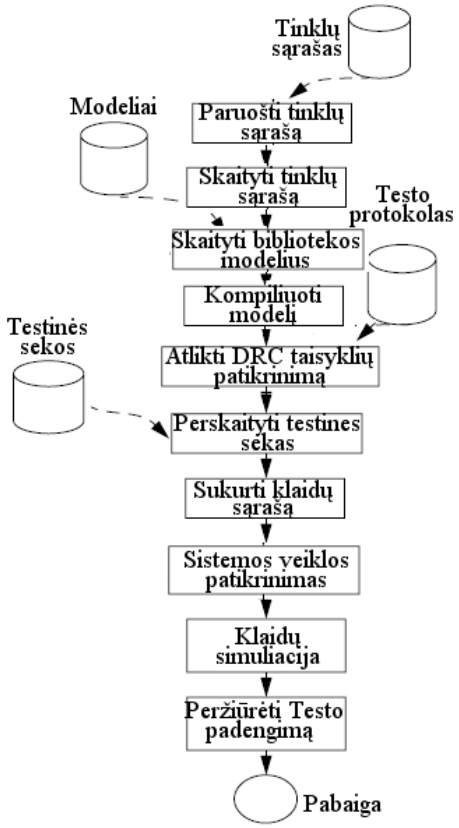
Šių vektorių pagrindu sugeneruojami testai visoms galimoms testuojamos schemos ar įrenginio realizacijoms gali būti pakankamai dideli, tačiau yra galimybė juos optimizuoti konkrečiai realizacijai, ir laikas skirtas optimizavimui atlikti, palyginus su visos schemos rinkai paruošimo laiku - mažas *[4, 14]*. Testo dydis tiesiogiai priklauso nuo surastų aktyvumo vektorių, kurių kiekis gali būti ribojamas. Jeigu testo, sudaryto iš riboto kiekio aktyvumo vektorių, klaidų padengimas netenkina – galima vektorių apribojimus atlaisvinti, papildyti.

2.6. Eksperimentavimo sistemų analizė

2.6.1. Klaidų simuliacijos proceso eigos analizė

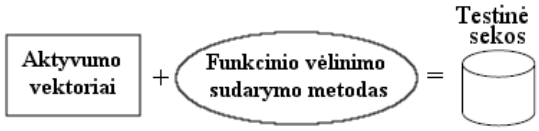
Siekiant automatizuoti testavimo proceso dalis, būtina apžvelgti naudojamus objektus klaidų simuliacijos proceso eigoje, kuri pavaizduota žemiau esančiame paveikslėlyje (4 pav. Klaidų simuliacijos proceso eiga). Simuliacijos metu perskaitomi

duomenų failai aprašantys schemos tinklų struktūra, testines sekas ir jas aprašantys papildomi failai, kurių reikalavimai detaliau aprašyti Synopsys TetraMAX vartotojo vedlyje. Simuliacijos rezultatas – tai tekstinis failas, kuriame pateikti klaidų simuliacijos rezultatai.



4 pav. Klaidų simuliacijos proceso eiga *[7]*

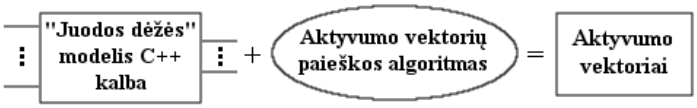
Schemų modelių bei tinklų sąrašų failai projektuojami žmonių, todėl šių failų sudarymo proceso automatizuoti nepavyks, tačiau galima automatizuoti testinių sekų sudarymo procesą. Testinės sekos, kuriamos sistemos atveju gaunamos aktyvumo vektoriams pritaikius kombinuojamų funkcinio vėlinimo sudarymo metodus (5 pav. Testinės sekų gavimas).



5 pav. Testinės sekų gavimas

Aktyvumo vektoriai sudaromi analizuojant „juodos dėžės“ modelį (6 pav. Aktyvumo vektorių sudarymas). Mūsų atveju „juodos dėžės“ modelis, tai C++ kalba

aprašyta schemos veikimo funkcija.



6 pav. Aktyvumo vektorių sudarymas

Kadangi siekiama, jog funkcinio vėlinimo testų sudarymo metodų taikymas, „juodų dėžių“ modeliu grįstas aktyvumo vektorių paieškos algoritmų automatizavimas būtų kaip galima labiau pakartotinai panaudojamas, sistemos komponentai kompiliuojami taip, kad failų sistemoje būtų pasiekiami atskirų failų pavidalu. Taip pat remiantis K.I.S.S. principu, pagal kurį paprastumas laikomas dizaino privalumu, „juodos dėžės“ modeliai bus sukompiliuojami į dinamiškai užkraunamas bibliotekas, aktyvumo vektorių paieškos algoritmas sukompiliuojamas į atskirą paleidžiamąjį failą, kaip ir kiekviena funkcinio vėlinimo sudarymo metodo realizacija.

2.6.2. Turinio ir informacijos tvarkymo sistemų analizė

Aktyvumo vektoriai ir testinės sekos - kaupiami duomenų saugykloje failų pavidalu ir yra lengvai pasiekiami bet kurios programinės įrangos, kuriai yra suteiktos tokios teisės. Testinės sekos gavimo paveikslėlyje (5 pav. Testinės sekų gavimas) pavaizduota, jog pritaikius vieną metodą gausime testines sekas iš schemos aktyvių reikšmių. Jei pritaikysime daugiau metodų – gausime daugiau testinių sekų duomenų failų. Grupuojant testinių sekų sudarymo metodus, šis skaičius dar labiau išauga. Kadangi Synopsys TetraMAX programa galutinius rezultatus, atspindinčius klaidų padengimą, taip pat suformuoja faile, mums reikia sistemos, kuri ne tik atvaizduotų, analizuotų testavimo rezultatus, bet ir automatiškai grupuotų failus į jiems paskirtas vietas.

Kad išsiaiškinti kokios sistemos reikia, buvo atlikta reikalavimų specifikacija (3 skyrius. Sistemos reikalavimų specifikacija ir analizė), kurios metu paaiškėjo keliami saugumo reikalavimai, duomenų prieinamumo, failų tvarkymo. Iš gautos informacijos paaiškėjo, jog labiausiai tinka vidinės įmonės portalo veikimo principu grįsta sistema, dėl šių funkcijų:

- Vartotojų kūrimas;
- Vartotojų teisių sistema;
- Saugumas;
- Duomenų saugyklos;

- Taikomųjų programų integracija;
- Intuityvi vartotojo sąsaja;
- Sistematinė ir dinaminė navigacija;
- Menu susiejimas su katalogais ir automatinis kelio apskaičiavimas.

Šios funkcijos labai artimos dinaminio turinio tvarkymo sistemoms (TTS), kurių pagrindu kartais kuriami eksperimentavimo bei automatizavimo sprendimai. Be standartinių funkcijų, mūsų sistemai reikia funkcijos kuri automatiškai suformuotų kelią iki katalogų, kuriuose bus laikomi duomenų bei rezultatų failai, susieti su tam tikru eksperimentu. Apžvelkime pagal funkcionalumą pirmas 10 vietų užimančias dinamines TTS*[16]* (12 lentelė. Bendra TTS informacija), jų saugumą ir funkcijas, kad nustatytume ar galime panaudoti kurią nors iš TTS eksperimentavimo sistemai vystyti. Kaip matome iš TTS bendros informacijos palyginimo lentelės, „Jalios“ siūlomo sprendimo kaina yra labai aukšta, dėl to šio sprendimo funkcionalumo neanalizuosime.

12 lentelė. Bendra TTS informacija

	Drupal 6.10	eZ Publish 4.2	Hippo CMS 6.0	Impress CMS 1.2.1	inxire ECM 5	Jahia Enterprise Edition v6 6.0	Jalios JCMS 5.7	Joomla! 1.5.10	Plone 3.0	WordPress 2.2.1
Aptarnaujanti programinė įranga	Apache	Kita	Kita	Apache	J2EE	J2EE	J2EE	CGI	Zope	Apache
Kaina	Nemokama	30€ + PVM	Nerasta	Nemokama	Nekomerciniams projektams - nemokama	Nekomercinei veiklai - nemokamai. Komercinei–iki ~50 000 €	Nuo 10 000€ iki 80 000 € už kiekvieną tarnybinę stotį.	Nemokama	Nemokama	Nemokama
Duomenų saugykla	MySQL	MySQL	MySQL	MySQL	Oracle	Postgres	Nėra	MySQL	Kita	MySQL
Licencija	Atviro kodo	Atviro kodo	Atviro kodo	Atviro kodo	Kodas nėra atviras	Atviro kodo	Kodas nėra atviras	Atviro kodo	Atviro kodo	Atviro kodo
Operacinė sistema	Visos platformos	Visos platformos	Visos platformos	Visos platformos	Visos platformos	Visos platformos	Visos platformos	Visos platformos	Visos platformos	Visos platformos
Programavimo kalba	PHP	PHP	Java	PHP	Java	Java	Java	PHP	Python	PHP
Aukščiausio lygio teisės	Ne	Ne	Taip	Ne	Taip	Taip	Taip	Ne	Ne	Ne
Shell priėjima	Ne	Ne	Taip	Ne	Taip	Taip	Taip	Ne	Taip	Ne

s										
Interneti nis servisas	Apach e	Apach e	Bet kuris	Apache	Bet kuris	Apache	Apache	Apac he	Apach e	Apach e
Adresas internete	http:// drupal .org	http:// ez.no	http:// wiki.o nehip po.co m	http://w ww.im pressc ms.org	http://ww w.inxire.c om	http://w ww.jahi a.com	http:// www.j alios.c om	http:// www .joom la.org	http:// plone. org	http:// wordp ress.or g

Suformuojam užimančių pirmas 9 vietas, pagal funkcionalumą TTS funkcijų palyginimo lentelė (13 lentelė. TTS funkcijų tyrimas), neįtraukiant „Julios“ TTS, kurią pripažinome netinkančia dėl aukštos kainos. Tušti langeliai reiškia, jog nagrinėjamas TTS tokio funkcionalumo neturi.

13 lentelė. TTS funkcijų tyrimas

Funkcijų palaikymas	Drupal 6.10	eZ Publish eZ Publish 4.2	Hippo CMS 6.0	ImpressCMS 1.2.1	inXire ECM 5	Jahia Enterprise Edition v6.6.0	Joomla! 1.5.10	Plone 3.0	WordPress 2.2.1
“Captcha” technologija	Į	Į		✓	✓	✓	Į	Į	
El. pašto verifikavimas	✓	✓		✓	✓	Į	✓	✓	Į
Privilegijų sistema	✓	✓	✓	✓	✓	✓	✓	✓	✓
Prisijungimų istorija	✓	✓		✓	✓	✓	✓	Į	Į
Problemų pranešimai		$		A	✓	✓		Į	Į
Testavimo zona		✓	✓	✓	✓	✓		✓	A
Sesijų palaikymas	✓	✓		✓	✓	✓	✓	Į	Į
SSL suderinamumas	✓	✓	✓		✓	✓	✓	✓	✓
SSL prisijungimas		✓		✓	✓	✓	✓	Į	Į
SSL puslapiai		✓			✓	✓	✓		A
Versijų palaikymas	✓	✓	✓	✓	✓	✓	Į	✓	Į
API palaikymas	✓	✓	✓	✓	✓	✓	✓	✓	✓
Masinis duomenų įkėlimas	Į	✓	✓	Į	✓	✓	✓	✓	Į
Puslapio kalbos	✓	✓	✓	✓	✓	✓	✓	✓	✓
Stilių vedlys	A				$	A		Į	
Šablonų kalba	A	✓		✓	✓	✓	✓	✓	
Funkcijos atšaukimas	A	✓	✓	A	✓	✓		✓	Į
editorius	Į	✓	✓	✓	✓	✓	✓	✓	✓
Puslapių kešavimas	✓	✓	✓	✓	✓	✓	✓	✓	Į
Statinio puslapio išvedimas		✓	$	✓	✓			Į	A
Išvaizdų temos	✓	✓		✓	A	✓	✓	✓	✓
Puslapio statistika	✓	Į		Į	✓	✓	✓	Į	Į
RSS naujienos	✓	✓	✓	✓	✓	✓	✓	✓	✓
UTF-8 koduotės palaikymas	✓	✓	✓	✓	✓	✓	✓	✓	✓
Duombazių analizė		A		Į	✓	✓	Į	✓	
Dokumentų valdymas	A	✓	$	Į	✓	✓	Į	✓	
Paieškos varikliai	✓	✓	✓	✓	✓	✓	✓	✓	✓
Navigacijos žemėlapis	Į	✓	✓	Į	✓	✓	Į	✓	Į

Sutartas žymėjimas: ✓ – funkcija palaikoma; Į – Nemokamas įskiepis; A – Apribota; $ - Už papildoma kaina

Ištyrus siūlomus TTS sprendimus ir jų funkcijas, pastebėta, jog visos sistemos turi daug naudingų funkcijų, tačiau nei vienos pagalba nepavyktų susieti dinamiškai kuriamų eksperimentų, kuriuos būtų galima pateikti, kaip menu elementus, su eksperimentų duomenims ir rezultatams talpinti skirtais katalogais, automatiškai suskaičiuoti kiekvienam eksperimentui reikalingus kelius iki katalogų, rikiuoti eksperimentus, ir pateikti juos vizualiai tvarkingoje formoje. Dėl šių priežasčių sistemą sukursime patys.

2.7. Vartotojų analizė

Bendraujant su srities specialistais buvo atlikta vartotojų analizė.

2.7.1. Vartotojų aibė, tipai ir savybės

Sistema naudosis šios vartotojų grupės:

Sistemos vartotojas (užsakovas);

Sistemos kūrėjas;

Sistemos vartotojas (užsakovas):

Vartotojo kategorija – projekto vadovas;

Patirtis dalykinėje srityje – srities specialistas (meistras);

Patirtis informacinėse technologijose – patyręs;

Vartotojų prioritetai – svarbiausi vartotojai;

Sistemos kūrėjas:

Kategorija – kūrėjas;

Patirtis dalykinėje srityje – informatikas;

Vartotojų prioritetai – antraeiliai vartotojai;

2.7.2. Vartotojų tikslai ir problemos

Sistemos vartotojas (užsakovas):

Vartotojo sprendžiami uždaviniai (atliekamos funkcijos):

- Peržiūrėti bei analizuoti funkcinio vėlinimo testų generavimo tarpinius duomenis ir rezultatus.

Sistemos kūrėjas:

Vartotojo sprendžiami uždaviniai (atliekamos funkcijos):

- Generuoti aktyvumo vektorių pradinį duomenų failą;
- Paleisti aktyvumo vektorių paieškos programą;
- Sudaryti aktyvumo vektorių poras;

- Peržiūrėti bei analizuoti rezultatus.

2.8. Esamų sprendimų analizė

Audrius Kažukauskas yra parašęs įėjimo bei išėjimo susietumams nustatyti programą. Programa remiasi pateikiamos schemos funkciniu aprašymu C++ kalboje ir gali būti pritaikyta nustatant svorius.

Testines sekas galima generuoti Synopsys TetraMAX ATPG, tačiau šis įrankis remiasi schemos loginių elementų išdėstymo aprašymu ir dėl to mums netinka.

2.9. Rizikos faktorių analizė, įvertinimas

2.9.1. Rizikos, su kuriomis susiduriame kurdami sistemą

Pagrindiniai rizikos faktoriai, su kuriais susiduriame kurdami informacinę sistemą yra besivystantys vartotojų poreikiai, sidabrinės kulkos sindromas (planas „nušauti vienu šūviu) bei dviprasmybių analizuojant reikalavimus atsiradimas.

Žemiau esančioje lentelėje (14 lentelė. Rizikų įvertinimai) pateiktas šių rizikų sąrašas, su kuriomis galime susidurti kurdami sistemą bei jų tikimybiniai įvertinimai.

14 lentelė. Rizikų įvertinimai

Rizikos faktorius	Tikimybinis įvertinimas
Dviprasmybės analizuojant reikalavimus	0.3
„Sidabrinės kulkos sindromas“ (vienu darbu atlikti kelis)	0.4
Besivystantys vartotojo reikalavimai	0.6

2.9.2. Rizikos suvaldymas

Atidžiau įvertinus galimybes ir grėsmes, daugiau bendraujant su projekto dalyviais bei vystant reikalavimų specifikaciją galime suvaldyti rizikas, išvengti neplanuotų nuostolių ir laiku užbaigti projektą.

Žemiau esančioje lentelėje (15 lentelė. Rizikų suvaldymas) pateikta rizikos faktorius sumažinantys sprendiniai iškylančioms problemoms spręsti.

15 lentelė. Rizikų suvaldymas

Rizikos faktorius	Sprendimas mažinantis riziką
Dviprasmybės analizuojant reikalavimus	Daugiau bendravimo tarp projektų vadovo ir darbuotojų dirbančių prie projekto.

„Sidabrinės kulkos sindromas“	Atidžiau įvertinti galimybes bei grėsmes.
Besivystantys vartotojo reikalavimai	Sudaryti reikalavimų specifikaciją bei dvišales sutartis.

2.10. Rezultatų kokybės kriterijai

Žemiau esančioje lentelėje pateikti išskiriami rezultatų kokybės kriterijai, kurie taikomi kuriamai sistemai (16 lentelė. Kokybės kriterijai nagrinėjami darbe).

16 lentelė. Kokybės kriterijai nagrinėjami darbe

Sistemos charakteristika	Aprašymas	Siekiami rezultatai
Patikimumas	Sistemos veikimas be klaidų.	Siekiama, kad sistema veiktų be klaidų, esant reikiamiems resursams, kuomet siekiama konkretaus rezultato.
Saugumas	Sistemos duomenų saugumas.	Duomenis trinti gali tik tie vartotojai, kurie turi paskyras, leidžiančias tai daryti.
Efektyvumas	Sistemos vartotojo tikslų su reikiamu tikslumu ir pilnumu tenkinimas.	Sistema patenkina vartotojo poreikius, tikslus.
Daugkartinis panaudojimas	Sistemos atominių dalių, komponentų daugkartinis taikymas.	Komponentai, gali būti naudojami pakartotinai, nedubliuojant kodo.
Lankstumas	Sistemos modifikavimo bei pritaikymo naudoti kitose sistemose lengvumas.	Sistema sudaryta iš atominių dalių. Kiekvienos dalies kodas pakoregavus nesunkiai kompiliuojamas.

2.11. Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti analizės išvados

Išanalizavus funkcinio vėlinimo sudarymo metodus, pastebėta, jog tik kombinuojant testus, galime gauti didžiausią signalo perdavimo klaidų padengimą. M8 metodu gauti rezultatai yra geriausi iš visų analizuotų metodų, ir jie gauti maišant „1 –

aptikimo“ SIT testus su „2 – aptikimų“ MIT testais. Analizė atskleidė, jog SIT ir MIT testų kombinacija duoda didžiausią perdavimo laidų padengimą. Sudarius „n-aptikimo“ testą, galime gauti didesnį klaidų padengimą, tačiau proporcingai apjungiamų testų dydžiams išauga ir bendro testo dydis.

Jei schemos įėjimo signalus nuo kurių priklauso išėjimų reikšmės sužymėsime, gausime aktyvumo vektorių, kuris gali būti padengiamas arba gali padengti kitą aktyvumo vektorių. Padengiantis vektorius yra laikomas esminiu. Panaudojus aktyvumo vektorius, galima sėkmingai kurti tiek SIT tiek MIT testus, o kombinuojant siūlomus metodus, gauti daugiau funkcinio vėlinimo klaidų padengiančius testus.

Pastebėta, jog labai plati įvairių portalų ir TTS sistemų rinka, tačiau prireikus specifinio branduolio, kuriame būtų integruota galimybė susieti menu punktus su katalogais, nelieka kitos išeities, kaip sistemą, kartu su jos branduoliu pasirašyti patiems.

Taip pat pastebėta, jog galime pritaikyti Audriaus Kažukausko parašytą programinę įrangą, nustatančią ryšius tarp schemos įėjimo bei išėjimo signalų, aktyvumo vektorių kiekiui riboti.

3. Sistemos reikalavimų specifikacija ir analizė

3.1. Reikalavimų specifikacija

Vartotojo tipai bei sistemos kūrimo tikslai buvo išanalizuoti antrame skyriuje Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti analizė“. Reikalavimų specifikacija sudaryta bendraujant su sistemos užsakovu ir sričių specialistais: duomenų bazių projektavimo, testavimo, sistemų kūrimo ir kt. Pasitelkus Volere reikalavimų šablonus *[17]*, buvo sudaryti funkciniai ir nefunkciniai sistemos reikalavimai. Teisinių reikalavimų, reikalavimų veikimo priežiūrai bei veikimo sąlygoms nėra.

3.1.1. Organizacijos charakteristika

3.1.1.1. Apie institutą

Kauno technologijos universiteto Informacinių technologijų plėtros institutas įsteigtas 2002 metais ir yra 1963 metais įkurto Skaičiavimo centro, 1981 metais jo bazėje įsteigto Kolektyvinio naudojimo skaičiavimo centro, nuo 1991 metų –

Skaičiavimo centro teisių ir įsipareigojimų perėmėjas ir tęsėjas, plėtojantis ne tik inžinerinę, bet ir mokslinę veiklą.

Instituto darbuotojai teikia paslaugas eksploatuojant Universiteto kompiuterius bei programinę įrangą, kompiuterių tinklus, talkina studijų procesui, teikia paramą pedagoginiams darbuotojams naudojant naujausias informacines technologijas, sudaro sąlygas Universiteto pedagogams užtikrinti reikiamą studijų proceso kokybę.

3.1.1.2. Instituto veiklos kryptys

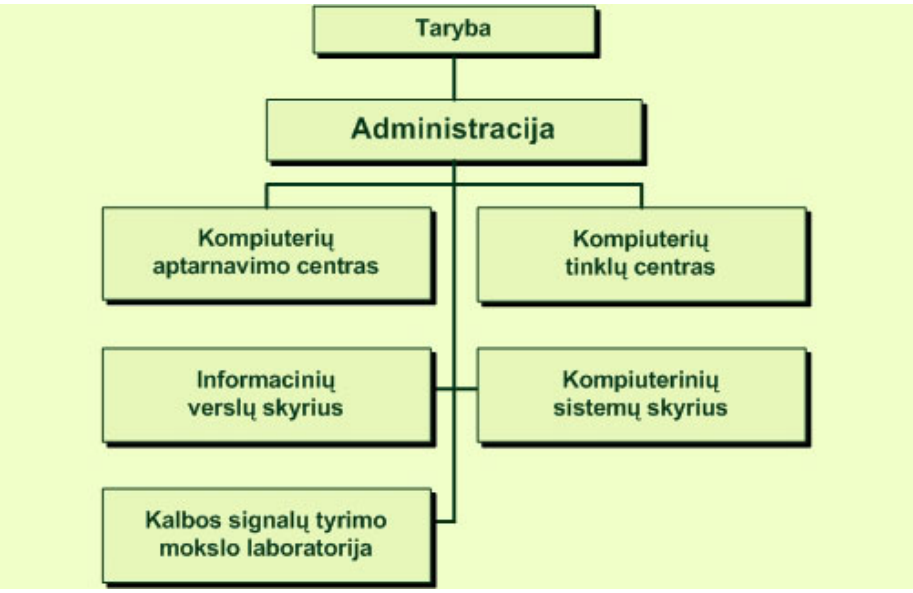
- Informacinių technologijų efektyvumo tyrimai ir jų plėtra
- Administruojami kompiuteriai ir jų tinklai fakultetuose, prižiūrimos operacinės sistemos bei taikomieji paketai, teikiamos interneto paslaugos, palaikomas mokymo procesas fakultetuose
- Teikiamos centralizuotos elektroninio pašto bei interneto paslaugos universiteto darbuotojams bei studentams, prižiūrimi ir plėtojami magistraliniai Universiteto tinklai, vykdomos LITNET tinklo valdymo funkcijos bei jo plėtra, teikiamos vardų sričių registravimo paslaugos Lietuvos interneto vartotojams
- Teikiamos centralizuotos distancinio mokymo paslaugos, rengiamos vaizdo konferencijos, kompiuterinio raštingumo ir kvalifikacijos ugdymo kursai

3.1.1.3. Instituto misija

Vykdyti mokslinius tyrimus ir informacinių technologijų (IT) plėtrą, kurti naujas mokslo žinias ir jas skleisti visuomenei bei taikyti studijų procese.

3.1.1.4. Organizacijos struktūros modelis

Žemiau pateikta organizacijos struktūros modelis (7 pav. Organizacijos struktūros modelis). Priklausau informacinių verslų skyriaus darbuotojų grupei, atsakingai už sistemos kūrimą.



7 pav. Organizacijos struktūros modelis

3.1.2. Apribojimai reikalavimams

3.1.2.1. Apribojimai sprendimui

Aktyvumo vektorių pradinis duomenų failas generuojamas remiantis pateiktu „Susietumu“ failu, kuriame pateikti susietumai, t.y. nuo kokių įėjimų priklauso išėjimai.

“Susietumų” duomenų failo struktūra:

Schemos vardas - 8 baitai <vardas>.

Išplėtimas – 4 baitai <tr>.

Išėjimų kiekis – 4 baitai <xxxxxxxx>.

Įėjimų kiekis – 4 baitai <xxxxxxxx>.

Išėjimų apdorojimo pasikliautinumo skaičiai – po 4 baitus < pasikliautinumo dydis – 4 baitai>; Viso m*4 baitų, kur m – išėjimų skaičius.

Išėjimų/įėjimų susietumas – po 9 baitus, <išėjimo numeris – 8 baitai, įėjimo numeris – 8 baitai, lygiškumas – 1 baitas>

Aktyvumo vektorių paieškos programa rašoma C++ programavimo kalba, o jos rezultatų failo struktūros reikalavimai:

Schemos vardas – 8 baitai;

Išplėtimas – 4 baitai;

Išėjimų apdorojimo pasikliautinumo skaičiai – po 4 baitus; Viso m*4 baitų, kur m – išėjimų skaičius.

Aktyvumo vektorių įrašai - įrašo ilgis – 2 baitai, išėjimo numeris – 4 baitai, funkcijos tipas – 1 baitas, aktyvumo vektoriaus reikšmėms 0,1,V,N po vieną baitą.

Funkcijos tipas (1 baitas) tai:

0 –atvirkštinės funkcijos aktyvumo vektorius

1 – tiesioginės funkcijos aktyvumo vektorius

2 – įėjimų susietumo su išėjimais svoris

Naujai rasti aktyvumo vektoriai papildo pateiktą duomenų failą.

3.1.3. Komunikuojančios sistemos

Pagrindinės komunikuojančios su būsima eksperimentavimo paketu sistemos yra šios:

Sqlite duomenų bazė;

Perl interpretatorius;

3.1.4. Numatoma darbo vietos aplinka

Darbo aplinka: aplinka, kurioje yra tinkama techninė bei programinė įranga;

3.1.5. Svarbūs faktai ir prielaidos

3.1.5.1. Svarbūs faktai

Audrius Kažukauskas yra parašęs įėjimo bei išėjimo susietumams nustatyti programinę įrangą. Programa remiasi pateikiamos schemos funkciniu aprašymu C++ kalboje.

3.1.5.2. Prielaidos

Audrius Kažukausko parašyta įėjimo bei išėjimo susietumams nustatyti programinė įranga formuoja teisingus rezultatus.

Apribodami randamus aktyvumo vektorių skaičiumi, kurį nustato numatyta svorių formulė, gauti iš šių aktyvumo vektorių testai bus kokybiški.

Sistema gali būti pritaikoma darbui Linux operacinės sistemos aplinkoje.

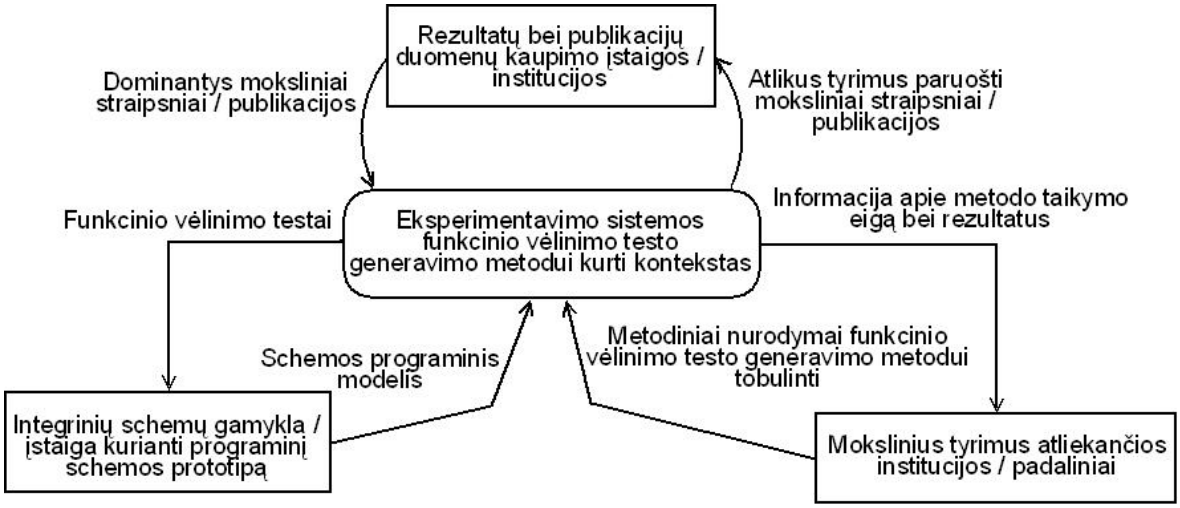
Sistemą gali prireikti plėtoti taip, kad galima būtų pritaikyti skaičiavimams

paskirstytų tinklų („Grid“) skaičiavimuose.

3.1.6. Veiklos sudėtis

3.1.6.1. Veiklos kontekstas

Išanalizavus būsimos sistemos santykį su kitomis įstaigomis, buvo sudaryta veiklos konteksto diagrama (8 pav. Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti konteksto diagrama).



8 pav. Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti konteksto diagrama

3.1.6.2. Veiklos padalinimas

17 lentelėje pateikiamas eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti veiklos įvykių sąrašas

17 lentelė. Veiklos padalinimas

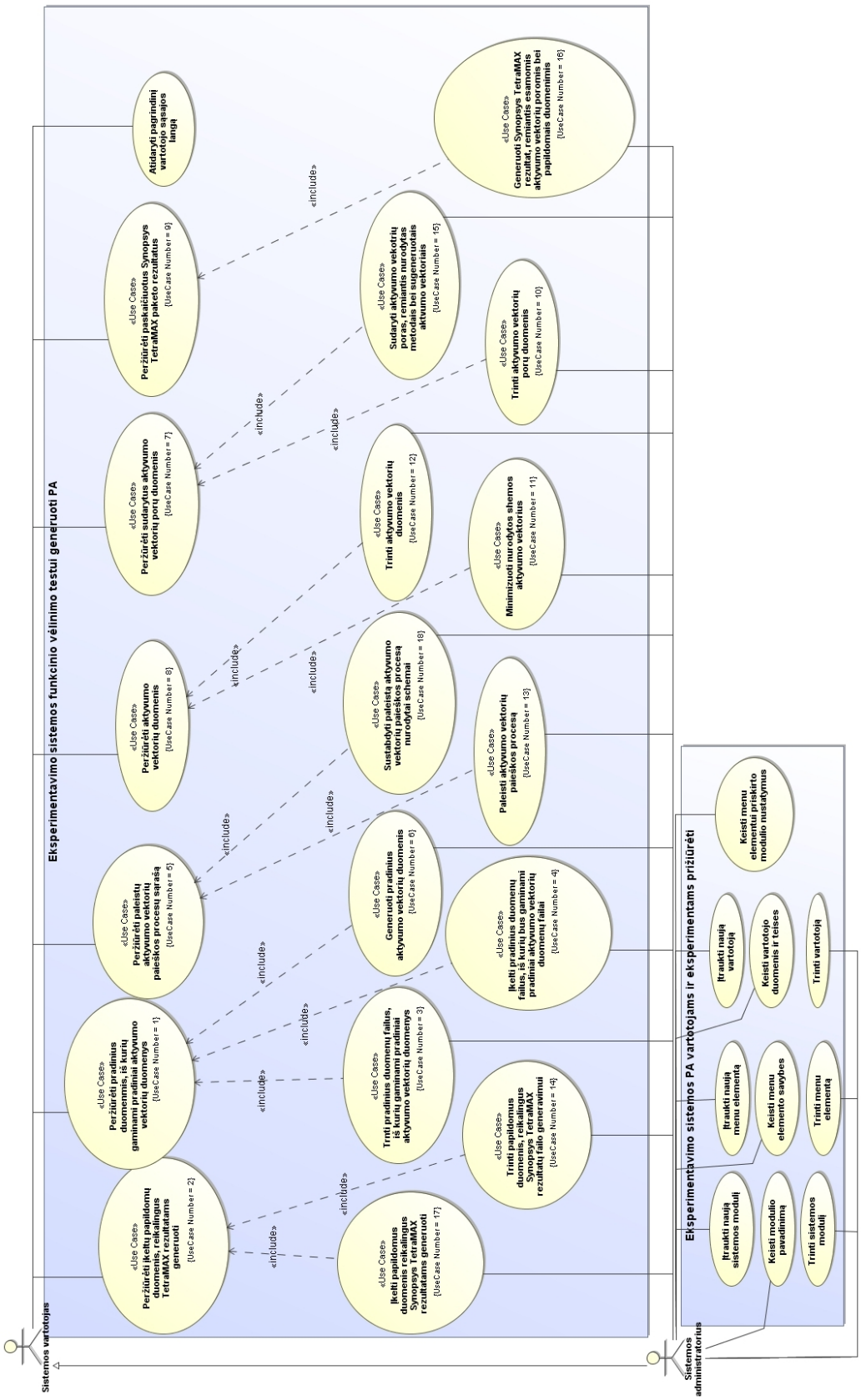
Eil. nr.	Įvykio pavadinimas	Įeinantys (in) / išeinantys (out) informacijos srautai
1	Sistemos vartotojas peržiūri paleistų procesų sąrašą	Paleistų aktyvumo vektorių paieškos programų sąrašas (out)
2	Sistemos vartotojas peržiūri pasirinktos schemos rastus bei minimizuotus aktyvumo vektorius	Aktyvumo vektorių sąrašas (out)
3	Sistemos vartotojas peržiūri pasirinktos	Informacija apie sudarytas aktyvumo

	schemos bei metodo sudarytas aktyvumo vektorių poras	vektorių poras (out)
4	Sistemos vartotojas peržiūri paskaičiuotus Synopsys TetraMAX paketo rezultatų failus	Synopsys TetraMAX programos rezultai (out)
5	Sistemos kūrėjas generuoja pradinius aktyvumo vektorių duomenų failus	Audriaus Kažukausko programinės įrangos, randančios schemos įėjimų / išėjimų susietumus duomenų failas (in)
6	Sistemos kūrėjas minimizuoja nurodytos schemos aktyvumo vektorius	Schemos pavadinimas (in)
7	Sistemos kūrėjas sudaro aktyvumo vektorių poras, remiantis numatytais metodais bei esamais aktyvumo vektoriais	Schemos pavadinimas (in)
8	Sistemos kūrėjas generuoja Synopsis TetraMAX rezultatų failus, remiantis esamomis aktyvumo vektorių poromis bei papildomais failais	Schemos pavadinimas (in)
9	Sistemos kūrėjas paleidžia aktyvumo vektorių paieškos procesą	Schemos pavadinimas, išėjimai, kuriems ieškoti aktyvumo vektorių (in)
10	Sistemos kūrėjas sustabdo paleistą aktyvumo vektorių paieškos procesą	Schemos pavadinimas (in)
11	Sistemos vartotojas peržiūri jau įkeltų schemoms papildomų duomenų failų, reikalingų TetraMAX rezultatų failams gauti	Menu parinktis „failai“ (in)
12	Sistemos kūrėjas įkelia papildomus duomenų failus reikalingus TetraMAX rezultatams generuoti	Papildomi duomenų failai bei schemos pavadinimas (in)
13	Sistemos kūrėjas trina papildomus duomenų failus	Schemos pavadinimas bei papildomo duomenų failo pavadinimas (in)

3.1.7. Sistemos sudėtis

3.1.7.1. Sistemos ribos (panaudojimo atvejų modelis)

Sistemos panaudojimo atvejus iliustruoja žemiau diagrama (9 pav. Panaudojimo atvejų diagrama).

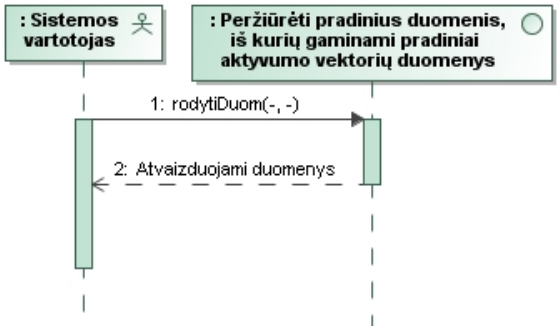


9 pav. Panaudojimo atvejų diagrama

3.1.7.2. Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti panaudos atvejų sąrašas

Peržiūrėti pagrindinius duomenis, iš kurių gaminami pagrindiniai aktyvumo vektorių duomenys.

Aprašas:	Peržiūrėti pagrindinius duomenis, iš kurių gaminami pagrindiniai aktyvumo vektorių duomenys.
Vartotojo/aktoriaus pavadinimas:	Sistemos vartotojas, sistemos kūrėjas.
Tinkamumo kriterijus:	Turi būti pateikta lentelė su papildomų duomenų failų vardais. Pasirinkus failo vardą, turi būti parodomas jo turinys.
Panaudojimo atvejo scenarijus:	Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (10 pav. „Panaudojimo atvejo peržiūrėti pagrindinius duomenis, iš kurių gaminami pagrindiniai aktyvumo vektorių duomenys diagrama“).
Alternatyvus scenarijus:	Nėra.



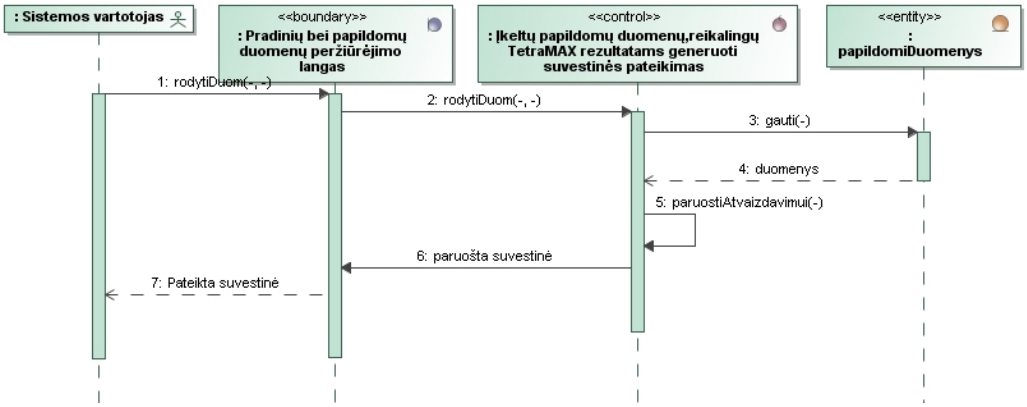
10 pav. „Panaudojimo atvejo peržiūrėti pagrindinius duomenis, iš kurių gaminami pagrindiniai aktyvumo vektorių duomenys diagrama“

Peržiūrėti įkeltus papildomus duomenis, reikalingus TetraMAX rezultatams generuoti.

Aprašas:	Peržiūrėti įkeltus papildomus duomenis, reikalingus TetraMAX rezultatams generuoti.
Vartotojo/aktoriaus pavadinimas:	Sistemos vartotojas, sistemos kūrėjas.
Tinkamumo kriterijus:	Turi būti pateikta lentelė su papildomų duomenų failų vardais. Pasirinkus failą, turi būti parodomas jo turinys.

Panaudojimo atvejo scenarijus: Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (11 pav. Panaudojimo atvejo peržiūrėti įkeltus papildomus duomenis, reikalingus TetraMAX rezultatams generuoti diagrama).

Alternatyvus scenarijus: Nėra.



11 pav. Panaudojimo atvejo peržiūrėti įkeltus papildomus duomenis, reikalingus TetraMAX rezultatams generuoti diagrama

Trinti pradinius duomenis, iš kurių gaminami pradiniai aktyvumo vektorių duomenys.

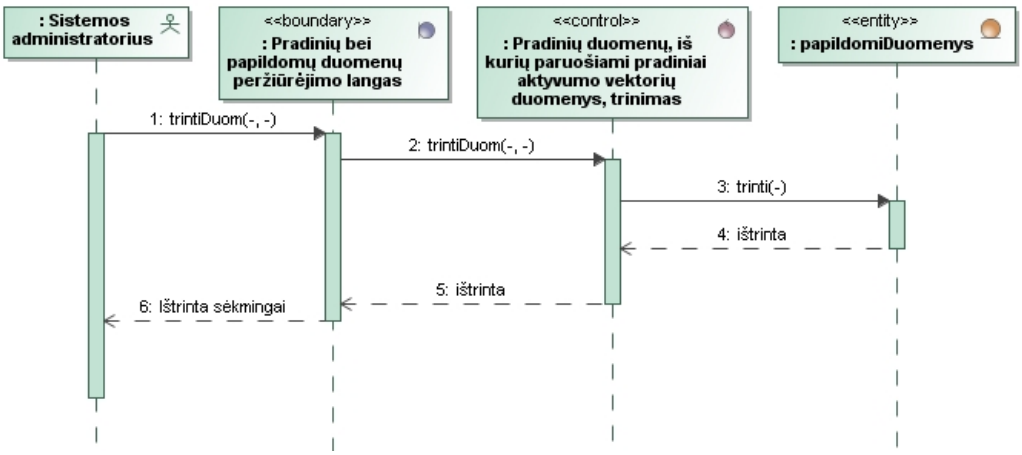
Aprašas: Trinti pradinius duomenis, iš kurių gaminami pradiniai aktyvumo vektorių duomenys.

Vartotojo/aktoriaus pavadinimas: Sistemos kūrėjas.

Tinkamumo kriterijus: Galimybė trinti po vieną failą, visą stulpelį arba eilutę.

Panaudojimo atvejo scenarijus: Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (12 pav. Panaudojimo atvejo trinti pradinius duomenis, iš kurių gaminami pradiniai aktyvumo vektorių duomenys diagrama).

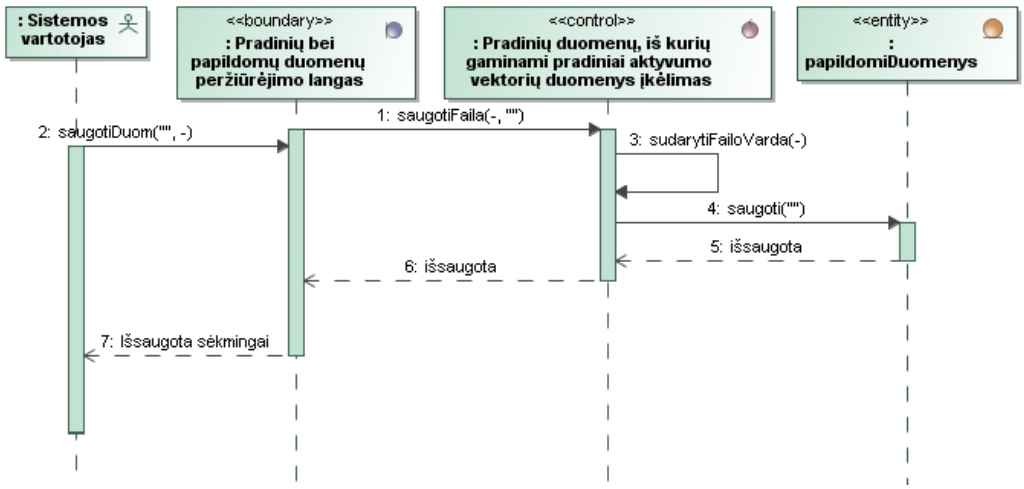
Alternatyvus scenarijus: Nėra.



12 pav. Panaudojimo atvejo trinti pradinius duomenis, iš kurių gaminami pradiniai aktyvumo vektorių duomenys diagrama

Įkelti pradinius duomenis, iš kurių bus gaminami pradiniai aktyvumo vektorių duomenys.

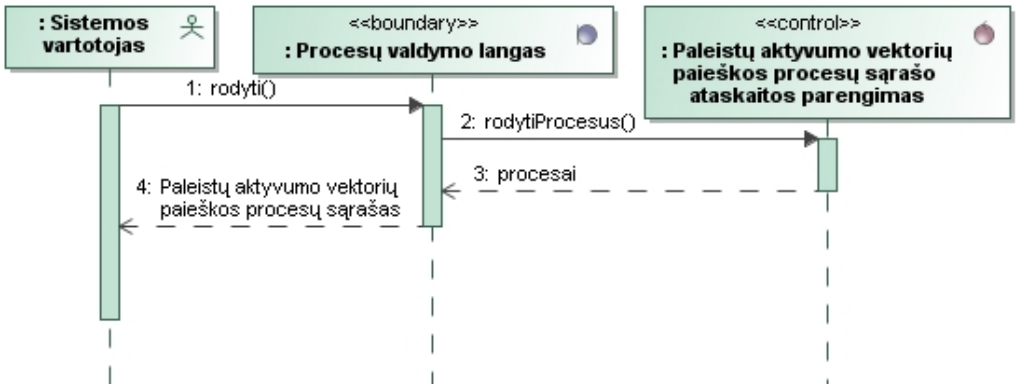
Aprašas:	Įkelti pradinius duomenis, iš kurių bus gaminami pradiniai aktyvumo vektorių duomenys.
Vartotojo/aktoriaus pavadinimas:	Sistemos kūrėjas.
Tinkamumo kriterijus:	Atsiranda duomenų failo pasirinkimo menu, kurio pagalba nurodome duomenų failą ir paspaudus numatytą siuntimui mygtuką, duomenų failas įkeliamas.
Panaudojimo atvejo scenarijus:	Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (13 pav. Panaudojimo atvejo įkelti pradinius duomenis, iš kurių bus gaminami pradiniai aktyvumo vektorių duomenys diagrama).
Alternatyvus scenarijus:	Nėra.



13 pav. Panaudojimo atvejo įkelti pradinius duomenis, iš kurių bus gaminami pradiniai aktyvumo vektorių duomenys diagrama

Peržiūrėti paleistų aktyvumo vektorių paieškos procesų sąrašą.

Aprašas:	Peržiūrėti paleistų aktyvumo vektorių paieškos procesų sąrašą.
Vartotojo/aktoriaus pavadinimas:	Sistemos vartotojas, sistemos kūrėjas.
Tinkamumo kriterijus:	Paleistų procesų sąraše turi būti parašyta su kuria schema procesas susijęs, kiek laiko paskutinis skaičiavimas užtruko bei kurį išėjimą šiuo metu procesas skaičiuoja.
Panaudojimo atvejo scenarijus:	Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (14 pav. Panaudojimo atvejo peržiūrėti paleistų aktyvumo vektorių paieškos procesų sąrašą diagrama).
Alternatyvus scenarijus:	Nėra.



14 pav. Panaudojimo atvejo peržiūrėti paleistų aktyvumo vektorių paieškos procesų sąrašą diagrama

Generuoti pradinius aktyvumo vektorių duomenis.

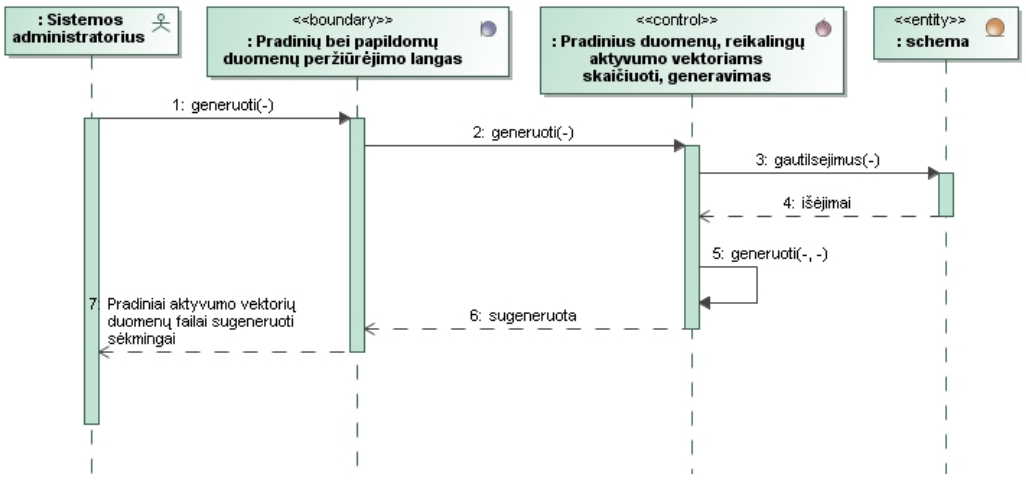
Aprašas: Generuoti pradinius aktyvumo vektorių duomenis., kuriuose apskaičiuojami įėjimams svoriai.

Vartotojo/aktoriaus pavadinimas: Sistemos kūrėjas.

Tinkamumo kriterijus: Sugeneruojamas pradinis aktyvumo vektorių duomenys, kuriuose pasikliautinumai lygūs 0, o svoriai apskaičiuojami pagal formulę. Duomenys nesusideda iš aktyvumo vektorių.

Panaudojimo atvejo scenarijus: Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (15 pav. Panaudojimo atvejo generuoti pradinius aktyvumo vektorių duomenis diagrama).

Alternatyvus scenarijus: Nėra.



15 pav. Panaudojimo atvejo generuoti pradinius aktyvumo vektorių duomenis diagrama

Peržiūrėti sudarytus aktyvumo vektorių porų duomenis.

Aprašas: Peržiūrėti sudarytus aktyvumo vektorių porų duomenis.

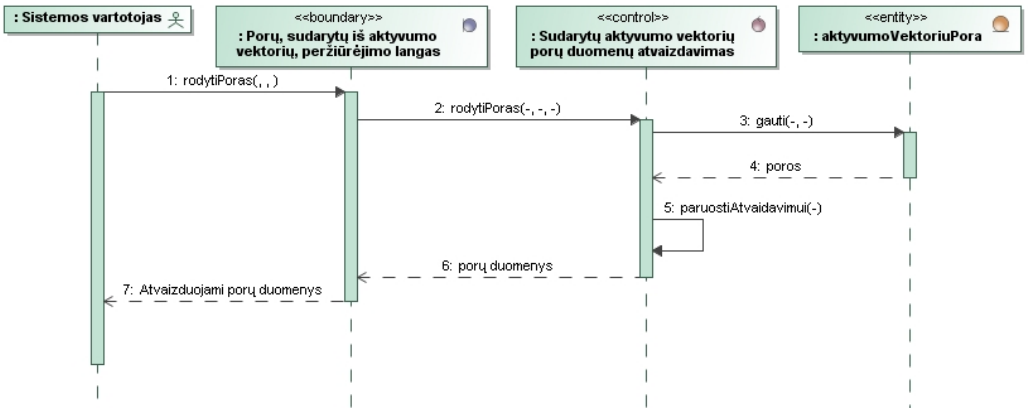
Vartotojo/aktoriaus pavadinimas: Sistemos vartotojas, sistemos kūrėjas.

Tinkamumo kriterijus: Aktyvumo vektorių porų duomenų failai pateikiami lentelėje, suskirstyti pagal metodus. Pasirinkus duomenų

failą, turi būti parodomas jo turinys.

Panaudojimo atvejo scenarijus: Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (16 pav. Panaudojimo atvejo peržiūrėti sudarytus aktyvumo vektorių porų duomenis diagrama).

Alternatyvus scenarijus: Nėra



16 pav. Panaudojimo atvejo peržiūrėti sudarytus aktyvumo vektorių porų duomenis diagrama

Peržiūrėti aktyvumo vektorių duomenis.

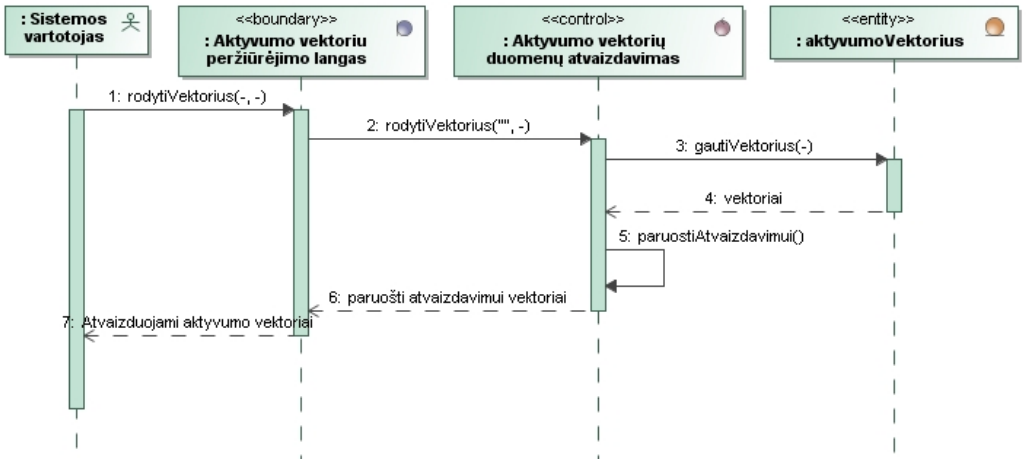
Aprašas: Peržiūrėti aktyvumo vektorių duomenis.

Vartotojo/aktoriaus pavadinimas: Sistemos vartotojas, sistemos kūrėjas.

Tinkamumo kriterijus: Aktyvumo vektorių duomenų failai pateikiami lentelėje. Pasirinkus duomenų failą, turi būti parodomas jo turinys.

Panaudojimo atvejo scenarijus: Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (17 pav. Panaudojimo atvejo peržiūrėti aktyvumo vektorių duomenis diagrama).

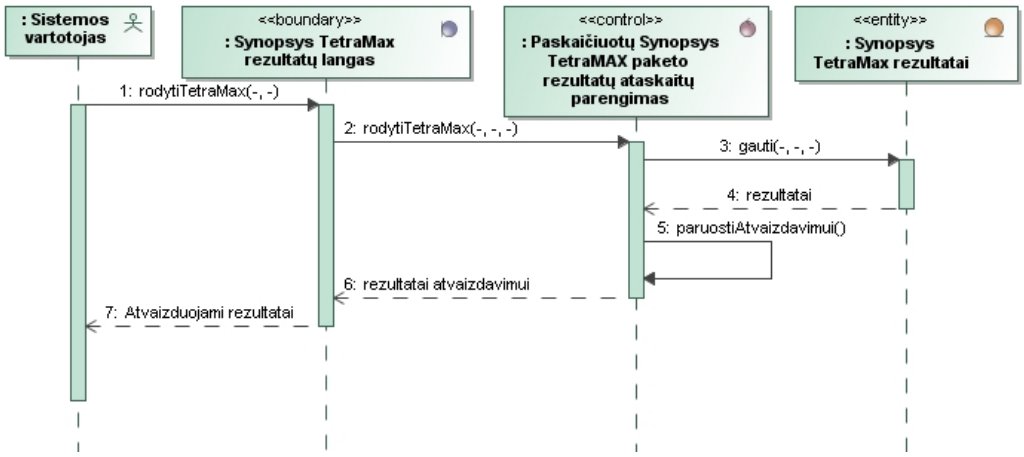
Alternatyvus scenarijus: Nėra



17 pav. Panaudojimo atvejo peržiūrėti aktyvumo vektorių duomenis diagrama

Peržiūrėti paskaičiuotus Synopsys TetraMAX paketo rezultatus.

Aprašas:	Peržiūrėti paskaičiuotus Synopsys TetraMAX paketo rezultatus.
Vartotojo/aktoriaus pavadinimas:	Sistemos vartotojas, sistemos kūrėjas.
Tinkamumo kriterijus:	Synopsys TetraMAX rezultatų failai pateikiami lentelėje, suskirstyti pagal metodus. Pasirinkus rezultatų failą, turi būti parodomas turinys, kuriame atsispindėtų panaudojamų porų bei klaidų padengimo santykis. Taip pat testo ilgis bei funkcinio vėlinimo klaidų padengimas.
Panaudojimo atvejo scenarijus:	Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (18 pav. Panaudojimo atvejo peržiūrėti paskaičiuotus Synopsys TetraMAX paketo rezultatus diagrama).
Alternatyvus scenarijus:	Nėra.



18 pav. Panaudojimo atvejo peržiūrėti paskaičiuotus Synopsys TetraMAX paketo rezultatus diagrama

Trinti aktyvumo vektorių porų duomenis.

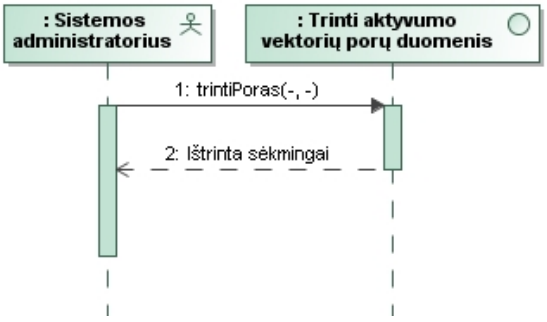
Aprašas: Trinti aktyvumo vektorių porų duomenis.

Vartotojo/aktoriaus pavadinimas: Sistemos kūrėjas.

Tinkamumo kriterijus: Galimybė trinti po vieną failą, visą stulpelį arba eilutę.

Panaudojimo atvejo scenarijus: Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (19 pav. Panaudojimo atvejo trinti aktyvumo vektorių porų duomenis diagrama).

Alternatyvus scenarijus: Nėra.



19 pav. Panaudojimo atvejo trinti aktyvumo vektorių porų duomenis diagrama

Minimizuoti nurodytos schemos aktyvumo vektorius.

Aprašas: Minimizuoti nurodytos schemos aktyvumo vektorius.

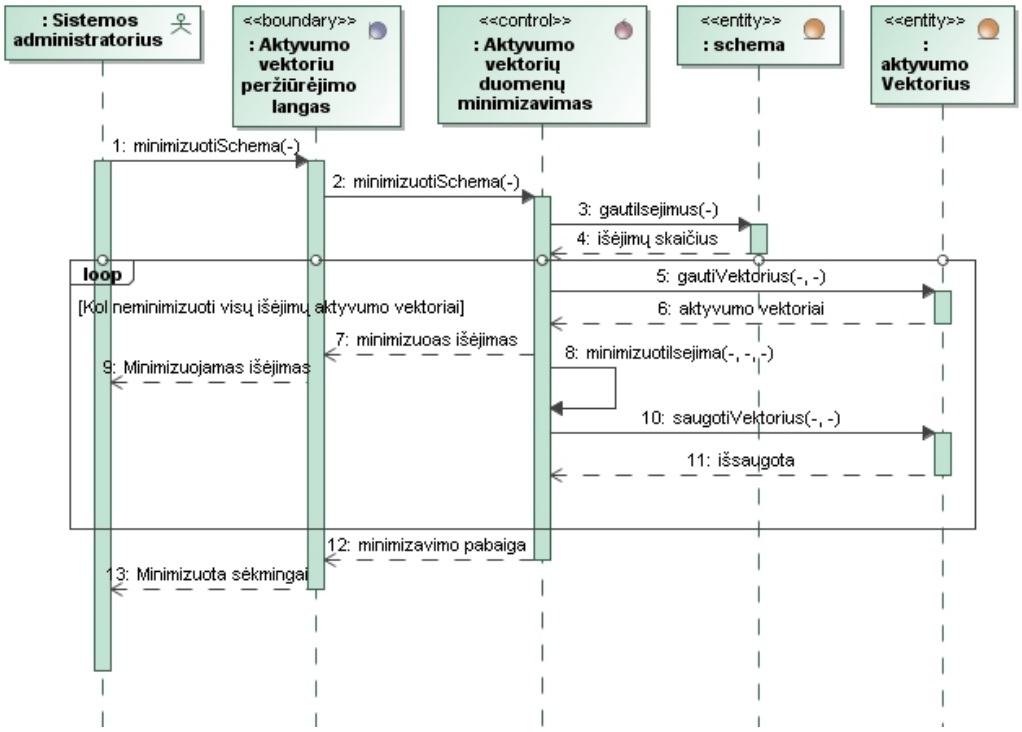
Vartotojo/aktoriaus pavadinimas: Sistemos kūrėjas.

Tinkamumo kriterijus: Sukuriamas atskiras duomenų failas, kuriame surašomi

minimizuoti aktyvumo vektoriai. Duomenų failo reikalavimai tokie patys, kaip ir aktyvumo vektorių duomenų failo.

Panaudojimo atvejo scenarijus: Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (20 pav. Panaudojimo atvejo minimizuoti nurodytos schemos aktyvumo vektorius diagrama).

Alternatyvus scenarijus: Nėra



20 pav. Panaudojimo atvejo minimizuoti nurodytos schemos aktyvumo vektorius diagrama

Trinti aktyvumo vektorių duomenis.

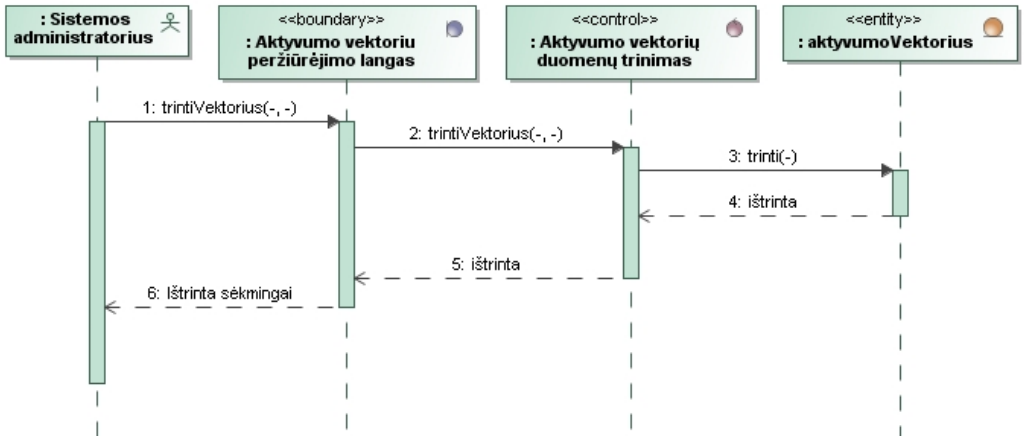
Aprašas: Trinti aktyvumo vektorių duomenis.

Vartotojo/aktoriaus pavadinimas: Sistemos kūrėjas.

Tinkamumo kriterijus: Galimybė trinti po vieną failą, visą stulpelį arba eilutę.

Panaudojimo atvejo scenarijus: Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (21 pav. Panaudojimo atvejo trinti aktyvumo vektorių duomenis diagrama).

Alternatyvus scenarijus: Nėra



21 pav. Panaudojimo atvejo trinti aktyvumo vektorių duomenis diagrama

Paleisti aktyvumo vektorių paieškos procesą.

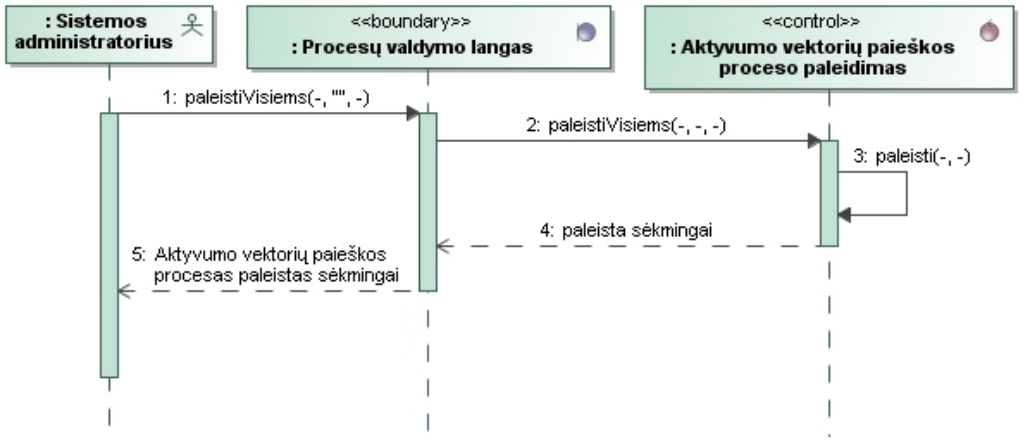
Aprašas: Paleisti aktyvumo vektorių paieškos procesą.

Vartotojo/aktoriaus pavadinimas: Sistemos kūrėjas.

Tinkamumo kriterijus: Procesas paleistas ir sėkmingai atlieka vektorių paieškos procesą nurodytai schemai, nurodytiems išėjimams.

Panaudojimo atvejo scenarijus: Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (22 pav. Panaudojimo atvejo paleisti aktyvumo vektorių paieškos procesą diagrama).

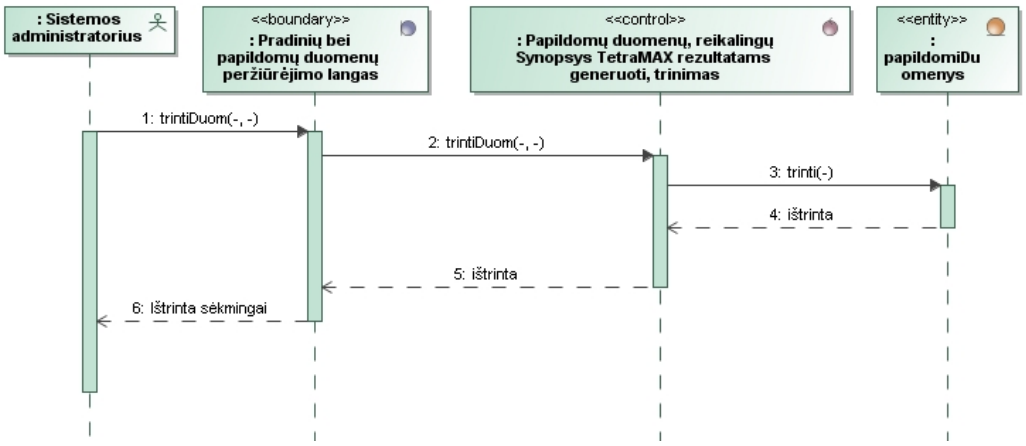
Alternatyvus scenarijus: Nėra.



22 pav. Panaudojimo atvejo paleisti aktyvumo vektorių paieškos procesą diagrama

Trinti papildomus duomenis, reikalingus Synopsys TetraMAX rezultatams generuoti.

Aprašas:	Trinti papildomus duomenis, reikalingus Synopsys TetraMAX rezultatams generuoti. T.y. scr bei testiniai duomenų failai.
Vartotojo/aktoriaus pavadinimas:	Sistemos kūrėjas.
Tinkamumo kriterijus:	Ištrinami nurodytos schemos src bei testiniai failai.
Panaudojimo atvejo scenarijus:	Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (23 pav. Panaudojimo atvejo trinti papildomus duomenis, reikalingus Synopsys TetraMAX rezultatams generuoti diagrama).
Alternatyvus scenarijus:	Nėra.

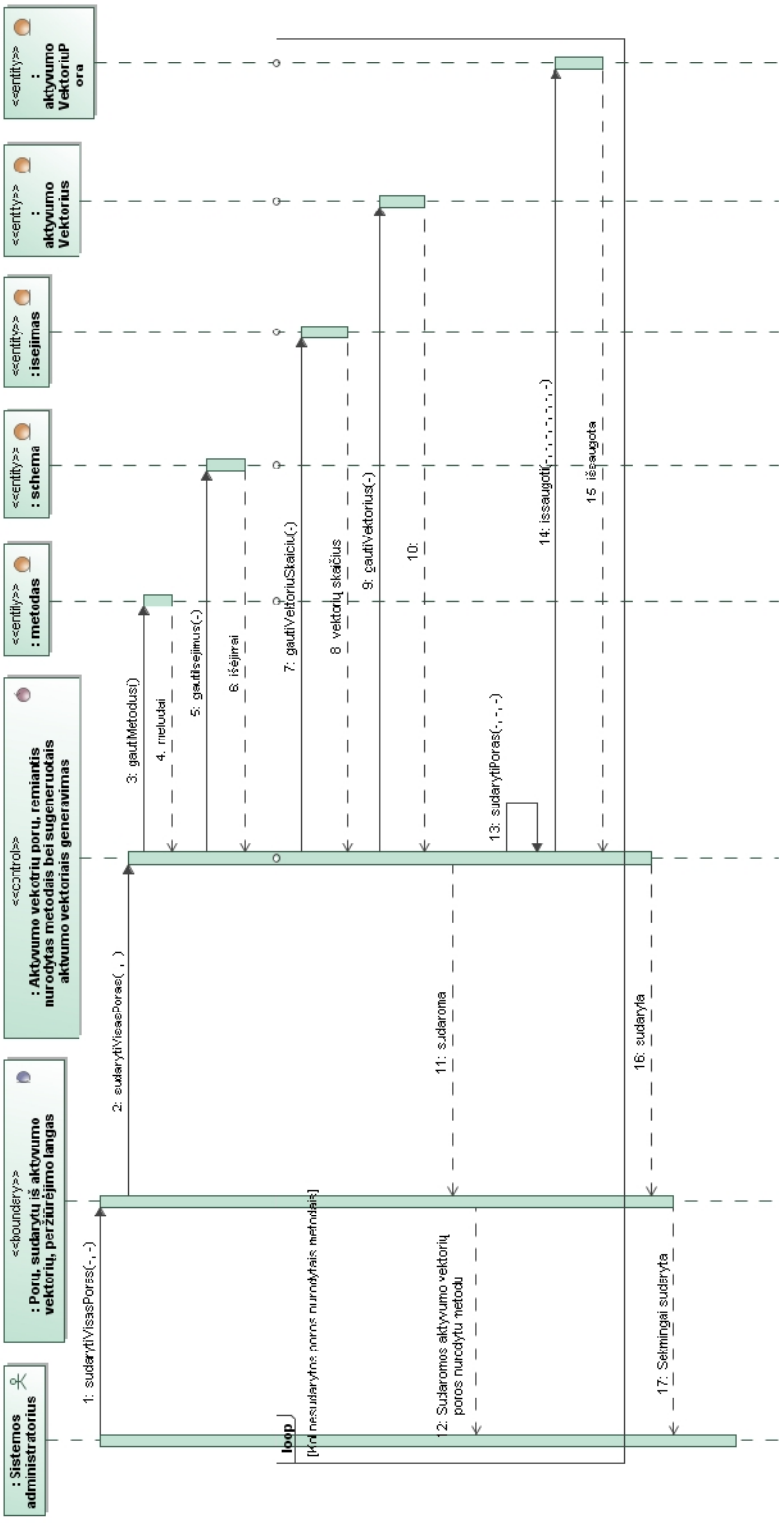


23 pav. Panaudojimo atvejo trinti papildomus duomenis, reikalingus Synopsys TetraMAX rezultatams generuoti diagrama

Sudaryti aktyvumo vektorių poras, remiantis nurodytais metodais bei sugeneruotais aktyvumo vektoriais.

Aprašas:	Sudaryti aktyvumo vektorių poras, remiantis nurodytais metodais bei sugeneruotais aktyvumo vektoriais.
Vartotojo/aktoriaus pavadinimas:	Sistemos kūrėjas.
Tinkamumo kriterijus:	Atskiruose failuose sugeneruojamos aktyvumo vektorių poros, remiantis nurodytais metodais iš jau esamų nurodytos schemos aktyvumo vektorių.

Panaudojimo atvejo scenarijus:	Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (24 pav. Panaudojimo atvejo sudaryti aktyvumo vektorių poras, remiantis nurodytais metodais bei sugeneruotais aktyvumo vektoriais diagrama).
Alternatyvus scenarijus:	Nėra.

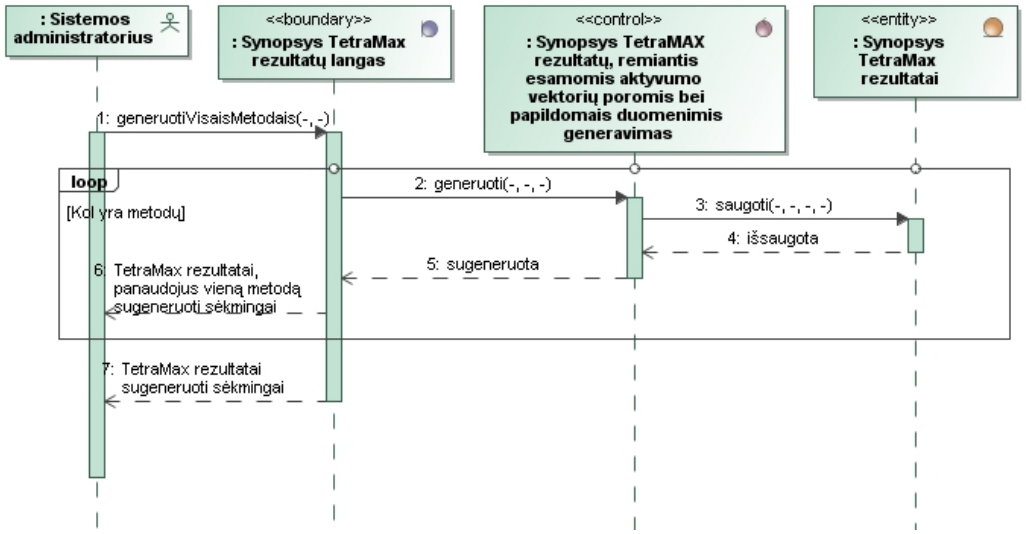


24 pav. Panaudojimo atvejo sudaryti aktyvumo vektorių poras, remiantis nurodytais metodais bei sugeneruotais aktyvumo vektoriais diagrama

Generuoti Synopsys TetraMAX rezultatus, remiantis esamomis

aktyvumo vektorių poromis bei papildomais duomenimis.

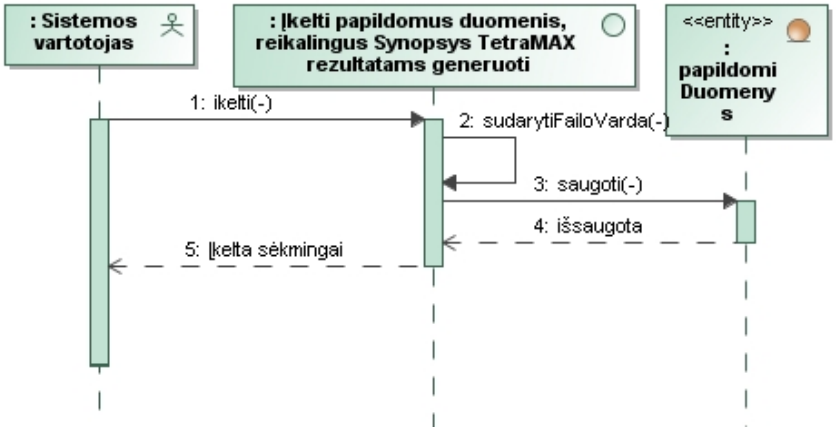
Aprašas:	Generuoti Synopsys TetraMAX rezultatus, remiantis esamomis aktyvumo vektorių poromis bei papildomais duomenimis.
Vartotojo/aktoriaus pavadinimas:	Sistemos kūrėjas.
Tinkamumo kriterijus:	Remiantis esamais failais sugeneruojami reikalingi papildomi failai Synopsys TetraMAX paketui sėkmingai atlikti testavimą, kuriame pateiktų testavimo rezultatus, atspindinčius funkcinio vėlinimo klaidų padengimą.
Panaudojimo atvejo scenarijus:	Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (25 pav. Panaudojimo atvejo generuoti Synopsys TetraMAX rezultatus, remiantis esamomis aktyvumo vektorių poromis bei papildomais duomenimis diagrama).
Alternatyvus scenarijus:	Nėra.



25 pav. Panaudojimo atvejo generuoti Synopsys TetraMAX rezultatus, remiantis esamomis aktyvumo vektorių poromis bei papildomais duomenimis diagrama

Įkelti papildomus duomenis, reikalingus Synopsys TetraMAX rezultatams generuoti.

Aprašas:	Įkelti papildomus duomenis, reikalingus Synopsys TetraMAX rezultatams generuoti. T.y. testų failas bei scr failas.
Vartotojo/aktoriaus pavadinimas:	Sistemos kūrėjas.
Tinkamumo kriterijus:	Atsiranda papildomo duomenų failo pasirinkimo menu, kurio pagalba nurodome papildomą duomenų failą ir paspaudus numatytą siuntimui mygtuką, duomenų failas įkeliamas. Tą patį turi būti galima atlikti ir su kitu papildomu duomenų failu.
Panaudojimo atvejo scenarijus:	Panaudojimo atvejo aprašas pateiktas žemiau esančioje diagramoje (26 pav. Panaudojimo atvejo įkelti papildomus duomenis, reikalingus Synopsys TetraMAX rezultatams generuoti diagrama).
Alternatyvus scenarijus:	Nėra.



26 pav. Panaudojimo atvejo įkelti papildomus duomenis, reikalingus Synopsys TetraMAX rezultatams generuoti diagrama

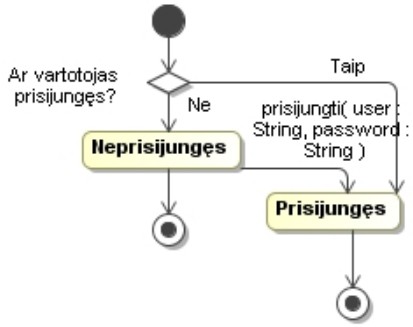
Panaudojus IBM Rational RequisitePro paketą, atliktas vartotojų ir panaudojimo atvejų susiejimas. 1 priede pateikta vartotojų ir panaudojimo atvejų susietumo matrica.

3.1.7.3. Sistemos elgsenos būsenų diagramos

Kad geriau suprasti sistemos elgseną, buvo sudarytos sistemos būsenų diagramos, kurios pateiktos 27 pav. - 31 pav.

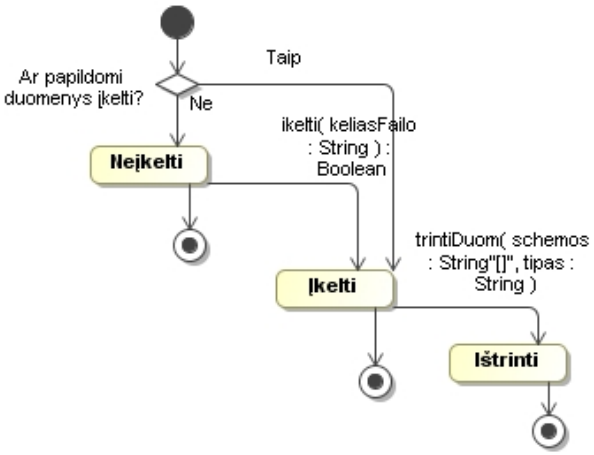
27 pav. (Vartotojo būsenų diagrama) pateikta vartotojo būsenų diagrama,

vaizduojanti vartotojo būsenos kaitą, kaip pateikus identifikacinius duomenis iš neprisijungusio vartotojo jis gali įgyti prisijungusio vartotojo būseną.



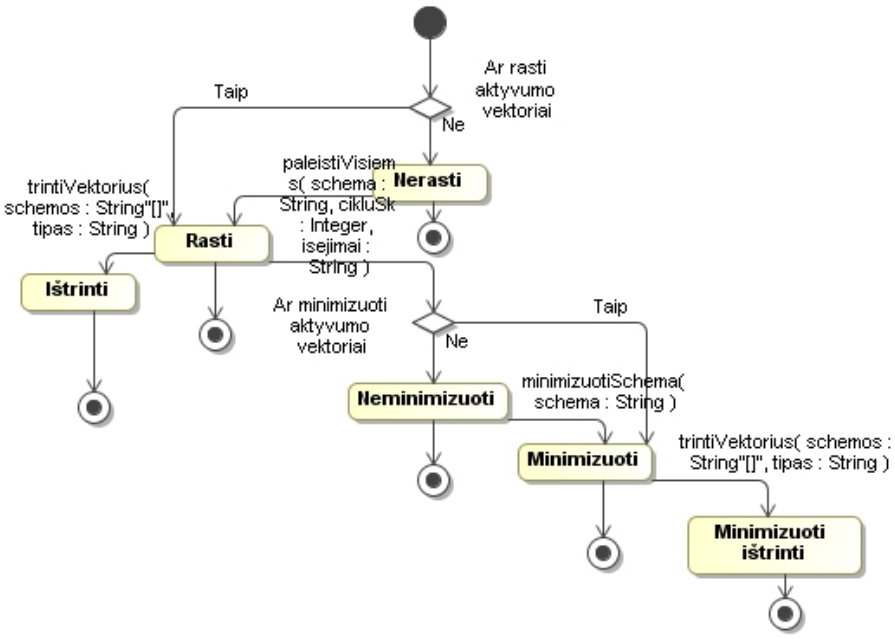
27 pav. Vartotojo būsenų diagrama

Žemiau esanti diagrama (28 pav. Papildomų duomenų būsenų diagrama) vaizduoja papildomų duomenų failų būsenų diagramą, kuomet papildomi duomenys gali būti būsenose: neįkelti, įkelti, o paskui ir ištrinti. Analogiška būsenų kaitų diagrama yra ir pradinių duomenų, reikalingų ieškoti aktyvumo vektorius ir papildomų duomenų reikalingų klaidų simuliacijai atlikti.



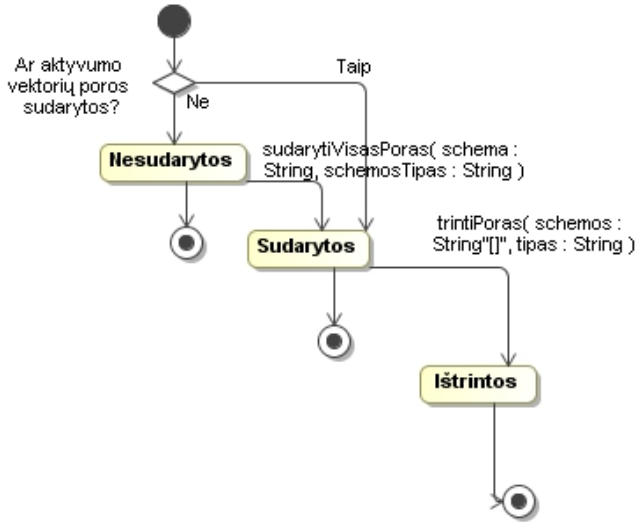
28 pav. Papildomų duomenų būsenų diagrama

29 pav. (Aktyvumo vektorių būsenų diagrama) pateikta aktyvumo vektorių būsenų kaitos diagrama, vaizduojanti kaip būdami aktyvumo vektoriai būsenoje „nerasti“, praėję būsenas: rasti, neminimizuoti, minimizuoti gali įgyti būseną „minimizuoti ištrinti“. Taip pat vaizduoja ir kitas galimas aktyvumo vektorių būsenas.



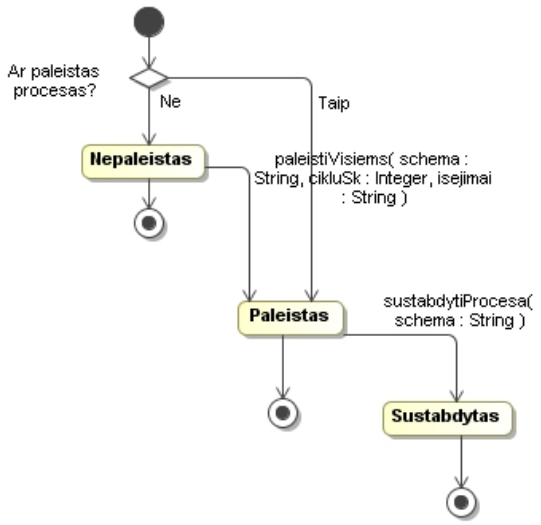
29 pav. Aktyvumo vektorių būsenų diagrama

Aktyvumo vektorių poros iš pradžių būna būsenoje „nesudarytos“, ir iš jos gali patekti į „sudarytos“, galiausiai „ištrintos“. Tai vaizduoja 30 pav. pateikta „Aktyvumo vektorių porų būsenų diagrama“.



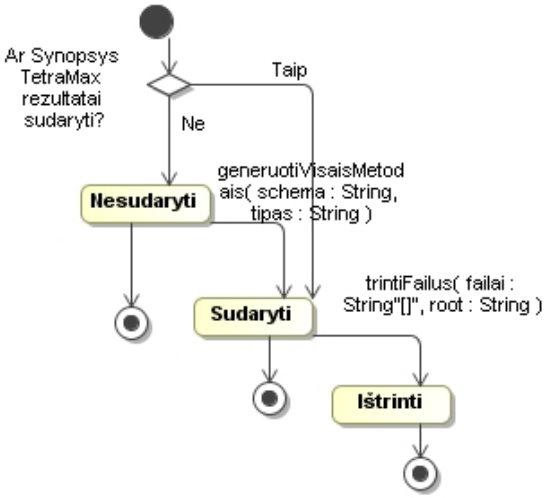
30 pav. Aktyvumo vektorių porų būsenų diagrama

Procesas gali būti būsenose: nepaleistas, paleistas bei sustabdytas. Tai ir vaizduoja žemiau esanti būsenų diagrama (31 pav. Procesų būsenų diagrama).



31 pav. Procesų būsenų diagrama

Synopsys TetraMAX rezultatai gali būti trijose būsenose taip pat: nesudaryti, sudaryti ir ištrinti. Tai pavaizduota žemiau esančioje diagramoje (32 pav. Synopsys TetraMAX rezultatų būsenų diagrama).



32 pav. Synopsys TetraMAX rezultatų būsenų diagrama

3.1.8. Funkciniai reikalavimai ir reikalavimai duomenims

3.1.8.1. Funkciniai reikalavimai

Reikalavimas #:	1	Reikalavimo tipas:	9	Įvykis/PA #:	1, 2, 7, 8, 9
Aprašymas:	Visose duomenų failų peržiūros lentelėse, turi būti atliekamas				

	rikiavimas pagal kokią nors vienareikšmę taisyklę.		
Pagrindimas:	Kad vartotojas lengviau ir greičiau atrastų reikiamą informaciją lentelėje.		
Šaltinis:	Sistemos vartotojas, sistemos kūrėjas.		
Tikimo kriterijus:	Visose duomenų failų peržiūros lentelėse, atliekamas rikiavimas pagal kokią nors vienareikšmę taisyklę.		
Užsakovo tenkinimas:	4	Užsakovo netenkinimas:	5
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:			

Reikalavimas #:	2	Reikalavimo tipas:	9	Įvykis/PA #:	8
Aprašymas:	Pateikto aktyvumo vektorių lentelėje, turi būti galimybė pasirinkti, kaip norima peržiūrėti aktyvumo vektorius: su antraštėmis ar be antraščių.				
Pagrindimas:	Jei vartotojas norės atsisiųsti aktyvumo vektorių failą ir patikrinti rezultatus, jam patogiau bus turėti duomenų failą su antraštėmis. Jei norės pats atlikti su pateiktu aktyvumo vektorių failu eksperimentus, tuomet jam patogiau bus turėti failą be antraščių.				
Šaltinis:	Sistemos vartotojas.				
Tikimo kriterijus:	Aktyvumo vektorių duomenų failų lentelėje ne minimizuotiems ir minimizuotiems aktyvumo vektoriams išskiriami atskiri stulpeliai, kuriuose galima pasirinkti norimą nurodytos schemos duomenų atvaizdavimo būdą.				
Užsakovo tenkinimas:	5	Užsakovo netenkinimas:	3		
Priklausomybės:	Nėra.	Konfliktai:	Nėra.		
Papildoma medžiaga:	Nėra.				
Istorija:					

Reikalavimas #:	3	Reikalavimo tipas:	9	Įvykis/PA #:	8
Aprašymas:	Peržiūrint aktyvumo vektorių bet kokiu būdų (su antraštėm arba be antraščių) skirtingomis spalvomis išskiriamos aktyvios reikšmės.				
Pagrindimas:	Išskirtos spalvomis aktyvios reikšmės suteikia papildomą informaciją apie schemą, ir tam tikrais atvejais galima vizualiai įžvelgti rezultatų prasmę.				
Šaltinis:	Sistemos vartotojas.				
Tikimo kriterijus:	Pasirinkus peržiūrėti norimos schemos norimus aktyvumo vektorius pageidaujamu būdu, aktyvios reikšmės išskiriamos skirtingomis spalvomis.				
Užsakovo tenkinimas:	4		Užsakovo netenkinimas:		2
Priklausomybės:	Nėra.		Konfliktai:		Nėra.
Papildoma medžiaga:	Nėra.				
Istorija:					

Reikalavimas #:	4	Reikalavimo tipas:	9	Įvykis/PA #:	7
Aprašymas:	Pateiktos aktyvumo vektorių porų lentelės duomenys turi būti suskirstyti pagal panaudotus metodus.				
Pagrindimas:	Jei sugeneruotos aktyvumo vektorių poros nesusietos su metodu, tuomet beveik nėra jokios naudos iš tokių duomenų.				
Šaltinis:	Sistemos vartotojas.				
Tikimo kriterijus:	Aktyvumo vektorių porų failai pateikti lentelėje bei suskirstyti pagal panaudotus metodus.				
Užsakovo tenkinimas:	5		Užsakovo netenkinimas:		5
Priklausomybės:	Nėra.		Konfliktai:		Nėra.
Papildoma medžiaga:	Nėra.				
Istorija:					

Reikalavimas #:	5	Reikalavimo tipas:	9	Įvykis/PA #:	9
Aprašymas:	Synopsys TetraMAX rezultatų duomenų lentelėje turi būti funkcinio vėlinimo klaidų padengimas išreikštas procentais bei testo dydis.				
Pagrindimas:	Kuomet yra daug sugeneruotų Synopsys TetraMAX rezultatų duomenų failų, šių duomenų tikrinimas kiekviename užtruktų ir laiko ir blaškytų, sunkiau padaryti rezultatų išvadas.				
Šaltinis:	Sistemos vartotojas.				
Tikimo kriterijus:	Synopsys TetraMAX rezultatų duomenų lentelėje pateikta funkcinio vėlinimo klaidų padengimas išreikštas procentais bei testo dydis				
Užsakovo tenkinimas:	5	Užsakovo netenkinimas:	5		
Priklausomybės:	Nėra.	Konfliktai:	Nėra.		
Papildoma medžiaga:	Nėra.				
Istorija:					

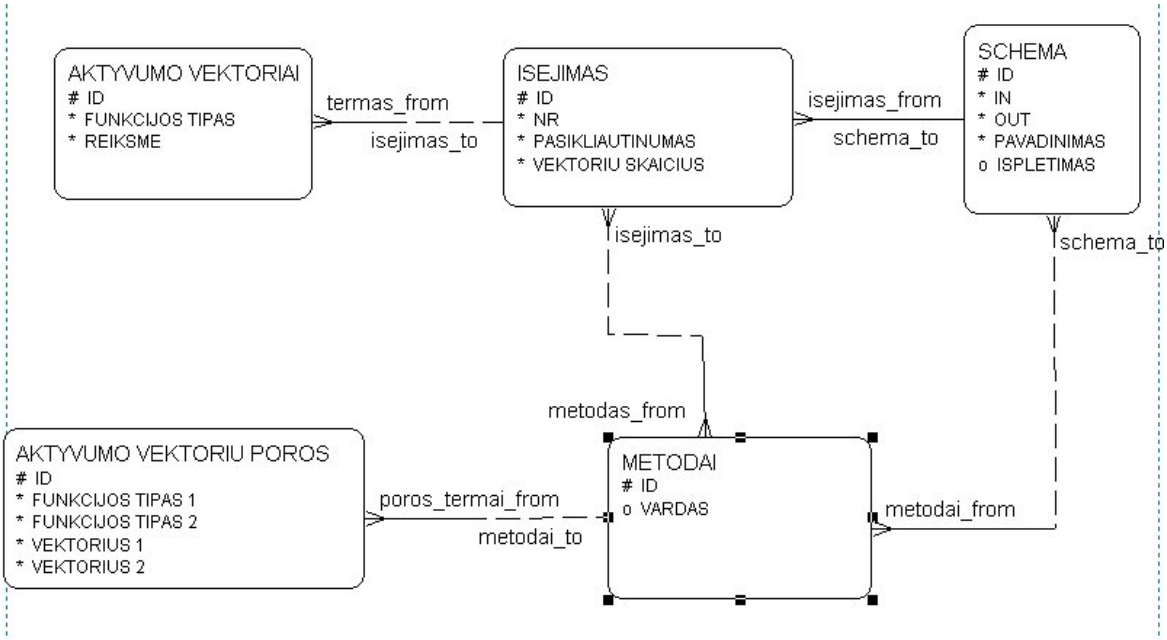
Reikalavimas #:	6	Reikalavimo tipas:	9	Įvykis/PA #:	1, 2, 7, 8, 9
Aprašymas:	Paspaudus ant lentelėje esančio egzistuojančio duomenų failo žymeklio, turi būti parodomas duomenų turinys.				
Pagrindimas:	Kartais gali prireikti paanalizuoti duomenų failus, ne tik galutinį rezultatą.				
Šaltinis:	Sistemos vartotojas, sistemos kūrėjas.				
Tikimo kriterijus:	Paspaudus ant bet kurio lentelėje pateikto duomenų failo žymeklio, parodomas jo turinys.				
Užsakovo tenkinimas:	4	Užsakovo netenkinimas:	4		
Priklausomybės:	Nėra.	Konfliktai:	Nėra.		
Papildoma medžiaga:	Nėra.				
Istorija:					

Reikalavimas #:	7	Reikalavimo tipas:	9	Įvykis/PA #:	11
Aprašymas:	Sistema turi neleisti minimizuoti aktyvumo vektorių duomenų failo, jeigu yra paleistas procesas susijęs su juo.				
Pagrindimas:	Paleistas procesas gali rasti naujų aktyvumo vektorių, kurie turi įtakos minimizavimo algoritmui				
Šaltinis:	Sistemos kūrėjas.				
Tikimo kriterijus:	Vietoje užrašo, išreiškiančio galimybę minimizuoti aktyvumo vektorius, turi būti pranešantis užrašas, jog yra šiuo metu praleistas procesas.				
Užsakovo tenkinimas:	3	Užsakovo netenkinimas:	2		
Priklausomybės:	Nėra.	Konfliktai:	Nėra.		
Papildoma medžiaga:	Nėra.				
Istorija:					

Panaudojus IBM Rational RequisitePro paketą, priskirtos atributų reikšmės. 2 priede pateikta sugeneruota svarbiausių funkcinių reikalavimų ataskaita (atributų matrica), kurioje išrikiuoti reikalavimai pagal užsakovo patenkinimą. Antras rikiavimo kriterijus – užsakovo netenkinimas.

3.1.8.2. Reikalavimai duomenims

33 paveiksle pateikta sistemos pradinė duomenų modelio ER diagrama.



33 pav. „Pradinė duomenų modelio ER diagrama“

3.1.9. Nefunkciniai reikalavimai

3.1.9.1. Reikalavimai sistemos išvaizdai

Reikalavimas #:	8	Reikalavimo tipas:	10	Įvykis/PA #:	1-17
Aprašymas:	Kiek galima intuityvesnė bei lengvai perprantama vartotojo sąsaja.				
Pagrindimas:	Vartotojui svarbiausiai galimybė analizuoti rezultatus, todėl jam užtektų elementarios, paprastos, lengvai perprantamos sąsajos su žymėjimais, kuo tiksliau apibūdinančiais sistemos būseną.				
Šaltinis:	Sistemos vartotojas.				
Tikimo kriterijus:	Duomenų bei rezultatų failai suskirstyti į grupes ir vaizduojami grupėmis, atskirose lentelėse, turinčiose prasminį pavadinimą.				
Užsakovo tenkinimas:	4	Užsakovo netenkinimas:	3		
Priklausomybės:	Nėra.	Konfliktai:	Nėra.		
Papildoma medžiaga:	Nėra.				
Istorija:					

3.1.9.2. Reikalavimai panaudojimui

Reikalavimas #:	9	Reikalavimo tipas:	11	Įvykis/PA #:	1-17
Aprašymas:	Sistema turi būti nesudėtinga sistemos vartotojui duomenų bei rezultatų failų analizavimui.				
Pagrindimas:	Vartotojui svarbiausiai galimybė analizuoti duomenų bei rezultatų failus, todėl jam svarbiausiai, kad nereiktų atlikti daug operacijų, norint pasiekti pageidaujamus rezultatus analizei.				
Šaltinis:	Sistemos vartotojas.				
Tikimo kriterijus:	Galima pasiekti visas grupes viena užklausa. Duomenų arba rezultatų failo turinį, pasiekus grupę taip pat galima pasiekti viena užklausa.				
Užsakovo tenkinimas:	3	Užsakovo netenkinimas:	4		
Priklausomybės:	Nėra.	Konfliktai:	Nėra.		
Papildoma medžiaga:	Nėra.				
Istorija:					

3.1.9.3. Reikalavimai vykdymo charakteristikoms

Reikalavimas #:	10	Reikalavimo tipas:	12	Įvykis/PA #:	
Aprašymas:	Sistema turi būti prieinama visą parą				
Pagrindimas:	Gali prireikti naudotis sistema bet kuriuo paros metu.				
Šaltinis:	Sistemos vartotojas, sistemos kūrėjas.				
Tikimo kriterijus:	Sistema prieinama visą parą.				
Užsakovo tenkinimas:	4	Užsakovo netenkinimas:	2		
Priklausomybės:	Nėra.	Konfliktai:	Nėra.		
Papildoma medžiaga:	Nėra.				
Istorija:					

3.1.9.4. Reikalavimai saugumui

Reikalavimas #:	11	Reikalavimo tipas:	15	Įvykis/PA #:	3, 4, 6,

			10, 11 - 17
Aprašymas:	Tik sistemos kūrėjas turi teisę dirbti su failais bei procesais.		
Pagrindimas:	Kadangi jis labiausiai nusimato sistemoje, tai jis labiausiai ir gali užtikrinti sklandžią veiklos eigą bei duomenų saugumą tiek nuo išorės, tiek nuo procesų pažeidžiamumo.		
Šaltinis:	Sistemos kūrėjas.		
Tikimo kriterijus:	Niekas, apart sistemos kūrėjo neturi galimybės trinti, kurti, generuoti, paleisti nieko, kas susiję su duomenų bei rezultatų failais, procesais.		
Užsakovo tenkinimas:	5	Užsakovo netenkinimas:	4
Priklausomybės:	Nėra	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:			

3.1.9.5. Kultūriniai – politiniai reikalavimai

Reikalavimas #:	12	Reikalavimo tipas:	16	Įvykis/PA #:	1-17
Aprašymas:	Sistemoje turi būti naudojama taisyklinga kalba.				
Pagrindimas:	Kadangi moksliniams tikslams kuriama sistema, tai kalba turi būti natūrali be žargonų, įžeidimų ar kitų kalbos detalių, teršiančių ją.				
Šaltinis:	Sistemos vartotojas, sistemos kūrėjas.				
Tikimo kriterijus:	Sistemoje vartojama taisyklinga kalba, kurioje nėra žargonų, įžeidimų ar kitų kalbos detalių galinčių ją teršti.				
Užsakovo tenkinimas:	1	Užsakovo netenkinimas:	5		
Priklausomybės:	Nėra.	Konfliktai:	Nėra.		
Papildoma medžiaga:	Nėra.				
Istorija:					

3.1.10. Atviri klausimai

Kai kurie rezultatų failai užima kelis šimtus megabaitų vietos diske, tikėtina, kad sudėtingesnių schemų duomenys gali užimti dar daugiau ir jų dydis bus matuojamas gigabaitais.

Sudėtingesnių schemų aktyvumo vektorių paieškos algoritmas vienam išėjimui užtrunka labai ilgai, galbūt pereiti prie paskirstytų tinklinių skaičiavimų?

3.1.11. Naujos problemos

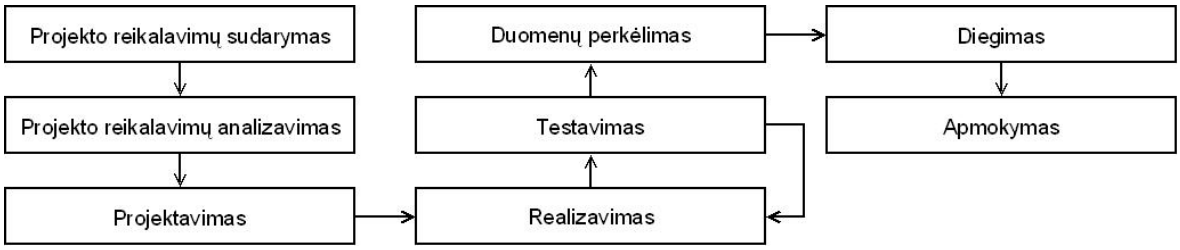
3.1.11.1. Naujos sistemos poveikis jau įdiegtoms sistemoms

Gali tekti kitoms sistemoms dalintis kompiuteriniais resursais su diegiamąja sistema.

3.1.12. Uždaviniai

3.1.12.1. Žingsniai reikalingi sistemai pateikti

34 paveikslėlyje pateikti sistemos kūrimo etapai.



34 pav. Sistemos kūrimo etapų diagrama

3.1.12.2. Vystymo etapai

Žemiau esančioje lentelėje (18 lentelė. Vystymo etapai) pateiktos sistemos kūrimo etapų laikinė ir finansinės sąnaudos.

18 lentelė. Vystymo etapai

Sistemos vystymo fazė (etapas)	Laiko sąnaudos, %	Finansinės sąnaudos, %
Projekto reikalavimų sudarymas	15	10
Projekto reikalavimų analizavimas	15	10
Projektavimas	25	20
Realizavimas	30	34
Testavimas	8	20

Duomenų perkėlimas	1	0
Diegimas	4	6
Apmokymas	2	1

3.1.13. Pritaikymas

3.1.13.1. Specialūs reikalavimai, esamiems duomenims „paimti“ bei procedūroms pritaikyti darbui su nauja sistema

Žemiau esančioje lentelėje (19 lentelė. Specialūs reikalavimai) pateikiamas pritaikymo veiklų sąrašas bei jų realizavimo grafikas

19 lentelė. Specialūs reikalavimai

Veiklos pavadinimas	Veiklos realizavimas
C++ kompiliatoriaus įdiegimas	Įgyvendinta
Perl interpretatoriaus įdiegimas	Įgyvendinta
Apache serviso įdiegimas	Įgyvendinta
Prisijungimo prie tarnybinės stoties serviso paruošimas	Įgyvendinta
Synopsys TetraMAX paketo tarnybinėje stotyje įdiegimas	Įgyvendinta

Duomenų perkėlimo darbams nereikia papildomos programinės įrangos. Ir jokie duomenys neturės būti transformuoti perkeliant į naują sistemą.

3.1.14. Kaina

Kadangi sistema kuriama neribotą laiką, mokant minimalų valstybės numatytą mėnesinį atlyginimą, numatyti sistemos kūrimo etapų kainas yra neįmanoma.

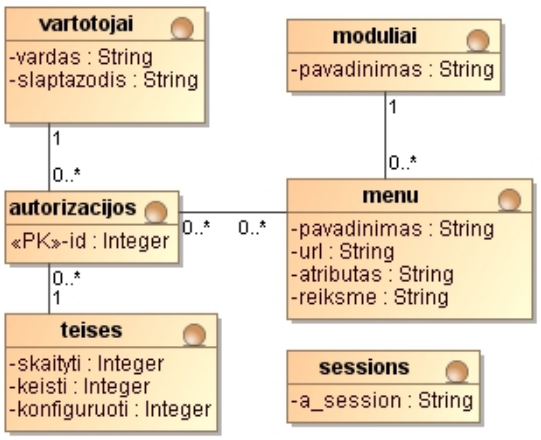
3.1.15. Perspektyviniai reikalavimai

Reikalavimas #:	13	Reikalavimo tipas:	17	Įvykis/PA #:	13
Aprašymas:	Sistema bus pritaikoma skaičiavimais paskirstytuose tinkluose atlikti („Grid“)				
Pagrindimas:	Aktyvumo vektorių paieška sudėtingom schemom trunka labai ilgai, todėl, kad sutaupyti laiką, reikėtų paskirstyti skaičiavimus keliems kompiuteriams.				
Šaltinis:	Sistemos vartotojas, sistemos kūrėjas.				

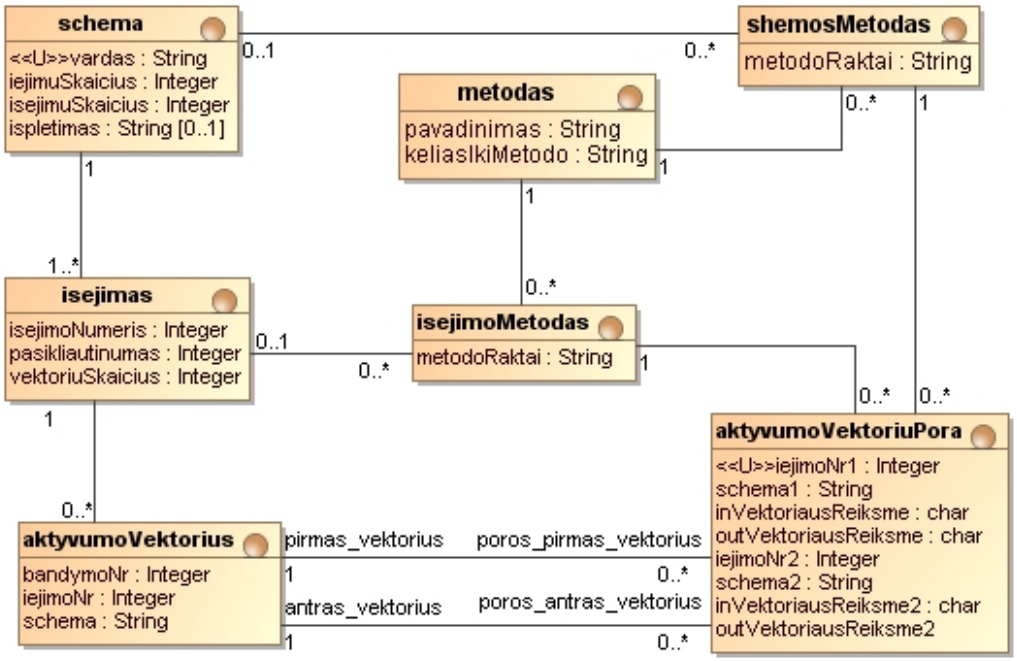
Tikimo kriterijus:	Sistemos išskirstyti darbai generuoja adekvačius rezultatus neskirstytiems darbams. Išskirstytų darbų rezultatų failai apjungus turi atitikti neskirstytų darbų rezultatų failus.		
Užsakovo tenkinimas:	5	Užsakovo netenkinimas:	5
Priklausomybės:	Nėra.	Konfliktai:	Nėra.
Papildoma medžiaga:	Nėra.		
Istorija:			

3.2. Dalykinės srities modelis

Dalykinės srities klasių diagrama pateikta žemiau esančiose diagramose (35 pav. Eksperimentavimo sistemos klasių diagrama ir 36 pav. Testavimo srities klasių diagrama). Diagramoje pavaizduota sistemos esybės bei jų parametrai, esybių ryšiai su kardinalumais tarp jų.



35 pav. Eksperimentavimo sistemos klasių diagrama



36 pav. Testavimo srities klasių diagrama

3.3. Reikalavimų analizės apibendrinimas

Sugeneravus svarbiausių funkcinių reikalavimų ataskaita (atributų matrica), kurioje išrikiuoti reikalavimai pagal užsakovo patenkinimą, nustatyti prioritetai, į kuriuos labiausiai reikia atkreipti dėmesį kuriant sistemą, nes būtent tie reikalavimai yra didžiausios svarbos galutiniam sistemos vartotojui. Įvertinti ir suvokti bendrą sistemos funkcionalumą padeda suformuota sistemos panaudos atvejų analizė.

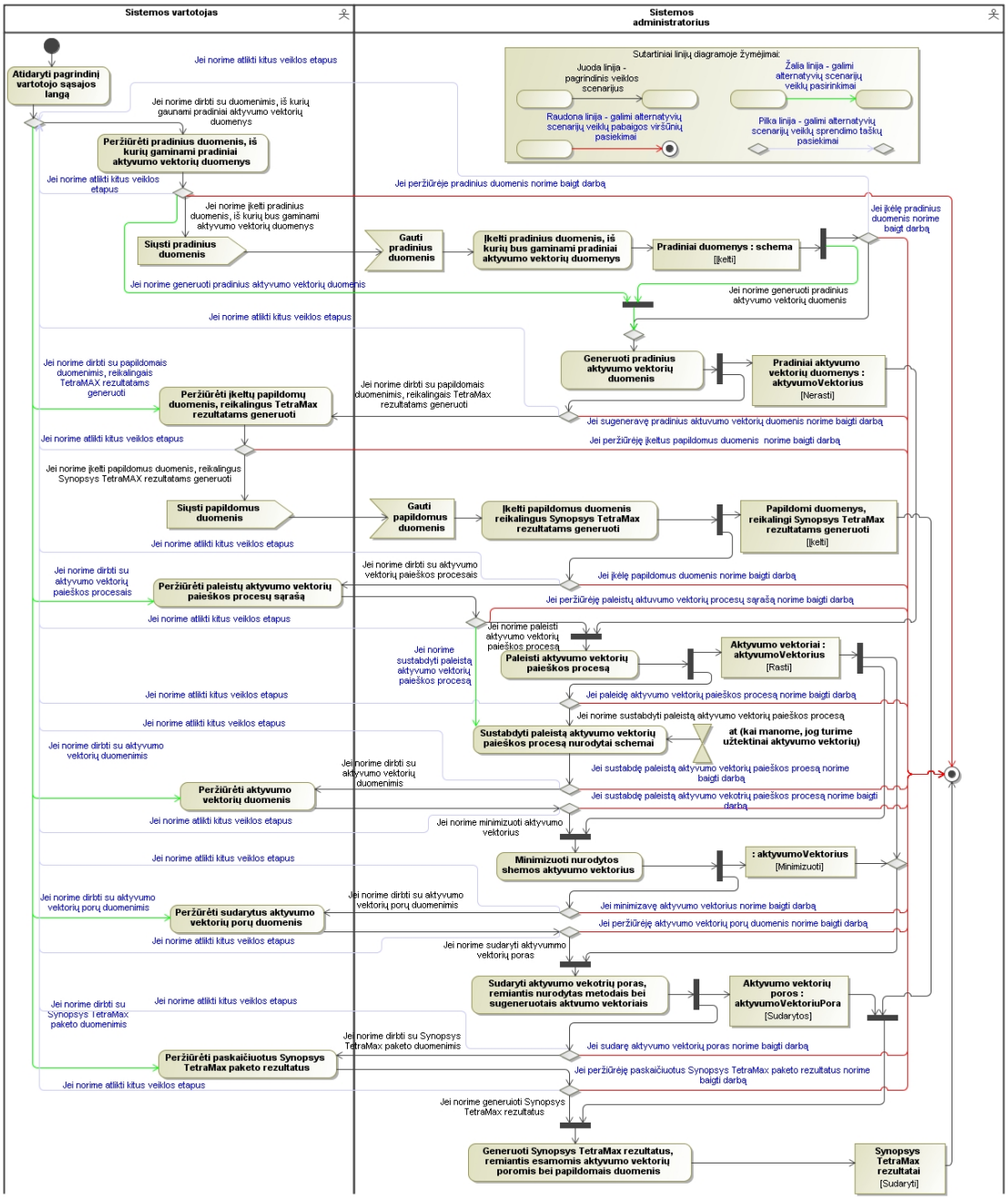
Detaliai numatyti sistemos reikalavimai padeda suvokti bendrą sistemos vaizdą, dėl to bus lengviau sistemą kurti.

4. Sprendimo aprašas

4.1. Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo procesai

Funkcinio vėlinimo testo eksperimentas gali būti atliekamas nenuosekliai, jeigu atliekamam eksperimentui reikalingi duomenys jau buvo paruošti ankstesnių eksperimentų. Tačiau tarkime, jog eksperimentas atliekamas pirmą kartą ir jokių pradinių duomenų failų neturime. Nusiuntus pradinius duomenų failus, nuo kurių viskas ir prasideda, galima pradėti aktyvumo vektorių paiešką. Kol vyksta paieška,

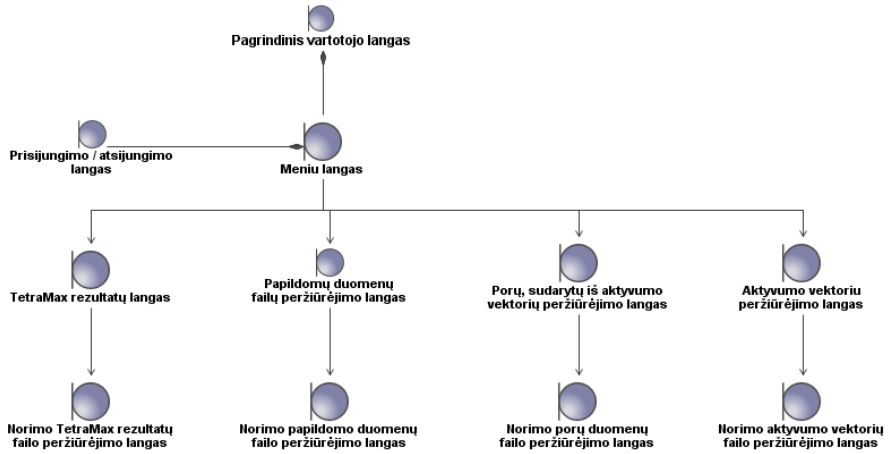
galima sukelti papildomus duomenų failus, reikalingus simuliacijai, o po to – sustabdyti paieškos procesą. Iškvietus kelias sistemos funkcijas, sugeneruojami testinių porų failai bei atliekama klaidų simuliacija. Gauti rezultatai automatiškai išanalizuojami ir patalpinami į palyginimų lentelę, kurią galima peržiūrėti. Visa tai atvaizduoja žemiau pateikta veiklos diagrama (37 pav. Veiklos diagrama).



37 pav. Veiklos diagrama

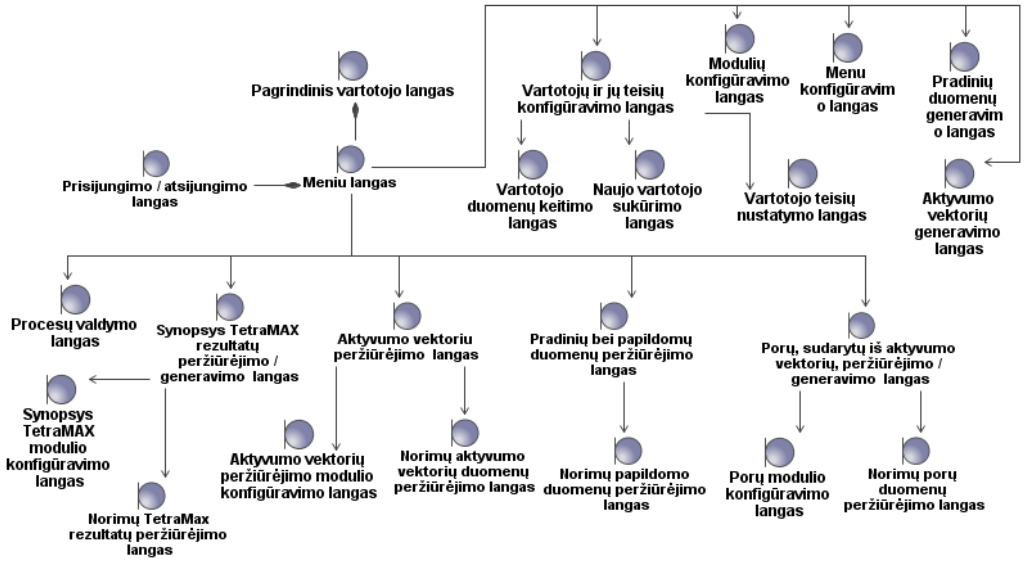
4.2. Vartotojo paslaugos

Vartotojo navigacijos plane (38 pav. Vartotojo navigacijos planas) pavaizduoti sistemos langai ir kaip vartotojas gali tarp jų keliauti, kad pasiektų reikiamą informaciją.



38 pav. Vartotojo navigacijos planas

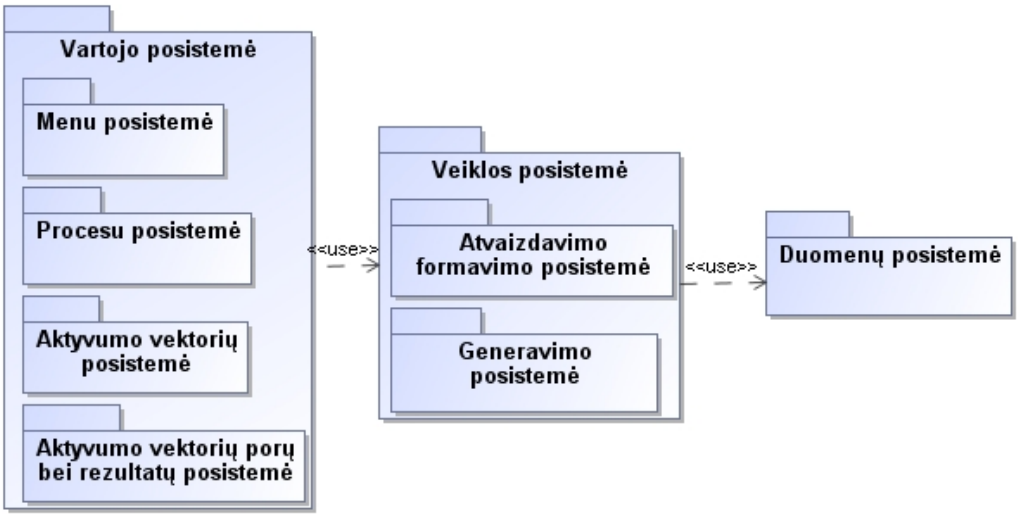
Sistemos kūrėjo navigacijos planas (39 pav. Sistemos kūrėjo navigacijos planas) papildytas keliais langais, kartu ir su jų siūlomu funkcionalumu: vartotojų sukūrimo, šalinimo, jų duomenų keitimo langai, modulių, meniu konfigūravimo langai, pradinių duomenų ir aktyvumo vektorių generavimo langai.



39 pav. Sistemos kūrėjo navigacijos planas

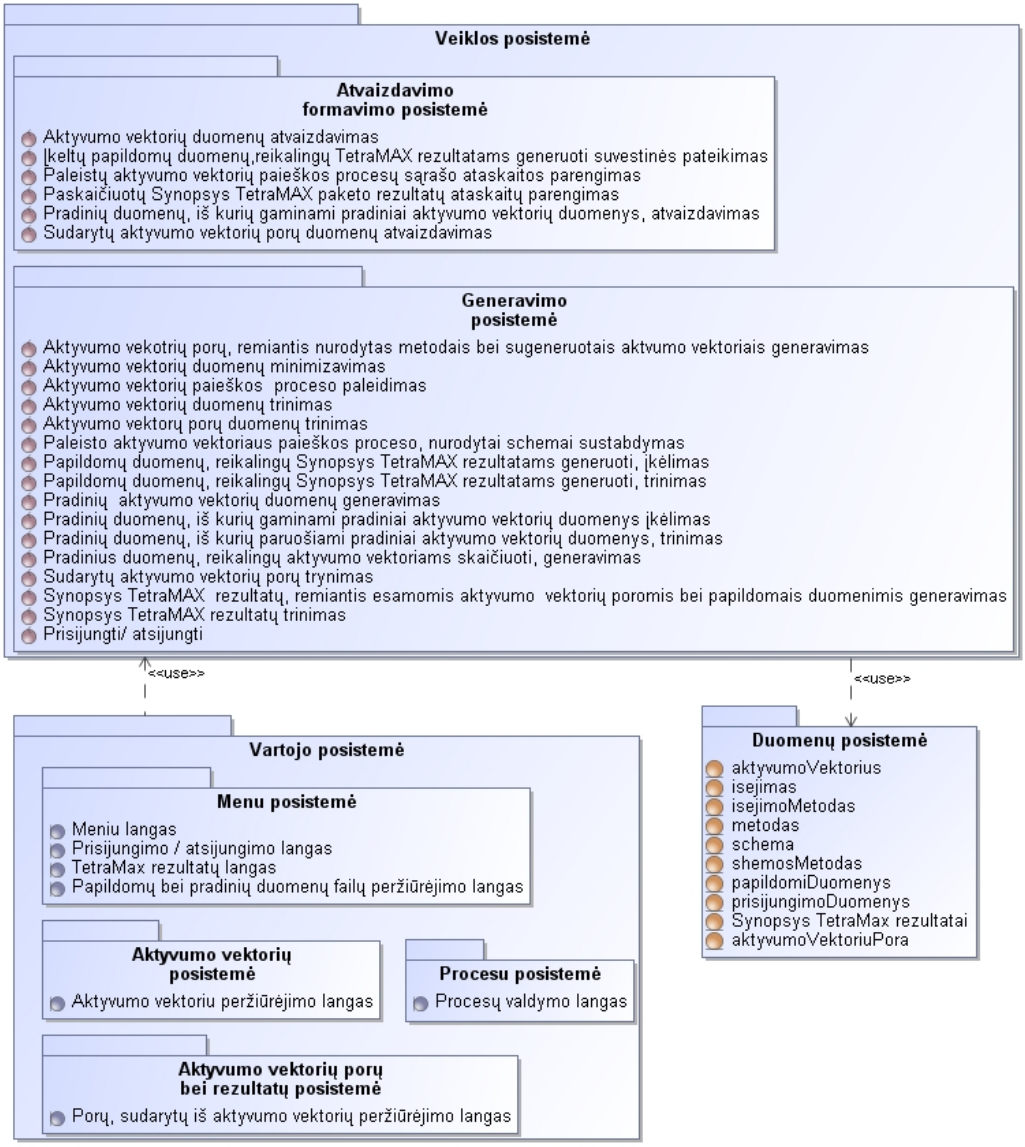
4.3. Loginė architektūra

Sistema kuriama trijų lygių architektūros pagrindu, atskiriant vartotojo, veiklos bei duomenų posistemes. Vartotojo posistemėje realizuojama vartotojo grafinė sąsaja, veiklos posistemėje – pagrindinės sistemos funkcijos, o duomenų posistemė atspindi naudojamus duomenis. Sąsają tarp šių posistemių atspindi žemiau pateikta diagrama (40 pav. Trijų lygių architektūra).



40 pav. Trijų lygių architektūra

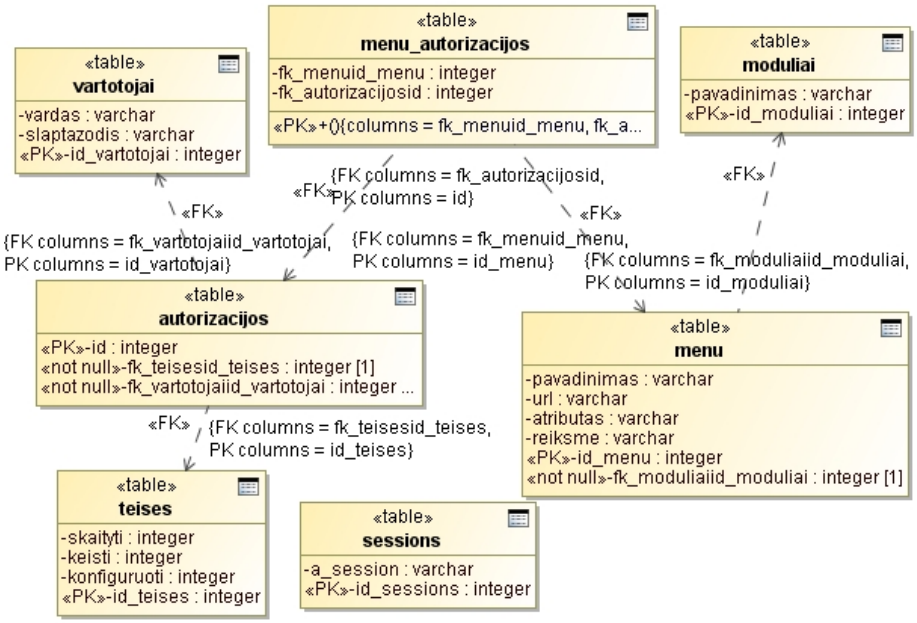
Detali šios architektūros sudėtis pateikta žemiau esančioje diagramoje (41 pav. Detali sistemos architektūra).



41 pav. Detali sistemos architektūra

4.4. Duomenų bazių schemos

Yra dvi duomenų bazių schemos: viena skirta sistemos veikimui ir jos naudojamiems duomenims talpinti (42 pav. Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti duomenų bazės schema), kita – testavimo procesų duomenims bei rezultatams (43 pav. Testavimo proceso etapų duomenų bazės schema).



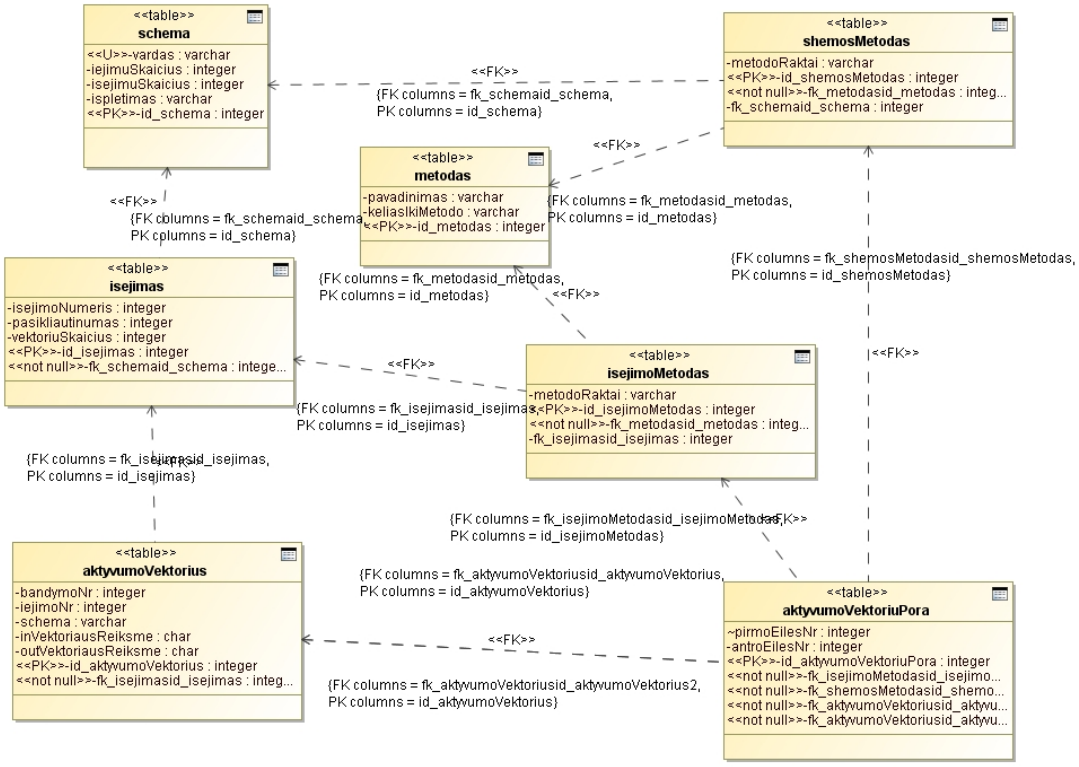
42 pav. Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti duomenų bazės schema

Žemiau esančioje lentelėje (20 lentelė. Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti duomenų bazės schemos lentelių aprašymas), pateikti 42 pav. (Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti duomenų bazės schema) pavaizduotos duomenų bazės schemos lentelių aprašymai.

20 lentelė. Eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti duomenų bazės schemos lentelių aprašymas

Lentelės pavadinimas	Aprašymas
vartotojai	Lentelė skirta vartotojų prisijungimo duomenims laikyti.
teises	Lentelė skirta laikyti galimų teisių aprašymams.
sessions	Šioje lentelėje laikoma prisijungimo sesijų identifikatoriai.
autorizacijos	Šioje lentelėje laikomos vartotojui suteiktos meniu teisės.
menu	Meniu pavadinimai, sąsajos su moduliais ir nustatymais.
moduliai	Modulių pavadinimai.
menu_autorizacijos	Tarpinė lentelė, sujungianti menu ir

	autorizacijų lenteles.



43 pav. Testavimo proceso etapų duomenų bazės schema

Žemiau esančioje lentelėje (21 lentelė. Testavimo etapų duomenų bazės schemos lentelių) pateikta šios schemos lentelių apibūdinimai.

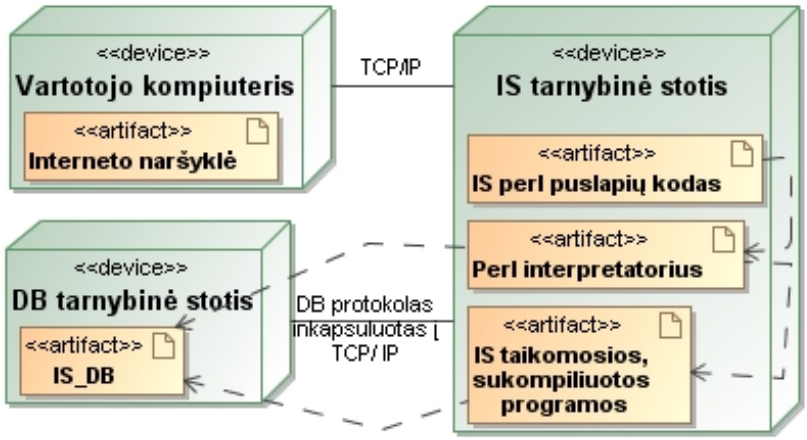
21 lentelė. Testavimo etapų duomenų bazės schemos lentelių aprašymai

Lentelės pavadinimas	Aprašymas
schema	Šioje lentelėje talpinama informacija apie programinį schemos modelį
metodas	Šioje lentelėje talpinama informacija apie metodą, kuris naudojamas sudaryti aktyvumo vektorių poroms.
schemosMetodas	Lentelė naudojama aprašyti metodų iškvietimo komandas, panaudojant raktus, jei tokių reikia.
isejimoMetodas	Išėjimui taikomo metodo raktai, kuriuos panaudojus iškviečiamas vieno schemos išėjimo testavimo poroms sudaryti metodas.
isejimas	Schemos išėjimams saugoti skirta lentelė.

aktyvumoVektorius	Aktyvumo vektoriams saugoti skirta lentelė.
aktyvumoVektoriuPora	Sudarytoms aktyvumo vektorių poroms saugoti skirta lentelė.

4.5. Realizacijos modelis (programinių komponentų architektūra, diegimo modelis)

44 pav. "Diegimo modelis" pateikiama diegimo diagrama, vaizduojanti realizuojamos sistemos išsiskirstymą techninėje įrangoje.



44 pav. Diegimo modelis

5. Sprendimo realizacija

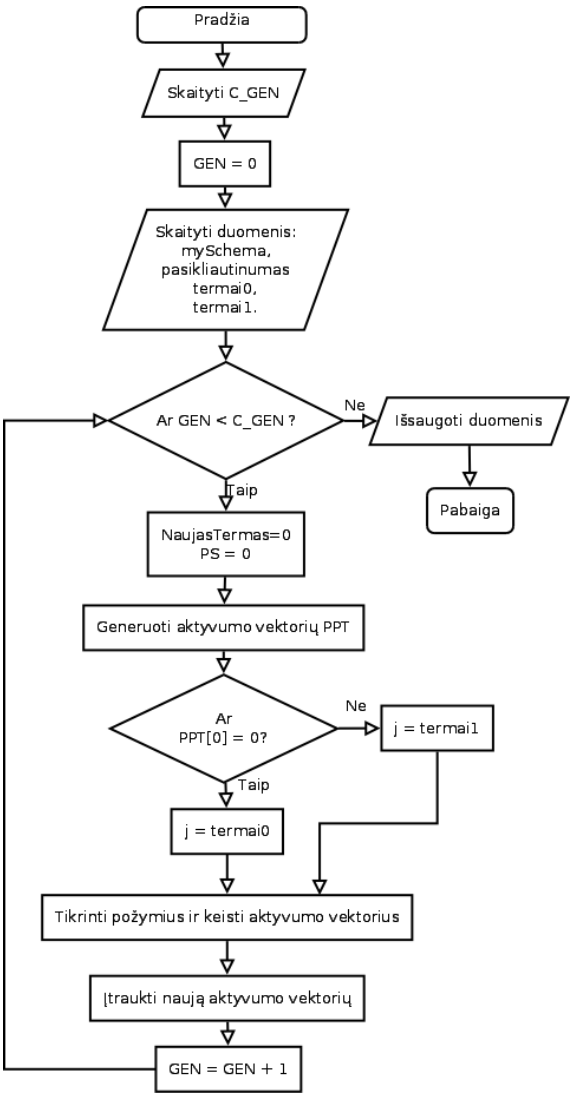
Realizacija – tai sukurta eksperimentavimo sistema funkcinio vėlinimo testų generavimo metodui kurti. Pagrindiniai algoritmai aktyvumo vektorių paieškai parašyti C++ kalba ir apjungti į automatizuotą sistemą PERL programavimo kalba, tokiu būdų suteikiant užsakovui galimybę valdyti eksperimentų kūrimo proceso eigą. Sistema realizuota taip, jog komponentai gali būti kuriami įvairioms užduotims spręsti ir tik kviečiant tam tikrus metodus atliekami eksperimentavimo sistemai reikalingi veiksmai, pavyzdžiui, automatinis kelio iki duomenų ar rezultatų failo suskaičiavimas.

5.1. Pagrindiniai eksperimentavimo sistemos algoritmai

5.1.1. Aktyvumo vektorių paieškos algoritmas

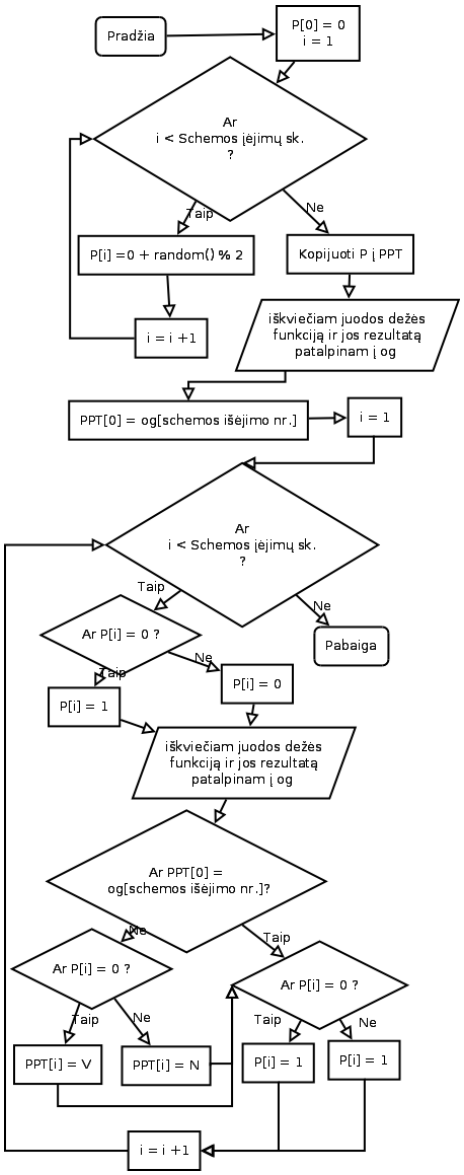
Aktyvumo vektorių paieškos algoritmas pavaizduotas 45 pav. „Aktyvumo vektorių paieškos algoritmo diagrama". Algoritmo pradžioje perskaitoma C_GEN

sveikas skaičius, nusakantis kiek kartų bus ieškoma vektorių. Iš duomenų failo perskaitomi schemos duomenys, nurodyto išėjimo pasikliautinumas, išėjimui jau iš ankščiau sugeneruoti aktyvumo vektoriai, kurie pagal schemos juodos dėžės išėjimui gražintą funkciją yra suskirstyti į dvi aibes: termai0 ir termai1, kur skaičiai pavadinime nurodo gražintą reikšmę.



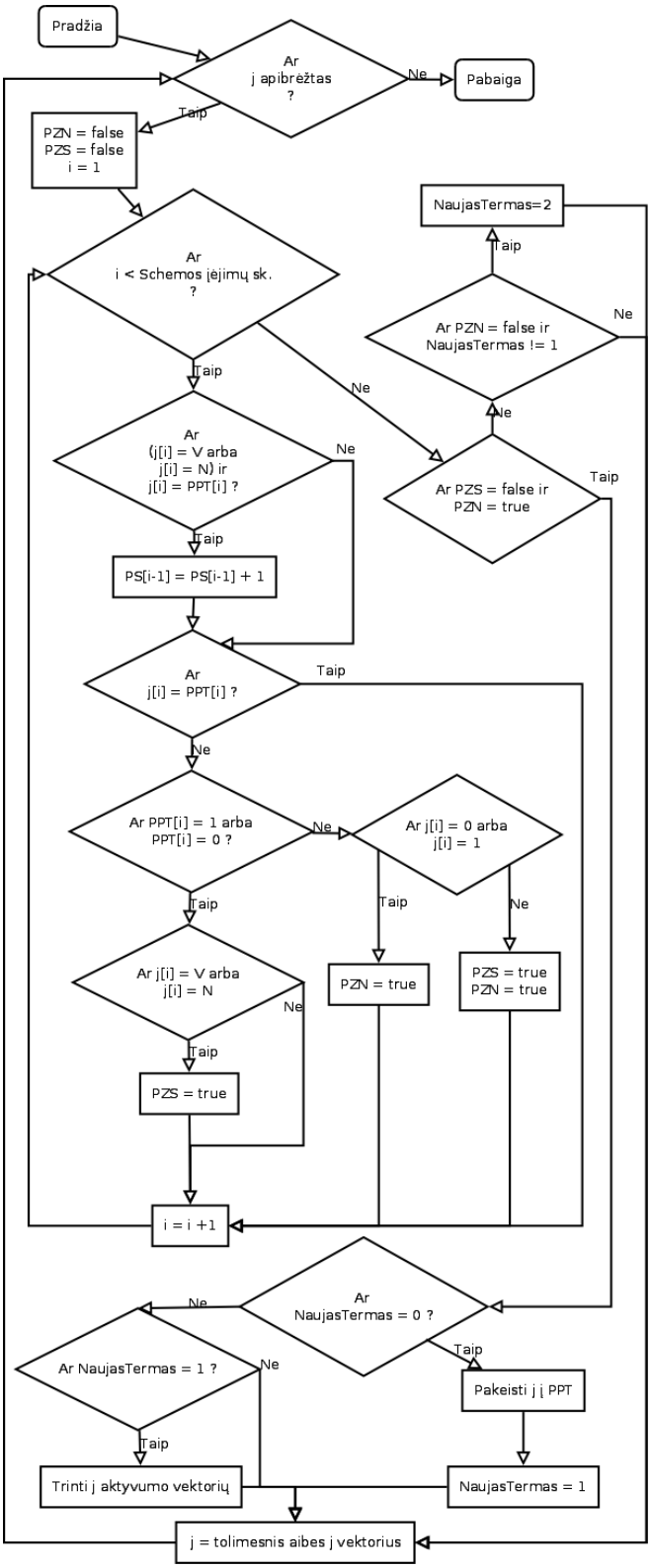
45 pav. Aktyvumo vektorių paieškos algoritmo diagrama

„Generuoti aktyvumo vektorių“ PPT algoritmas pateiktas 46 pav. „Aktyvumo vektoriaus generavimo algoritmo diagrama“. Joje matyti, jog aktyvumo vektoriaus reikšmės generuojamos atsitiktinai, iškvietus funkciją „random()“, o schemos išėjimo signalai matuojami kviečiant juodos dėžės modelio funkciją. Pastaroji rezultatus gražina kintamajam „og“.



46 pav. Aktyvumo vektoriaus generavimo algoritmo diagrama

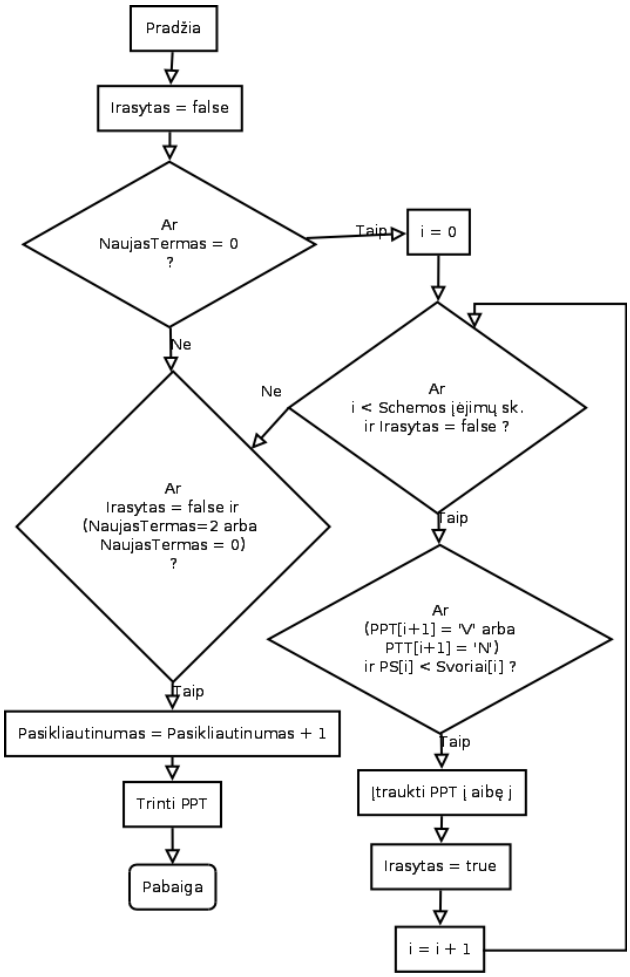
Palyginus sugeneruotą aktyvumo vektorių remiantis tam tikrais požymiais su esamais, nustatoma ar naujasis vektorius dengia seną. Jei dengia – senasis vektorius pakeičiamas naujuoju. Požymių tikrinimo ir aktyvumo vektorių keitimo algoritmas pavaizduotas žemiau esančioje diagramoje (47 pav. Požymių tikrinimo ir aktyvumo vektorių keitimo algoritmo diagrama).



47 pav. Požymių tikrinimo ir aktyvumo vektorių keitimo algoritmo diagrama

Surastas aktyvumo vektorius įtraukiamas, kaip naujas aibės elementas, praėjus tam tikrus patikrinimo etapus įtraukimo algoritme (48 pav. „Aktyvumo vektoriaus

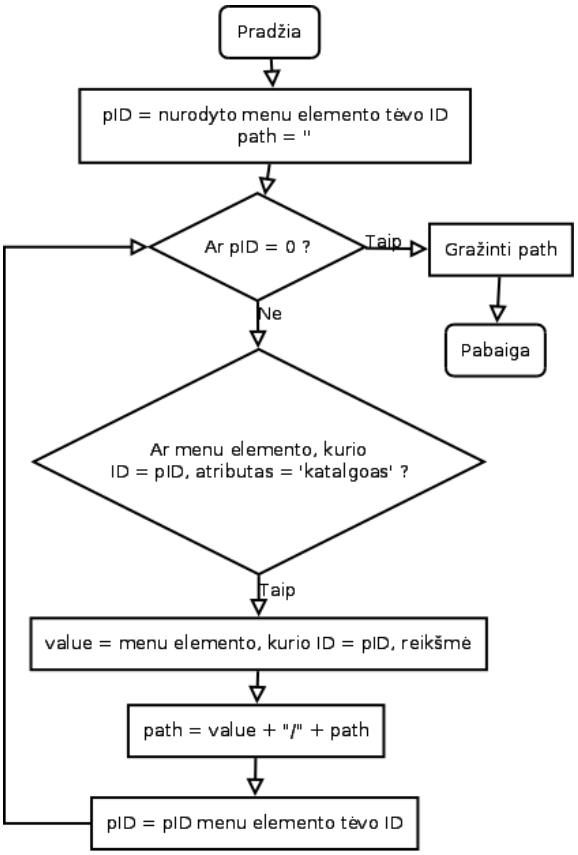
įtraukimo į aibę algoritmo diagrama“).



48 pav. Aktyvumo vektoriaus įtraukimo į aibę algoritmo diagrama

5.1.2. Automatizuotas kelio iki katalogo sudarymo algoritmas

Kelio iki katalogo formavimas prasideda gavus tėvinio menu elemento ID. Algoritme tėvinis menu elementas išanalizuojamas, ir jei jo atributo pavadinimas sutampa su „katalogas“, tuomet reikšmė pridedama prie sudaromo kelio, atskiriant pasviruoju brūkšniu. Vėliau analizuojamas aukštesnis tėvinis menu elementas, ir taip, kol nelieka analizuojamų elementų. Sudarytas kelias gražinamas kartu su kintamojo reikšme. Algoritmo diagrama pateikiama žemiau esančiame paveikslėlyje (49 pav. Automatizuoto kelio iki katalogo sudarymo algoritmo diagrama).



49 pav. Automatizuoto kelio iki katalogo sudarymo algoritmo diagrama

5.2. Sistemos grafinis vaizdas

Sistema ir su jos rezultatai pasiekiami internete bet kurios modernios naršyklės pagalba, adresu: http://kopustas.elen.ktu.lt/~mantas/ . Šiuo adresu apsilankę ir neprisijungę vartotojai pirmiausiai atpažįstami, kaip sistemos svečiai arba anoniminiai vartotojai, su ribotomis teisėmis ir galimybėmis.

5.2.1. Anoniminio vartotojo aplinka

Pirminis vaizdo langas, kurį anoniminis vartotojas mato, pateiktas 50 pav. „Pirminis lango vaizdas“. Šiame lange vartotojai gali pasirinkti norimą peržiūrėti skyriaus siūlomas galimybes, rezultatus arba autentifikuoti save, jei turi sistemoje paskirą.

50 pav. Pirminis lango vaizdas

Pasirinkus vieną iš aktyvumo vektorių skyrių atvaizduojamas su tuo skyriumi susietų schemų aktyvumo vektorių lentelė (51 pav. Skyriaus susietų schemų aktyvumo vektorių langas).

Aktyvumo vektoriai	V/N -> 1/0		o/z	
	Ne Minimizuoti		Minimizuoti	
	Su antraštėm	Be antraščių	Su antraštėm	Be antraščių
c7552	c7552	c7552	c7552	c7552
c6288	c6288	c6288	c6288	c6288
c5315	c5315	c5315	c5315	c5315
c3540	c3540	c3540	c3540	c3540
c2670	c2670	c2670	c2670	c2670
c1908	c1908	c1908	c1908	c1908
c1908x	c1908x	c1908x	c1908x	c1908x
c1355	c1355	c1355	c1355	c1355
c880	c880	c880	c880	c880
c499	c499	c499	c499	c499
c432	c432	c432	c432	c432
c17	c17	c17	c17	c17
s208_2	s208_2	s208_2	s208_2	s208_2
s27_2	s27_2	s27_2	s27_2	s27_2

51 pav. Skyriaus susietų schemų aktyvumo vektorių langas

Pasirinkus norimą atvaizdavimui būdą, pavyzdžiui tiesiog atvaizduoti C17 schemos aktyvumo vektorius, atvaizduojama schemos informacija, aktyvumo vektoriai ir išskiriami schemos įėjimų aktyvūs simboliai (52 pav. C17 schemos aktyvumo vektoriai).

```
Schemos vardas: c17[8 baitai]
Ispletimas: [4 baitai]
Iejimu kiekis: 5[4 baitai]
Isejimu kiekis: 2[4 baitai]
Pasikliautinumai:
0 - 102000100
1 - 102000100

OUT=0    ICount=7
-    Svoriai
2    5 3 2 3 0
Fja    Termas
0    N1VV0
0    1NN10
0    NN100
1    V0V10
1    0V1N1
1    0VN10

OUT=1    ICount=7
-    Svoriai
2    0 2 2 2 4
Fja    Termas
0    1N01N
0    11VV1
1    101NV
1    10N1V
1    1V1N0
1    0VN10
```

52 pav. C17 schemos aktyvumo vektoriai

Sugeneruoti porų duomenų failai atitinkamuose skyriuose pateikiami lentelės pavidalu, kur stulpeliai atitinka katalogų pavadinimus, kuriuose surūšiuoti duomenys. Pastarieji pavadinimai susieti su panaudotais metodais poroms generuoti (53 pav. Sugeneruoti porų duomenų failai).

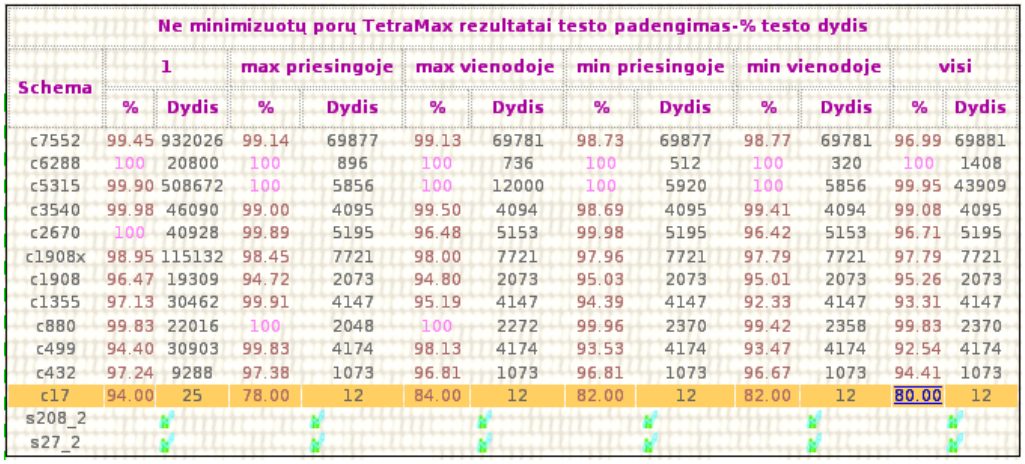
53 pav. Sugeneruoti porų duomenų failai

Paspaudus ant pageidaujamos schemos, pavyzdžiui C17 porų rezultato failo, kuris yra „visi“ kataloge (šis katalogas susietas su MIT metodu, pagal kurį visi aktyvumo vektoriai yra antra poros signalų seka, turinti pakeistas aktyvias reikšmes

priešingomis pirmajai signalų sekai), gausime atitinkamu metodu sudarytą testinę seką ir schemos informaciją:

Schemos vardas: c17[8 baitai]
Ispletimas: [4 baitai]
Iejimu kiekis: 5[4 baitai]
Isejimu kiekis: 2[4 baitai]
Pasikliautinumai:
0 - 102000100
1 - 102000100
11000
01110
11110
10010
11100
00100
00010
10110
00111
01101
00110
01010
11011
10010
11001
11111
10110
10101
10110
10011
10110
11100
00110
01010

Bet kuriame TetraMAX rezultatų skyriuje atvaizduojami rezultatai, gauti simuliuojant klaidas Synopsys TetraMAX programiniu paketu. Rezultatai pateikiami automatiškai suformuotos lentelės pavidalu. Lentelės eilutėse pateikiami schemų pavadinimai, o stulpeliuose pavadinimas katalogo susieto su naudojamo testinėms sekoms generuoti metodo pavadinimu, perdavimo signalų padengimas, išreikštas procentais ir testo dydis (54 pav. TetraMAX rezultatų lentelė).



Ne minimizuotų porų TetraMax rezultatai testo padengimas-% testo dydis

Schema	1		max priesingoje		max vienodoje		min priesingoje		min vienodoje		visi	
	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis
c7552	99.45	932026	99.14	69877	99.13	69781	98.73	69877	98.77	69781	96.99	69881
c6288	100	20800	100	896	100	736	100	512	100	320	100	1408
c5315	99.90	508672	100	5856	100	12000	100	5920	100	5856	99.95	43909
c3540	99.98	46090	99.00	4095	99.50	4094	98.69	4095	99.41	4094	99.08	4095
c2670	100	40928	99.89	5195	96.48	5153	99.98	5195	96.42	5153	96.71	5195
c1908x	98.95	115132	98.45	7721	98.00	7721	97.96	7721	97.79	7721	97.79	7721
c1908	96.47	19309	94.72	2073	94.80	2073	95.03	2073	95.01	2073	95.26	2073
c1355	97.13	30462	99.91	4147	95.19	4147	94.39	4147	92.33	4147	93.31	4147
c880	99.83	22016	100	2048	100	2272	99.96	2370	99.42	2358	99.83	2370
c499	94.40	30903	99.83	4174	98.13	4174	93.53	4174	93.47	4174	92.54	4174
c432	97.24	9288	97.38	1073	96.81	1073	96.81	1073	96.67	1073	94.41	1073
c17	94.00	25	78.00	12	84.00	12	82.00	12	82.00	12	80.00	12
s208_2												
s27_2												

54 pav. TetraMAX rezultatų lentelė

Šioje lentelėje galima pasirinkti norimą atvaizduoti TetraMAX programinio paketo sugeneruotą klaidų simuliacijos rezultatų failą. Tarkime, jog norime atvaizduoti C17 schemos sugeneruotą „visi“ kataloge esantį klaidų simuliacijos rezultatų failą, tuomet paspaudus ant atitinkamos schemos ir stulpelio sankirtoje esančio klaidų padengimo rezultato, gausime išsamią simuliacijos ataskaitą:

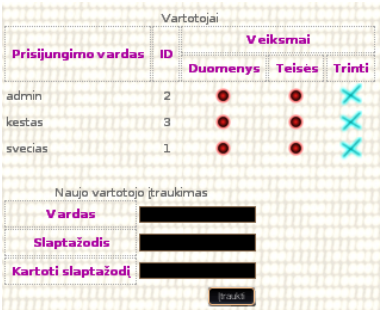
```
#patterns      #faults      test       process
 simulated   detect/active  coverage   CPU time
 ---------   -------------  --------   --------
 12              40     10     80.00%       0.00
 Fault simulation completed: #patterns=12, CPU time=0.00

 fault class                     code    #faults
 ------------------------------  ----  ---------
 Detected                         DT          40
 Possibly detected                PT           0
 Undetectable                     UD           0
 ATPG untestable                  AU           0
 Not detected                     ND          10
 -----------------------------------------------
 total faults                                 50
 test coverage                            80.00%
 -----------------------------------------------
```

5.2.2. Administratoriaus paskyros aplinka

Įvedus teisingus administratoriaus paskyros prisijungimo duomenis, pateksime į praplėstą anoniminio vartotojo paskyrą. Prasiplečia ne tik menu su naujais elementais (55 pav. Praplėstas sąrašas meniu elementų administratoriaus paskyrai), bet ir kiekvieno skyriaus galimybės.

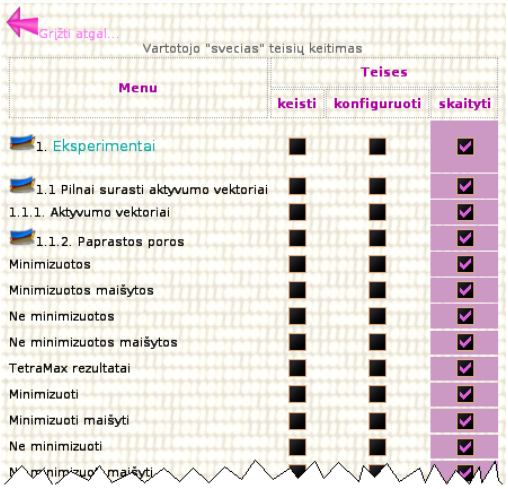
55 pav. Praplėstas sąrašas meniu elementų administratoriaus paskyrai

Vartotojų informacija arba teisės kiekvieno skyriaus atžvilgiu, gali būti keičiamos pasirinkus „Vartotojai ir teisės“ meniu elementą (56 pav. Vartotojai ir teisės).



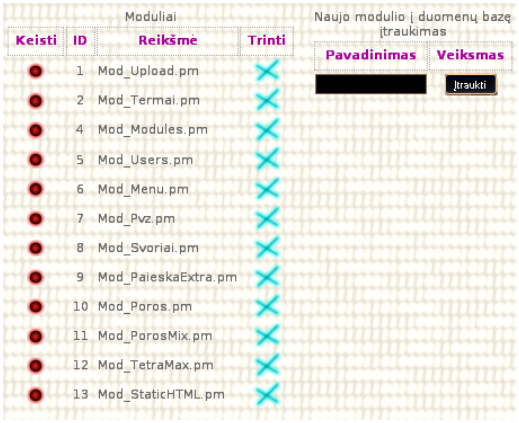
56 pav. Vartotojai ir teisės

Pasirinkus keisti norimo vartotojo, pavyzdžiui „svečias“ teises, patenkama į teisių keitimo langą, kuriame žymint atitinkamuose stulpeliuose esančius elementus, galima nustatyti norimas vartotojo teises kiekviename skyriuje. Šios funkcijos valdymo vizualinis fragmentas atvaizduotas žemiau esančiame paveikslėlyje (57 pav. Anoniminio vartotojo teisių nustatymo lango fragmentas).



57 pav. Anoniminio vartotojo teisių nustatymo lango fragmentas

Kiekvienas meniu elementas gali būti susietas su kuriame nors modulyje aprašytu funkcionalumu. Modulių sąrašas tvarkomas pasirinkus „Moduliai“ meniu elementą (58 pav. Modulių tvarkymo langas).



58 pav. Modulių tvarkymo langas

Menu elementai kuriami, susiejami su moduliais ir kitaip tvarkomi, pasirinkus meniu elementą „Menu“. Šių elementų koregavimo lango fragmentas pavaizduotas žemiau esančiame paveikslėlyje (59 pav. Meniu elementų koregavimo lango fragmentas).

Menu

Keisti	Pavadinimas	URL	Modulis	ID	Tevo ID	Atributas	Reikšmė	Trinti
	1. Eksperimentai			12	0	katalogas	Eksperimentai	
	1.1 Pilnai surasti aktyvumo vektoriai			20	12	katalogas	Pilni vektoriai	
	1.1.1. Aktyvumo vektoriai	Termai	Mod_Termai.pm	11	20	failas	vektoriai.cfg	
	1.1.2. Paprastos poros			29	20	katalogas	Paprastos poros	
	Minimizuotos	Min Pilni Svoriai	Mod_Poros.pm	31	29	failas	Min.cfg	
	Minimizuotos maišytos	Min Mix Pilni Svoriai	Mod_PorosMix.pm	37	29	failas	MinMix.cfg	
	Ne minimizuotos	NeMin Pilni Svoriai	Mod_Poros.pm	30	29	failas	NeMin.cfg	
	Ne minimizuotos maišytos	Ne Min Mix Pilni Svoriai	Mod_PorosMix.pm	38	29	failas	NeMinMix.cfg	
	TetraMax rezultatai			43	29	subMenu		
	Minimizuoti	TetraMax Min Pilni	Mod_TetraMax.pm	44	43	failas	tetraMaxMin.cfg	
	Minimizuoti maišyti	TetraMax Min Mix Pilni	Mod_TetraMax.pm	50	43	failas	tetraMaxMinMix.cfg	
	Ne minimizuoti	TetraMax Ne Min Pilni	Mod_TetraMax.pm	45	43	failas	tetraMaxNeMin.cfg	

59 pav. Meniu elementų koregavimo lango fragmentas

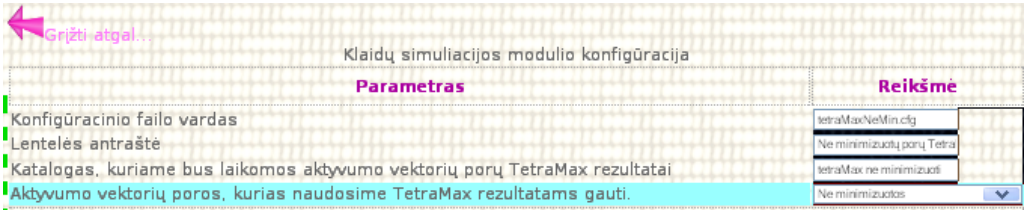
Sistemos moduliai, skirti generuoti aktyvumo vektorių duomenų failus, testines sekas ar galutinius klaidų padengimo rezultatų failus yra konfigūruojami. Konfigūracijos galimybė atsiranda, kuomet turintis vartotojas konfigūracijos teises, pasirenka kurį nors skyrių. Vartotojo galimybės gali prasiplėsti priklausomai nuo modulio programinio kodo, t.y. be galimybės konfigūruoti modulį, gali atsirasti ir naujų funkcijų, pavyzdžiui, trinti sudarytus klaidų padengimo rezultatų failus (60 pav. TetraMAX rezultatų failų atvaizdavo lango administratoriaus).

Konfiguruoti...

Ne minimizuotų porų TetraMax rezultatai testo padengimas-% testo dydis													Veiksmai	
Schema	1		max priesingoje		max vienodoje		min priesingoje		min vienodoje		visi			
	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis	Sudaryti rezultatus	Trinti
c7552	99.45	932026	99.14	69877	99.13	69781	98.73	69877	98.77	69781	96.99	69881		
c6288	100	20800	100	896	100	736	100	512	100	320	100	1408		
c5315	99.90	508672	100	5856	100	12000	100	5920	100	5856	99.95	43909		
c3540	99.98	46090	99.00	4095	99.50	4094	98.69	4095	99.41	4094	99.08	4095		
c2670	100	40928	99.89	5195	96.48	5153	99.98	5195	96.42	5153	96.71	5195		
c1908x	98.95	115132	98.45	7721	98.00	7721	97.96	7721	97.79	7721	97.79	7721		
c1908	96.47	19309	94.72	2073	94.80	2073	95.03	2073	95.01	2073	95.26	2073		
c1355	97.13	30462	99.91	4147	95.19	4147	94.39	4147	92.33	4147	93.31	4147		
c880	99.83	22016	100	2048	100	2272	99.96	2370	99.42	2358	99.83	2370		
c499	94.40	30903	99.83	4174	98.13	4174	93.53	4174	93.47	4174	92.54	4174		
c432	97.24	9288	97.38	1073	96.81	1073	96.81	1073	96.67	1073	94.41	1073		
c17	94.00	25	78.00	12	84.00	12	82.00	12	82.00	12	80.00	12		
s208_2													Skaičiuoti rezultatus	
s27_2													Skaičiuoti rezultatus	

60 pav. TetraMAX rezultatų failų atvaizdavo lango administratoriaus funkcijos

Pasirinkus nuorodą „konfigūruoti“, galima keisti įvairius modulio parametrus. Konfigūracijos lango pavyzdys pateiktas žemiau esančiame paveikslėlyje (61 pav. Klaidų padengimo rezultatų generavimo modulio konfigūracijos langas).



61 pav. Klaidų padengimo rezultatų generavimo modulio konfigūracijos langas

6. Eksperimentavimo sistemos funkcinio vėlinimo testui generuoti testavimas

Kuriant eksperimentavimo sistemą funkcinio vėlinimo testo generavimo metodui kurti, norima, kad sistema turėtų kiek galima mažiau klaidų. Siekiant užtikrinti sukurtos sistemos kokybę, buvo atliktas eksperimentavimo sistemos funkcinio vėlinimo testui generuoti testavimas, kuris apžvelgiamas šiame skyriuje. Testavimo metu buvo atliekamas funkcinio vėlinimo testų ir jų rezultatų generavimas.

6.1. Testavimo tikslai ir objektai

Vienas iš testavimo tikslų yra patikrinti ar sukurta sistema veikia ir gražina rezultatą, kokio mes tikimės. Patikrinus, ar eksperimentavimo sistemos procesų rezultatai sutampa su numatomais, bus galima generuojamus rezultatus priimti, kaip patikimus. Kitas testavimo tikslas – nustatyti ar atliekant įprastos veiklos proceso funkcijas, sistema tvarkingai atlieka jai paskirtą darbą ir neišveda klaidų pranešimų, kurie signalizuotų apie galimas programinio kodo klaidas. Testuojami objektai: sistemos branduolys, atsakingas už modulių užkrovimą, sistemos skirtingos paskirties moduliai, grafinė sistemos sąsaja ir funkcinio vėlinimo testų generavimo metodai, kurių testavimas prilygsta eksperimento atlikimą.

6.2. Testavimo metodai ir resursai

Atliekant visos sistemos testavimą, matuosime pagrindinių funkcinio vėlinimo testų generavimo etapų funkcijų pradžios ir pabaigos laikus, kurie leis nustatyti apytiksliai sutaupomą laiką tarp iškviečiamų funkcijų. Šiam tikslui, į pageidaujamų testuoti funkcijų pradžią ir galą įterpsime nuorodą į kodo fragmentą (Priedas nr. 4: Kodo fragmentas, skirtas laikui fiksuoti testavime), kuris testavimo rezultatų failą „laikai.txt“ automatiškai papildys sistemos laikais.

O atliekant sistemos testavimą pagal pateiktus testavimo atvejus, užtikrinsime

kiek galima mažesnį eksperimentavimo sistemos funkcinio vėlinimo testo generavimo metodui kurti klaidų kiekį.

Viso testavimo metu bus kviečiamos funkcijos, kurių galutiniame rezultate iš sugeneruotų aktyvumo vektorių sukuriamos testavimo sekos. Klaidų simuliacijos metu, bus nustatomas signalo perdavimo klaidų padengimas, kuris leis įvertinti metodų gerumą.

C++ programavimo kalba parašytos programos testuojamos su „Valgrind“ programiniu paketu, siekiant patikrinti korektiška darbą su kompiuterio resursais ir surasti galimas kodo klaidas.

Perl programavimo kalba parašytos kodo dalys testuojamos pasitelkus nemokamą modulį „CGI::Carp“ ir nurodžius, kad pastarasis naudotų funkcijas „fatalsToBrowser“ ir „warningsToBrowser“, leidžiančias akimirksniu iškilusias klaidas pamatyti naršyklės ekrane. Taip pat nurodome „–w –T“ raktus Perl interpretatoriui, kurie įjungia kodo diagnozavimą, padedantį aptikti klaidas ir įspėjanti apie galimas saugumo spragas.

Testavimui atlikti užtenka vieno kompiuterio, su kuriuo buvo sukurta sistema. Atlikus testavimą, turėsime tuo pačiu ir atliktą eksperimentą, funkcinio vėlinimo testų generavimo metodams palyginti.

6.3. Testavimo apribojimai

Eksperimentavimo sistema funkcinio vėlinimo testo generavimo metodui kurti buvo kuriama smukliomis iteracijomis, kurių metu buvo realizuojama dalis sistemos. Gale kiekvienos iteracijos atliekami minimalūs funkcionalumo ir kodo teisingumo testavimai, dėl to nėra nuoseklaus testavimo plano, o didžioji dalis klaidų buvo pataisyta sistemos kūrimo metu. Kadangi visada išlieka tikimybė, jog kuriamoje sistemoje bus nepastebėtų klaidų, didžiausią dėmesį skirsime bendrų sistemos funkcijų testavimui, kuomet pateikdami sistemai nurodymus, gautą rezultatą sulyginsim su rezultatu, kurį tikimės gauti. Testavimo išsamumą riboja laikas, kurį galime skirti sistemos testavimui.

Testuosime laiką kviečiamų funkcijų pradžioje ir pabaigoje. Tačiau šį testavimą atliksime tik su atsirinktomis funkcijomis, nes kai kurios funkcijos, pavyzdžiu, aktyvumo vektorių paieška, gali trukti neribotą laiką (priklauso nuo eksperimentą atliekančio žmogaus, nes jis nustato trukmę).

6.4. Programinės įrangos testavimas, eksperimentų atlikimas

Testuojama eksperimentavimo sistema funkcinio vėlinimo testo generavimo metodui kurti, kuri automatiškai sukelia tarpinius duomenų failus ir rezultatų failus į jiems skirtus katalogus, suformuoja rezultatų lenteles. Testuojant atsižvelgiama į reikalavimų specifikaciją, ir norima, jog gauti rezultatai sutaptų su tais, kuriuos tikimės gauti.

6.4.1. Testavimo atvejai

Žemiau esančiose 22 – 27 lentelėse pateikiami naudoti testavimo atvejai sistemai testuoti.

22 lentelė. Prisijungimo / atsijungimo lango testavimas

Veiksmas	Pateikti duomenys	Laukiamas rezultatas	Gauti rezultatai
Įvedami teisingi prisijungimo prie sistemos duomenys	Teisingas prisijungimo paskyros vardas bei slaptažodis	Vartotojas sėkmingai prisijungia prie sistemos	Vartotojas sėkmingai prisijungė prie sistemos
Atsijungiama nuo sistemos	Užklausa atsijungti	Vartotojas atjungiamas nuo sistemos	Vartotojas atjungtas nuo sistemos
Įvedami neteisingi prisijungimo prie sistemos duomenys	Pateiktas neteisingas prisijungimo vardas, vėliau slaptažodis, ir trečią kartą – abu.	Vartotojas gauna pranešimą apie neteisingai įvestus prisijungimo duomenis	Vartotojas gavo pranešimą apie neteisingai įvestus prisijungimo duomenis

23 lentelė. Modulių tvarkymas

Veiksmas	Pateikti duomenys	Laukiamas rezultatas	Gauti rezultatai
Įvedamas naujas modulis	Naujo modulio pavadinimas	Naujas modulis sėkmingai įtrauktas į modulių sąrašą	Naujas modulis sėkmingai įtrauktas į modulių sąrašą
Koreguojamas modulio pavadinimas	Naujas modulio pavadinimas	Pakeistas modulio pavadinimas nauju	Pakeistas modulio pavadinimas nauju
Pasirinkto modulio	Užklausa trinti	Modulis sėkmingai	Modulis sėkmingai

trynimas	pasirinktą modulį	ištrintas iš modulių sąrašo	ištrintas iš modulių sąrašo

24 lentelė. Meniu elementų tvarkymas

Veiksmas	Pateikti duomenys	Laukiamas rezultatas	Gauti rezultatai
Įvedamas naujas meniu elementas	Naujo meniu elemento pavadinimas, adresas, pasirinktas modulis, tėvinis meniu elementas, atributas, reikšmė.	Naujas meniu elementas sėkmingai įtrauktas į meniu elementų sąrašą su visais jam priskirtais parametrais	Naujas meniu elementas sėkmingai įtrauktas į meniu elementų sąrašą su visais jam priskirtais parametrais
Koreguojamas meniu elementas	Naujas meniu elemento pavadinimas, adresas, pasirinktas modulis, tėvinis meniu elementas, atributas, reikšmė.	Pakeisti meniu elemento visi parametrai	Pakeisti meniu elemento visi parametrai
Pasirinkto meniu elemento trynimas	Užklausa trinti pasirinktą meniu elementą	Meniu elementas sėkmingai ištrintas iš meniu sąrašo	Meniu elementas sėkmingai ištrintas iš meniu sąrašo

25 lentelė. Vartotojų ir teisių tvarkymas

Veiksmas	Pateikti duomenys	Laukiamas rezultatas	Gauti rezultatai
Įvedamas naujas vartotojas	Naujo vartotojo paskyros vardas ir slaptažodis.	Naujas vartotojas sėkmingai įtrauktas į vartotojų sąrašą, o slaptažodis užkoduotas pasirinktu algoritmu	Naujas vartotojas sėkmingai įtrauktas į vartotojų sąrašą, o slaptažodis užkoduotas pasirinktu algoritmu
Keičiami vartotojo duomenys	Keičiamas vartotojo prisijungimo vardas	Pakeistas vartotojo prisijungimo vardas	Pakeistas vartotojo prisijungimo vardas

Keičiami vartotojo duomenys	Keičiamas vartotojo prisijungimo slaptažodis	Pakeistas vartotojo prisijungimo slaptažodis	Pakeistas vartotojo prisijungimo slaptažodis
Pasirinkto vartotojo paskyros trynimas	Užklausa trinti pasirinktą vartotojo paskyrą	Pasirinkta vartotojo paskyra sėkmingai ištrinta iš vartotojų sąrašo	Pasirinkta vartotojo paskyra sėkmingai ištrinta iš vartotojų sąrašo

26 lentelė. Aktyvumo vektorių tvarkymas

Veiksmas	Pateikti duomenys	Laukiamas rezultatas	Gauti rezultatai
Pasirenkama konfigūruoti aktyvumo vektorių modulį	Užklausa konfigūruoti aktyvumo vektorių modulį	Aktyvumo vektorių atvaizdavimo konfigūracijos langas	Aktyvumo vektorių atvaizdavimo konfigūracijos langas
Aktyvumo vektorių modulio konfigūracijos saugojimas	Aktyvumo vektorių modulio konfigūracijos nustatymai	Aktyvumo vektorių modulio nustatymai sėkmingai išsaugoti nurodytame faile	Aktyvumo vektorių modulio nustatymai sėkmingai išsaugoti nurodytame faile
Aktyvumo vektorių duomenų pradinių failų generavimas	Šablonas aktyvumo vektorių duomenų failui formuoti, katalogas kur formuoti duomenų failus ir svoriai	Pradiniai aktyvumo vektorių duomenų failai turintys nurodytus svorius sugeneruoti sėkmingai	Pradiniai aktyvumo vektorių duomenų failai turintys nurodytus svorius sugeneruoti sėkmingai
Aktyvumo vektorių generavimas	Skaičius nurodantis, kiek kartų generuoti, kelias iki schemą aprašančios bibliotekos, funkcijos pavadinimas, failas, kuriame bus saugomi rezultatai, ėjimo numeris, kuriam taikome algoritmą	Pasirinktam išėjimui aktyvumo vektoriai sėkmingai sugeneruoti	Pasirinktam išėjimui aktyvumo vektoriai sėkmingai sugeneruoti

Aktyvumo vektorių atvaizdavimas	Užklausa atvaizduoti aktyvumo vektorių duomenų failą pageidaujamu būdu	Aktyvumo vektoriaus duomenų failas atvaizduojamas pasiriktu būdu, ir aktyvios reikšmės išskiriamos spalvomis	Aktyvumo vektoriaus duomenų failas atvaizduojamas pasiriktu būdu, ir aktyvios reikšmės išskiriamos spalvomis

27 lentelė. Testinių sekų tvarkymas

Veiksmas	Pateikti duomenys	Laukiamas rezultatas	Gauti rezultatai
Pasirenkama konfigūruoti testinių sekų modulį	Užklausa konfigūruoti testinių sekų modulį	Testinių sekų atvaizdavimo konfigūracijos langas	Testinių sekų atvaizdavimo konfigūracijos langas
Aktyvumo testinių sekų modulio konfigūracijos saugojimas	Testinių sekų modulio konfigūracijos nustatymai	Testinių sekų modulio nustatymai sėkmingai išsaugoti nurodytame faile	Testinių sekų modulio nustatymai sėkmingai išsaugoti nurodytame faile
Testinių sekų generavimas	Nulio žymėjimas duomenų failuose, vieneto žymėjimas duomenų failuose, schemos pavadinimas, pilnas kelias iki pagrindinio aktyvumo vektorių katalogo, pilnas kelias iki katalogo, kuriame yra aktyvumo vektoriai, pilnas kelias iki katalogo, kuriame bus patalpinami sub katalogai su rezultatais	Pasirinktuose sub kataloguose sėkmingai sugeneruojamos testinės sekos	Pasirinktuose sub kataloguose sėkmingai sugeneruotos testinės sekos

Testinių sekų trynimas	Užklausa trinti testines sekas nurodytos schemos	Testinės sekos nurodytos schemos sėkmingai ištrintos	Testinės sekos nurodytos schemos sėkmingai ištrintos
Maišytų testinių sekų generavimas	Kelias iki sub katalogų, kuriuose patalpintos nemaišytos testinės sekos, kelias iki sub katalogų, kuriuose turi būti patalpintos maišytos testinės sekos	Pasirinktuose sub kataloguose sėkmingai tarpusavyje sumaišomos testinės sekos	Pasirinktuose sub kataloguose sėkmingai tarpusavyje sumaišytos testinės sekos
Testinių sekų peržiūrėjimas	Užklausa parodyti norimos schemos bei metodo testinių sekų duomenų failą	Parodomas norimos schemos bei metodo testinių sekų duomenų failas	Parodomas norimos schemos bei metodo testinių sekų duomenų failas

28 lentelė. TetraMAX rezultatų tvarkymas

Veiksmas	Pateikti duomenys	Laukiamas rezultatas	Gauti rezultatai
Pasirenkama konfigūruoti TetraMax modulį	Užklausa konfigūruoti TetraMAX modulį	TetraMAX modulio konfigūracijos langas	TetraMAX modulio konfigūracijos langas
TetraMax modulio konfigūracijos saugojimas	TetraMax modulio konfigūracijos nustatymai	TetraMax modulio nustatymai sėkmingai išsaugoti nurodytame faile	TetraMax modulio nustatymai sėkmingai išsaugoti nurodytame faile
TetraMax rezultatų failų generavimas	Pilnas kelias iki katalogo, kuriame patalpinti sub katalogai su porų duomenų failais viduje, schemos pavadinimas, kelias iki katalogo, kurio	Pasirinktuose sub kataloguose sėkmingai sugeneruojami TetraMAX rezultatų failai	Pasirinktuose sub kataloguose sėkmingai sugeneruoti TetraMAX rezultatų failai

	sub kataloguose bus patalpinti rezultatai, pilnas kelias iki katalogo, kuriame galima talpinti laikinus duomenų failus tarpiniams veiksmams atlikti, pilnas kelias iki Perl programos, pasirūpinančios TetraMAX testų atlikimu, pilnas kelias iki Perl programos, sukuriančio "PIS", "POS" bei "SEQ" tarpinius duomenų failus, pilnas kelias iki katalogo, kuriame laikomi testą aprašančių duomenų failai, pilnas kelias iki katalogo, kuriame laikomi klaidų modeliai		
TetraMAX rezultatų failų trynimas	Užklausa trinti TetraMAX rezultatų failus nurodytos schemos	TetraMAX rezultatų failai nurodytos schemos sėkmingai ištrinti	TetraMAX rezultatų failai nurodytos schemos sėkmingai ištrinti
TetraMAX rezultatų lentelės peržiūrėjimas	Užklausa rodyti TetraMAX rezultatų lentelę	Pavaizduojama TetraMAX rezultatų lentelė su perdavimo klaidų padengimu bei testo dydžio skaitinėm išraiškom	Pavaizduojama TetraMAX rezultatų lentelė su perdavimo klaidų padengimu bei testo dydžio skaitinėm išraiškom

Peržiūrėti pasirinktos schemos ir metodo TetraMAX rezultatų failą	Pasirinkta schema ir metodas	Parodomas pasirinktos schemos ir metodo TetraMAX rezultatų failas	Parodytas pasirinktos schemos ir metodo TetraMAX rezultatų failas

6.4.2. Automatizavimo testavimas

Testuojant pagal panaudotus atvejus sistemos veikimą ir funkcionalumą, įtraukta kodo dalis matavo laiką funkcijų iškvietimo pradžioje ir pabaigoje. Žemiau pateikiama rezultatų lentelė, gauta generuojant pradinius aktyvumo vektorių duomenų failus (29 lentelė. Pradinio aktyvumo vektorių duomenų failų formavimo trukmės atskaitos sekundėmis). Paryškintos lentelėje reikšmės, reiškia tarpus tarp komandos pasiuntimo ir reakcijos - tai laikas, kuris yra mums aktualus, nes įprastai daugiausiai laiko sugaištume rašydami komandas. Uždelsto laiko vidurkis formuojant pradinius aktyvumo vektorių duomenų failus: (4+6+4+5+5+4+5+5+4+5)/10 = 4.7 sek.

29 lentelė. Pradinio aktyvumo vektorių duomenų failų formavimo trukmės atskaitos sekundėmis

Testo nr.	Proceso etapas	Atskaitos laikas sekundėmis	Skirtumas tarp žemesnio ir aukštesnio stulpelių
1.	pradžia	2813	0
	pabaiga	2813	4
2.	pradžia	2817	0
	pabaiga	2817	6
3.	pradžia	2823	0
	pabaiga	2823	4
4.	pradžia	2827	0
	pabaiga	2827	5
5.	pradžia	2832	0
	pabaiga	2832	5
6.	pradžia	2837	1
	pabaiga	2838	4
7.	pradžia	2842	0

	pabaiga	2842	5
8.	pradžia	2847	0
	pabaiga	2847	5
9.	pradžia	2852	0
	pabaiga	2852	4
10.	pradžia	2856	0
	pabaiga	2856	5
11.	pradžia	2861	0
	pabaiga	2861	

Tokiu pačiu būdų suskaičiuoti uždelsimo laikai dominančių procesų pateikti žemiau esančioje lentelėje (30 lentelė. Uždelsimo laikas tarp procesų, naudojant automatizuotą sistemą). Susumavus visus laikus, gauname 4.7+8.4+2.2+5.2+2.5 = 23 sekundes.

30 lentelė. Uždelsimo laikas tarp procesų, naudojant automatizuotą sistemą

Procesas	Uždelsimo laikas (sekundėmis)
Pradinio aktyvumo vektorių duomenų failų formavimas	4.7
Surastų aktyvumo vektorių naudojimas kitose eksperimentuose	8.4
Testinių sekų sub katalogų kūrimas	2.2
Testinių sekų kūrimas	5.2
TetraMAX rezultatų failų generavimas	2.5

Neautomatizuotos sistemos atveju uždelsimo laiką reikalingą iškviesti procesą apskaičiuosime remdamiesi rašymo greičiu. Pasitelkus nemokamą internetinį įrankį, buvo nustatytas mano rašymo klaviatūra vidutinis greitis: 347 simboliai per minutę, arba 5.7 simboliai per sekundę, o tai yra remiantis testu (http://speedtest.aoeu.nl/) yra rezultatas, kuris lenkia 83.12% atlikusių šiuo įrankiu greito rašymo testą žmonių.

Suskaičiuota, jog atliekant 11 testų pasitelkiant tik dalį realizuotos automatizuotos sistemos funkcijų, paspartinančių darbą, reikėtų suvesti daugiau negu 10350 eilučių, kas mano rašymo greičiu užimtų apie 1815,8 sekundžių, arba apie 30.2 minutes. Palyginus su automatizuotu procesu, tai beveik 79 kartais ilgiau trunkantis procesas.

Procesas	Vidutinis simbolių kiekis, reikalingas atlikti 11 testų	Uždelsimo laikas (sekundėmis)
Pradinio aktyvumo vektorių duomenų failų formavimas	1000	175.4
Surastų aktyvumo vektorių naudojimas kitose eksperimentuose	2050	438.6
Testinių sekų sub katalogų kūrimas	700	122.8
Testinių sekų kūrimas	2500	438.6
TetraMAX rezultatų failų generavimas	4100	719.3

Tačiau verta paminėti, jog rankomis vedant komandas, įveliama klaidų, kurios yra taisomos, daroma žmogiškųjų klaidų, ir nurodomi netinkami keliai arba failai, taip gadinami galutiniai rezultatai, arba sugaištama papildomai laiko atliekant eksperimentus. Kuomet eksperimentų daug, ir jie periodiškai susisumuoja, šis laikas gali išaugti iki valandų, arba dienų.

6.4.3. Funkcinio vėlinimo testo generavimo metodų eksperimentas, testavimas

Atliekant sistemos testavimą pagal panaudotus atvejus, įvairiais funkcinio vėlinimo testo generavimo metodais buvo sukurtos testinės sekos ir remiantis šiomis sekomis buvo atliekami signalo perdavimo klaidų simuliacija. Iš Synopsys TetraMAX paketo automatiškai sugeneruotų rezultatų buvo suformuotos V1-V4, vieno signalo pokyčio ir visų aktyvių signalo reikšmių pokyčio metodų palyginimo lentelė, kuri pateikta žemiau esančiame paveikslėlyje (62 pav. Metodų palyginimo lentelė).

Ne minimizuotų testinių sekų signalo perdavimo padengimo rezultatai (klaidų padengimas-%, testo dydis)

Schema	1		V1		V2		V3		V4		visi	
	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis
c7552	99.45	932026	99.14	69877	99.13	69781	98.73	69877	98.77	69781	96.99	69881
c6288	100	20800	100	896	100	736	100	512	100	320	100	1408
c5315	99.90	508672	100	5856	100	12000	100	5920	100	5856	99.95	43909
c3540	99.98	46090	99.00	4095	99.50	4094	98.69	4095	99.41	4094	99.08	4095
c2670	100	40928	99.89	5195	96.48	5153	99.98	5195	96.42	5153	96.71	5195
c1908x	98.95	115132	98.45	7721	98.00	7721	97.96	7721	97.79	7721	97.79	7721
c1908	96.47	19309	94.72	2073	94.80	2073	95.03	2073	95.01	2073	95.26	2073
c1355	97.13	30462	99.91	4147	95.19	4147	94.39	4147	92.33	4147	93.31	4147
c880	99.83	22016	100	2048	100	2272	99.96	2370	99.42	2358	99.83	2370
c499	94.40	30903	99.83	4174	98.13	4174	93.53	4174	93.47	4174	92.54	4174
c432	97.24	9288	97.38	1073	96.81	1073	96.81	1073	96.67	1073	94.41	1073

62 pav. Metodų palyginimo lentelė

Analizuojant tik MIT testus, vidutiniškai daugiausiai signalų perdavimo klaidų padengė testinės sekos, sudarytos V1 metodu (31 lentelė. Signalų perdavimo klaidų padengimo vidurkiai).

31 lentelė. Signalų perdavimo klaidų padengimo vidurkiai

Schema	Klaidų padengimas procentais			
	V1	V2	V3	V4
C432	99.93	99.35	99.35	99.2
C499	99.83	98.13	93.53	93.47
C880	100	100	99.96	99.42
C1355	99.91	95.19	94.39	92.33
C1908	98.45	98	97.96	97.79
C2670	99.89	96.48	99.98	96.42
C3540	99	99.5	98.69	99.41
C5315	100	100	100	100
C6288	100	100	100	100
C7552	99.14	99.13	98.73	98.77
Vidurkis	99.6	98.58	98.25	97.68

Maišant metodus tarpusavyje, iš gautų rezultatų, sistema automatiškai suformavo palyginimo lenteles (63 pav. Maišytų tarpusavyje metodų analizės lentelės pirma dalis ir 64 pav. Maišytų tarpusavyje metodų analizės lentelės antra dalis).

Schema	1 ir V1		1 ir V2		1 ir V3		1 ir V4		1 ir visi		V1 ir V2		V1 ir V3		V1 ir V4	
	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis
c7552	99.49	1001903	99.49	1001807	99.45	1001903	99.48	1001807	99.46	1001907	99.18	139658	99.23	139754	99.24	139658
c6288	100	20800	100	20800	100	20800	100	20800	100	20800	100	896	100	896	100	896
c5315	100	510144	100	509920	100	510336	100	510784	99.97	552581	100	5856	100	5856	100	5856
c3540	99.99	50185	99.99	50184	99.99	50185	99.98	50184	99.98	50185	99.77	8189	99.83	8190	99.91	8189
c2670	100	40928	100	40928	100	40928	100	40928	100	40928	99.95	10348	100	7648	100	10240
c1908	96.47	21382	96.47	21382	96.53	21382	96.47	21382	96.58	21382	95.46	4146	95.73	4146	95.69	4146
c1355	99.94	34609	99.04	34609	99.10	34609	99.04	34609	97.13	34609	100	4608	99.94	8294	100	4608
c880	100	24064	100	24064	99.96	24386	99.92	24374	99.83	24386	100	2048	100	2048	100	2048
c499	99.83	35077	100	35040	98.60	35077	100	35040	94.40	35077	100	5728	100	5728	100	5728
c432	97.45	10361	97.38	10361	97.38	10361	97.45	10361	97.24	10361	97.45	2146	97.38	2146	97.45	2146

63 pav. Maišytų tarpusavyje metodų analizės lentelės pirma dalis

Schema	V1 ir visi		V2 ir V3		V2 ir V4		V2 ir visi		V3 ir V4		V3 ir visi		V4 ir visi	
	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis	%	Dydis
c7552	99.26	139758	99.24	139658	99.26	139562	99.23	139662	98.97	139658	98.81	139758	98.86	139662
c6288	100	896	100	736	100	736	100	736	100	512	100	512	100	320
c5315	100	5856	100	12000	100	12000	100	12000	100	5920	100	5920	100	5856
c3540	99.90	8190	99.94	8189	99.97	8188	99.87	8189	99.68	8189	99.98	8190	99.93	8189
c2670	100	9344	99.98	10348	97.13	10306	100	9280	100	9920	100	9344	100	5888
c1908	96.06	4146	95.87	4146	95.73	4146	95.85	4146	95.81	4146	96.33	4146	95.96	4146
c1355	99.94	8294	97.85	8294	95.22	8294	95.22	8294	94.99	8294	97.25	8294	95.22	8294
c880	100	2048	100	2272	100	2272	100	2272	99.96	4728	99.96	4740	99.92	4728
c499	99.83	8348	99.59	8348	98.13	8348	98.13	8348	94.93	8348	98.19	8348	98.13	8348
c432	97.38	2146	97.38	2146	97.31	2146	97.03	2146	97.03	2146	97.17	2146	97.24	2146

64 pav. Maišytų tarpusavyje metodų analizės lentelės antra dalis

Išanalizavus tarpusavyje maišytų metodų rezultatus, pastebėta, jog geriausią

vidutinį signalų perdavimo klaidų padengimą pasiekiantis testas yra gaunamas maišant V1 ir V2 arba V1 ir V4 metodus tarpusavyje (32 lentelė. Signalų perdavimo klaidų padengimo vidurkiai maišytuose MIT metoduose). Tačiau klaidų padengimas išauga vos 0.18%, o tuo tarpu testo dydis beveik padvigubėja.

32 lentelė. Signalų perdavimo klaidų padengimo vidurkiai maišytuose MIT metoduose

Schema	Klaidų padengimas procentais				
	V1irV2	V1irV3	V1irV4	V1irAVI	V2irV3
C432	100	99.93	100	99.93	99.93
C499	100	100	100	99.83	99.59
C880	100	100	100	100	100
C1355	100	99.94	100	99.94	97.85
C1908	98.89	98.7	98.66	98.74	98.58
C2670	99.95	100	100	100	99.98
C3540	99.77	99.83	99.91	99.9	99.94
C5315	100	100	100	100	100
C6288	100	100	100	100	100
C7552	99.18	99.23	99.24	99.26	99.24
Vidurkis	99.78	99.76	99.78	99.75	99.51

6.5. Testavimo išvados

Viso testavimo metu nebuvo gauta jokių pranešimų apie klaidas iškylančias sistemos veikimo metu, todėl eksperimentavimo sistema funkcinio vėlinimo testo generavimo metodui kurti parašyta be akivaizdžių klaidų.

Atsižvelgus į reikalavimų specifikaciją, sukurta sistema tenkina pagrindinius užsakovo reikalavimus. Atlikus testus, nustatyta, jog automatizuota sistema paspartina testavimo metodų testavimo eksperimentą, o pasiūlytas V1 metodas padengia daugiau signalo perdavimo klaidų ir testo dydis išlika nedidelis palyginus su testais, gaunamais maišant metodus tarpusavyje.

7. Išvados

Apžvelgus įvairius literatūros šaltinius, buvo išanalizuoti funkcinio vėlinimo testų generavimo metodai ir nustatyta, jog geriausi rezultatai gaunami maišant skirtingus testų sudarymo metodus. Geriausi rezultatai, pasak šaltinių, yra kuomet SIT testai, maišomi su MIT, nes vien tik SIT jau nebeužtenka. Bendraujant su funkcinio vėlinimo srities specialistais, buvo pasiūlyti keturi funkcinio vėlinimo testų generavimo metodai, kurie remiasi schemos elgsenos modelio analize. Pagal šį aprašą generuojant funkcinio vėlinimo testus, galima sutaupyti bendrą schemos paruošimo rinkai laiką, nes testų generavimas gali būti atliekamas dar nepradėjus loginės schemos modelio sintezės, tačiau turint funkcinį schemos aprašą, kitaip dar vadinamą „juodos dėžės“ modeliu.

Taip pat ištirta 10 daugiausiai funkcijų turinčių turinio ir informacijos tvarkymo sistemų, iš kurių nei viena savo savybėmis netiko būti tėvine sistema, kuriamai eksperimentavimo sistemai funkcinio vėlinimo testo generavimo metodui kurti. Atlikus detalią reikalavimų specifikaciją, išsiaiškinti svarbiausi funkciniai reikalavimai, labiausiai susiję su duomenų atvaizdavimu bei generavimu. Šiems reikalavimams skiriant daugiausiai dėmesio, buvo suprojektuota eksperimentavimo sistema. Pagal sukurtus projekto modelius ir diagramas, visi testavimo proceso etapai, nuo aktyvumo vektorių suradimo iki Synopsys TetraMAX rezultatų lentelės, rodančios metodų gerumą formavimo, buvo dalinai automatizuoti. Sukuriant sistema yra paremta vidinės įmonės portalo veikimo idėjomis, siekiant integruoti programinę įrangą. Sukurta sistema automatiškai sudėlioja duomenų ir rezultatų failus į jiems skirtus katalogus ir prireikus atvaizduoja arba panaudoja juose esančią informaciją. Tokiu būdu išvengiama galimų žmogiškųjų klaidų, dėl kurių kaltės gali tekti kartoti eksperimentą ir patirti finansinių, arba laiko išlaidų. Sistema taip pat analizuoja galutinius klaidų padengimo failus ir suformuoja metodų palyginimo lentelę.

Atliekant testavimus, nustatyta, jog sistema veikia be klaidų, o vykdant testavimo procesų eigą su eksperimentavimo sistema, užtrunkama beveik 79 kartus trumpiau, negu užtruktų žmogus atlikdamas testus. Atliekant testus buvo sugeneruoti funkcinio vėlinimo testų generavimo metodais paremti klaidų padengimo rezultatai. Iš jų nustatyta, jog V1 metodas iš visų keturių siūlytų, yra geriausias, nes vidutiniškai padengia daugiausiai signalo perdavimo klaidų (99.6%), lyginant su kitais nemaišytais metodais. Šiuo metodu sukurto testo dydis palyginus su nežymiu 0.18% daugiau klaidų

padengiančių tarpusavyje maišomų metodų rezultatais yra dvigubai mažesnis, kas daro V1 metodą geresniu pasirinkimu. Pagal pastarąjį metodą, testai yra generuojami iš aibės $A0^j$ ($A1^j$) aktyvumo vektorių, kurių kiekvienas yra laikomas antru testinės poros vektoriumi. Tuo tarpu iš aibės $A1^j$ ($A0^j$) parenkamas pirmasis sudaromos testinės poros vektorius, turintis daugiausiai priešingų reikšmių antro vektoriaus aktyvioms reikšmėms.

8. Literatūros sąrašas

[1] Kapur R., Chandramouli R., Williams T.W., „Strategies for Low-Cost Test“, IEEE Design and Test of Computers, ISSN: 0740-7475, 2001m. Peržiūrėta: 2008-11. Prieiga internete:

http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00970423

[2] Bareiša E., Jusas V., Motiejūnas K., Šeinauskas R., „The Realization-Independent Testing Based on the Black Box Models“, Informatica, Vol. 16, No. 1, p. 19-36, 2005m. Peržiūrėta: 2009-02-10. Prieiga internete:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.122.9367&rep=rep1&type=pdf

[3] Krstic A., Jing-Jia Liou, Kwang-Ting Cheng, Wang Li-C., “On Structural vs. Functional Testing for Delay Faults”, Quality Electronic Design, 2003m. Peržiūrėta: 2009-01. Prieiga internete:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1194772

[4] Bareiša E., Jusas V., Motiejūnas K., Šeinauskas R., „Development of Functional Delay Tests“, Digital System Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on, ISBN: 978-0-7695-3277-6, 2008m. Peržiūrėta: 2009-01-23. Prieiga internete:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4669293

[5] Bareiša E., Jusas V., Motiejūnas K., Šeinauskas R., „The Criteria of Functional Delay Test Quality Assessment“, Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on, ISBN: 978-0-7695-2978-3. Peržiūrėta: 2009-01-17. Prieiga internete:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4341470

[6] Joonhwan Yi, John P. Hayes, „High-Level Delay Test Generation for Modular Circuits“, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 3, March 2006, p. 576-590. Peržiūrėta: 2009m. Prieiga internete:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1597390&userType=&tag=1

[7] „TetraMAX® ATPG User Guide“ Synopsys, 2006m. Peržiūrėta: 2009m. Prieiga internete:

http://solvnet.synopsys.com

[8] Butkienė R., Čeponienė L., Nemuraitė L., „Informacinių sistemų inžinerijos magistrų darbų metodiniai nurodymai“, Kaunas 2008m. Peržiūrėta: 2009m.

[9] Biddick M., „Autonomic Computing: Vision VS. Reality“, straipsnis, 2006-10-26. Peržiūrėta: 2009m. Prieiga internete:

http://goliath.ecnext.com/coms2/gi_0198-360272/AUTONOMIC-COMPUTING-VISION-VS-REALITY.html

[10] Chernicoff D., Perschke S., „Business process automation. Managing Cost in Your Enterprise“, 2007m. Peržiūrėta: 2009m. Prieiga internete:

http://www.networkautomation.com/automate/ebook/

[11] Vaishnav B., Kokku R., Mudigonda S., „Open Source Technology Solutions for Business Process Automation“, 2008m. liepa. Peržiūrėta: 2009m. Prieiga internete:

http://www.ciol.com/resources/UserFiles/developer/Open_Source_WP.doc

[12] Bareiša E., Jusas V., Motiejūnas K., Šeinauskas R., „Transition Faults Testing Based on Functional Delay Tests“, 2007m., ISBN: 1-4244-1162-9. Peržiūrėta: 2009m. Prieiga internete:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4295315

[13] Niraj Jha, Sandeep Gupta, „Testing of Digital Systems“, Cambridge, 2003 m., ISBN: 0-521-77356-3. Peržiūrėta: 2009m.

[14] Jusas V., Motiejūnas K., „Generation of Functional Delay Test with Multiple Input Transitions“, Information Technology And Control, Kaunas, Technologija, 2007, Vol. 36, No. 3, p. 259 - 267. Peržiūrėta: 2009m. Prieiga internete:

http://itc.ktu.lt/itc363/Jusas363.pdf

[15] Jusas V., Smilingis M., Šeinauskas R., „Functional Delay Test Generation Approach Based On Extracting Information from the Software Prototype“, IT 2009, ISSN: 2029-0020. Peržiūrėta: 2009m.

[16] Informacinis portalas „CMS Match“, „Top 10 CMS Scored By Functionality“, 2009m. Peržiūrėta: 2009m. Prieiga internete:

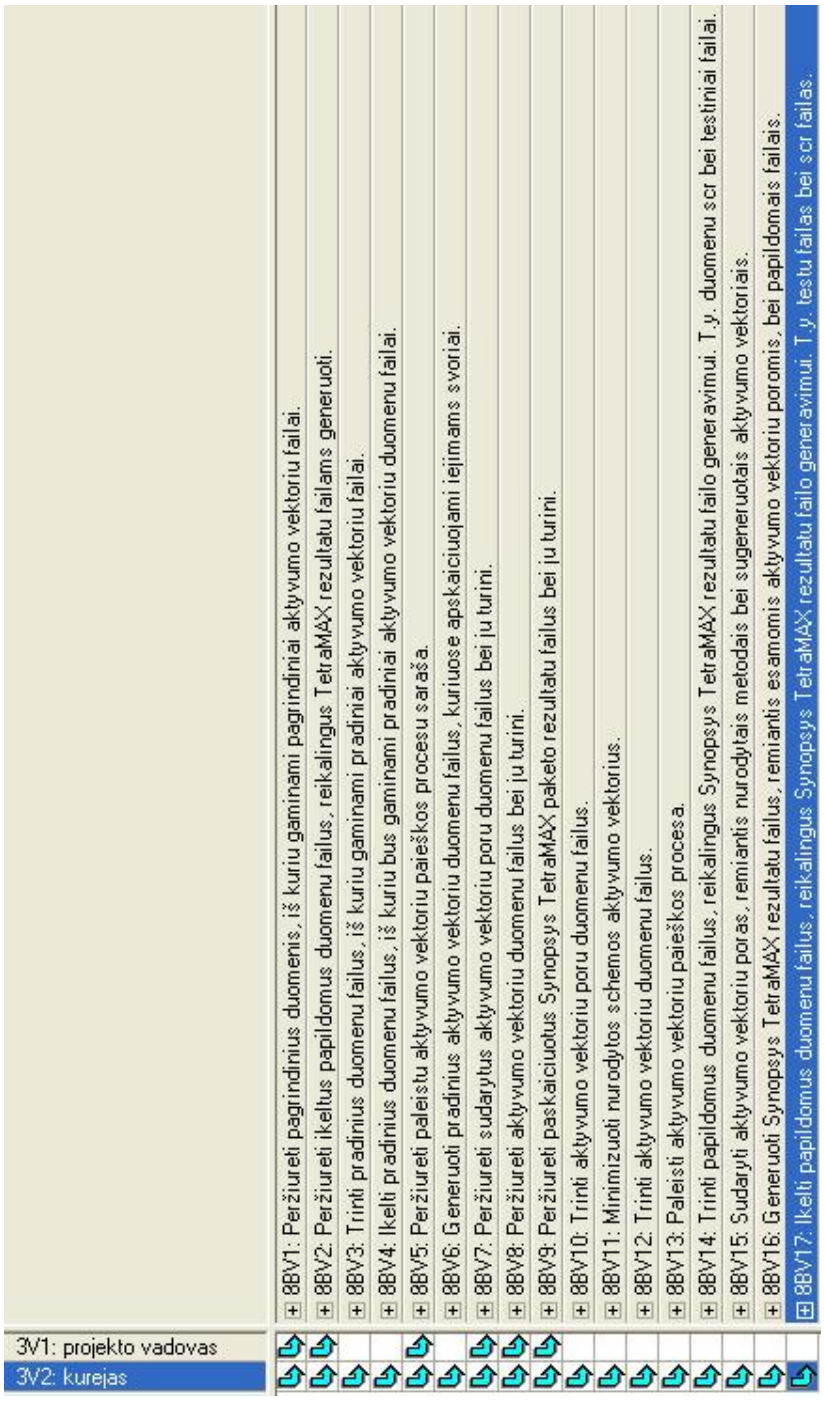
http://www.cmsmatch.com/news/top-10-cms-scored-functionality.html

[17] Robertson J., Robertson S., „Volere Requirements Specification Template“, Edition 13, August 2007. Peržiūrėta: 2008m.

[18] Nemuraitė L., „Informacinių sistemų projektavimas UML CASE įrankiais“, 2008m. Peržiūrėta: 2009m.

9. Priedai

Priedas nr. 1: Vartotojų ir panaudojimo atvejų susietumo matrica



65 pav. Vartotojų ir panaudojimo atvejų susietumo matrica

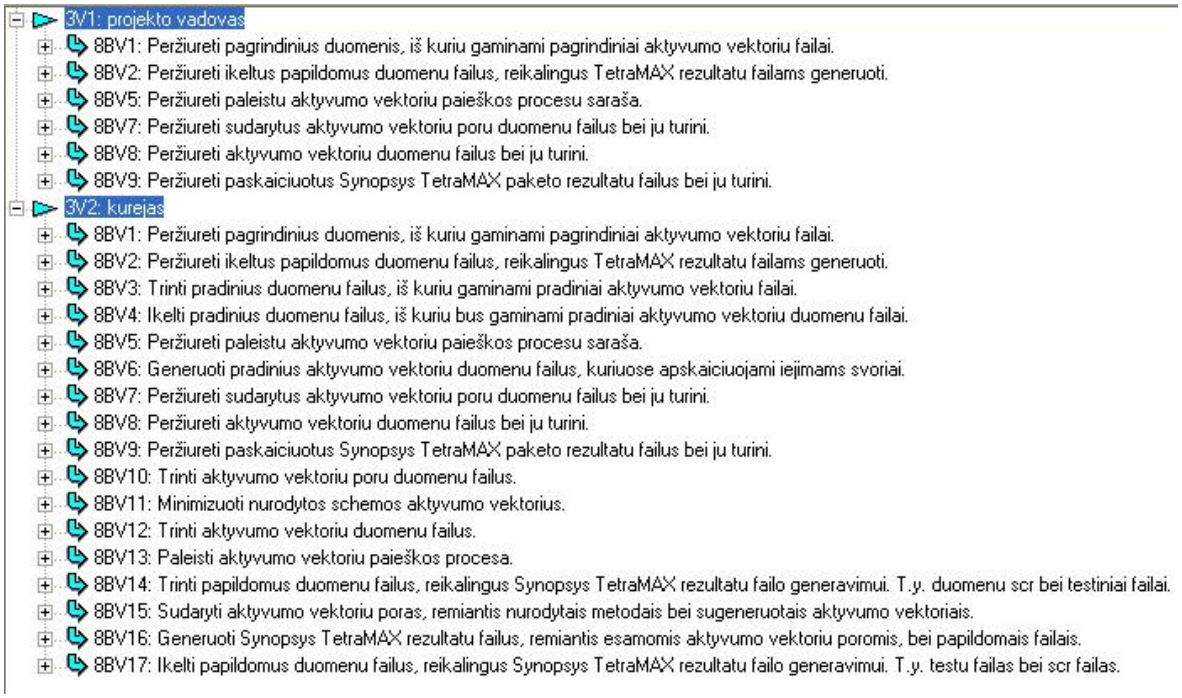
Priedas nr. 2: Svarbiausi funkciniai reikalavimai, surikiuoti pagal užsakovo patenkinimą

Sutapus užsakovo patenkinimo įverčiui – rikiuojama pagal užsakovo nepatenkinimą nuo labiausiai neįvykdžius reikalavimo nepatenkinamo įverčio iki žemiausio.

Requirements:	Užsakovo pat	Užsakovo nep
	1 - Flt:N Str:D	2 - Flt:N Str:D
9AV4: Synopsys TetraMAX rezultatu duomenu lenteleje turi buti funkcinio velinimo klaidu padengimas išreikštas procentais bei testo dydis.	5	5
9AV5: Pateiktos aktyvumo vektoriu poru lenteles duomenys turi buti suskirstyti pagal panaudotus metodus.	5	5
9AV8: Pateikto aktyvumo vektoriu lenteleje, turi buti galimybe pasirinkti, kaip norima peržiureti aktyvumo vektorius: su antraštemis ar be antrašciu.	5	3
9AV9: Visose duomenu failu peržiuros lentelese, turi buti atliekamas rikiavimas pagal kokia nors vienareikšme taisykle.	4	5
9AV3: Paspaudus ant lenteleje esancio egzistuojancio duomenu failo žymeklio, turi buti parodomas duomenu turinys.	4	4
9AV6: Peržiurint aktyvumo vektoriu bet kokiu budu (su antraštem arba be antrašciu) skirtingomis spalvomis išskiriamos aktyvios reikšmes.	4	2
9AV1: Sistema turi neleisti minimizuoti aktyvumo vektoriu duomenu failo, jeigu yra paleistas procesas susijes su juo.	3	2

66 pav. Svarbiausi funkciniai reikalavimai, surikiuoti pagal užsakovo patenkinimą

Priedas nr. 3: Vartotojų keliamų reikalavimų susietumo medis



67 pav. Vartotojų keliamų reikalavimų susietumo medis

Priedas nr. 4: Kodo fragmentas, skirtas laikui fiksuoti testavime

```
#!/usr/bin/perl -w
use strict;
my $name = $ARGV[0];
my ($second, $minute, $hour) = localtime();
my $theTime = "$hour:$minute:$second";
my $fh;
open($fh, '>>', 'laikai.txt') or die($!);
print $fh "$name-$theTime\n";
close($fh);
```

Priedas nr. 5: Straipsnis: "Functional Delay Test Generation Approach Based on Extracting Information From The Software Prototype"

FUNCTIONAL DELAY TEST GENERATION APPROACH BASED ON EXTRACTING INFORMATION FROM THE SOFTWARE PROTOTYPE

Vacius Jusas[1], Mantas Smilingis[2], Rimantas Šeinauskas[2]

[1]*Software Engineering Department, Kaunas University of Technology, Studentų 50-404, LT-51368 Kaunas, Lithuania*
[2]*Information Technology Development Institute, Kaunas University of Technology, Studentų 48A, LT-51368 Kaunas, Lithuania*

Abstract. The suggested approach for test generation is based on extracting information about device behaviour from the software prototype. This approach presents a new attitude towards the problem of the test generation and it enables generating high quality tests in initial design stages when just the software prototype of the device under design is available. Functional test constructed detects near to all transition faults at the gate level of the benchmarks examined. The tests for the device can be generated in parallel with the design activities of the device, and deterministic test generation methods can be used before logical synthesis of the device is completed. When the synthesis of the device is completed, the functional delay test can be minimized, adjusting it to the particular structural implementation. The best test construction approach was selected among the considered ones and the further activities on the test quality improvement were considered.

1. Introduction

Generally test generation is performed after logical synthesis of the device is completed. In this case, test generation has an effect on the overall duration of a design cycle of the device. Therefore, the efforts are made to compile tests in the initial stages of the device design reducing the costs of test preparation after logical synthesis of the device is completed. A software prototype of a device is composed in the initial design stages; this prototype is used for verifying device behaviour and analyzing it. Such a software prototype enables the given input stimuli to estimate the output responses using simulation. We will discuss the use of device's software prototype for test generation.

The device under design has the primary inputs and the primary outputs, meanwhile the software prototype operates with the variables consisting of several bits. Therefore, the bits of input variables are linked to the primary inputs of the device and the bits of output variables are linked to the primary outputs of the device. We will assume that the association between the bits of input/output variables and the primary inputs/outputs of the device is known.

The development of the methods of the test generation at the behavioral level is closely related to the following two activities: 1) the development of the new functional faults models and the methods of test generation based on software prototype; 2) the extraction of the information from the software prototype for deterministic test generation using simulation.

The second activity was not clearly defined until now and it is more promising as one can use results that are gathered while developing deterministic test generation methods for many years. It has to be mentioned that information obtained from the software prototype using simulation can be incomplete; however, it can be used as the basis for the test generation using the deterministic methods.

The investigated problem allows estimating the device behavior on the base of the software prototype. The device behaviour can be partly described by the input stimuli with the labeled values of the active inputs that are named as the activity vectors. The input is active if the change of its value into the opposite one changes the output value. The activity vector is considered essential if the values of its active inputs differ from the values of active inputs of the other vectors. The essential activity vectors, which are obtained using random search methods and simulation, can describe the device behavior incompletely, however, the deterministic tests can be generated on their basis. The deterministic methods of test generation can be modified in such a way that the test generation may use incomplete information about the device behavior.

The functional test is generated for all the possible implementations of the device; therefore, its length is much larger in comparison with the length of the test for the particular implementation. When the synthesis of the device is completed, the functional test redundancy can be eliminated by removing such tests that do not detect new faults of the particular implementation. Therefore the length of the functional test is not a critical parameter. However, it is purposeful to have several functional tests of the increasing completeness and of the increasing length at the same time. A more complete test uses more input stimuli that define device functioning. The least complete test is analyzed first of all. If it does not detect all the faults of the particular implementation, a more complete test is analyzed further. During the analysis of device faults, one estimates the output sub-circuits in regard to the undetected faults and the more complete test is analyzed only for such output sub-circuits. The tests with different level of the completeness can be generated easily on the base of the activity vectors.

2. Related work

Models of physical faults are needed at higher levels of abstraction in order to be able to develop test patterns from functional or behavioral description. Researchers have experienced that the stuck-at fault model works quite well at logic level. Many efforts have been devoted to the problem of finding behavioral level fault model. But no such fault model has been discovered at behavioral or higher level which is universally accepted.

Behavior level fault models can be broadly classified into two main categories: 1) fault models related to the description code [4]-[8], [12]; 2) black-box fault models related to input stimuli and output responses [1], [9], [15], [16]. Testing at higher level of abstraction has a lot in common with software testing. Therefore, the pattern generation methods based on the fault model related to the description code can be further classified, namely, code oriented methods and fault oriented methods. The code oriented methods exploit the most widely used metrics developed for automated software testing: statement coverage [7], branch coverage [6] and path coverage [12]. Although there are similarities there are also important differences due to different sources of errors/faults and models in these two cases. The purpose of software validation is to detect design errors whereas the purpose of testing is to detect physical defects and fabrication faults.

The fault oriented methods use single bit stuck-at fault model [8], which was firstly introduced in [5], and the variable bit stuck-at fault model [4]. The variable stuck-at fault model means that the variable is stuck-at a particular value. Multiple bit stuck-at faults where all bits have a stuck-at fault are equivalent to variable stuck-at faults. Together with bit stuck-at fault models, a condition stuck-at fault, which means that a condition is either stuck-at true or stuck-at false, is used [8]. These models have been derived from the logic level stuck-at fault model but they do not give adequate coverage of physical faults. Faults inside elements that implement operators cannot be modeled in this way. To resolve this problem the fault oriented methods [4] use the operator mutation fault model. This fault model implies that the operator will make a miscalculation for a subset of operand values. It is obvious that for an operator with a large number of inputs, it is practically impossible to enumerate all possible operator mutations and then generate test patterns to test them.

Black-box fault models are more universal as they do not depend on the description code; however, such black-box fault models are of little use still. Several black-box fault models were suggested that do not examine the description code, and they are based on the input stimuli and the output responses [1], [9], [15], [16]. The most universal is the / the design process, while the synthesized description of the device is not available yet. The very promising results are achieved in functional test generation for detecting stuck-at faults when the generation is based on PP fault model [1]. The average percent of undetected faults did not exceed 0.5% for ISCAS'85 benchmark circuits. The test sets for PP faults are larger than the test sets generated at the gate level by TetraMAX only 6 times on average.

The functional delay fault is denoted as follows [11], [14]: (I, O, $t\ I$, $t\ O$), where $t\ I$ is rising (r) or falling (f) transition on input I and $t\ O$ is rising (r) or falling (f) transition on output O. If we compare the functional delay and PP fault models we see that both models have almost the same meaning with one distinction: the functional delay model is intended for detection of malfunctions in the dynamic behavior of module and the PP fault model – for detection of malfunctions in the static state of module. Based on this observation, we can define how to extend the PP fault test to the functional delay fault test. Every input pattern that detects PP faults is transformed only into k input pattern pairs in such a way that the single signal value transition occurs on every input that is associated with PP fault detection on the considered test pattern [2]. There is another way described in [3] to obtain functional delay fault tests from PP tests. By applying the approach from [3] every input pattern that detects PP faults is transformed only into one input pattern pair in such a way that the signal value transition occurs on every input that is associated with PP fault detection on the considered test pattern. Consequently, if the test for PP faults consists of p input patterns the constructed functional delay fault test has p input pattern pairs, as well.

Thus, the obtained test is much shorter than by applying single transition test. The experiments on ISCAS'85 benchmark circuits demonstrated the test shortening of 3.8 times on average [3]. However, the test pattern pairs constructed by applying the approach from [3] possess the change of signal value on more than one input. Therefore, these pattern pairs are multi-input transition (MIT) tests [10] and some of functional delay faults that are functional robustly detectable on single-input transition (SIT) test may become functional non-robustly detectable [10] or even worse not detectable on considered test pattern pair, because some activation conditions needed for signal transition propagation from particular input to particular output may be corrupted. It is desirable that the generated MIT test would guarantee the function-robust propagation (FRP) property. According to definition from [10] a transition *t I* on input *I* is function-robustly propagated as a *t O* transition to output *O* when the signal value on *O* changes *if and only if* the signal value on *I* changes. Our experiences on functional delay fault testing revealed that not all the delay faults of the circuits can be detected by SIT tests. Some delay faults require MIT tests. MIT test launches several signal transitions on the inputs of the circuit at the same time. Some of these signal transitions can overlap or partly block each other. Therefore, some delay fault effects can be propagated as the signal glitches, which can be observed and measured by the test equipment. Such a propagation of the fault effect is called a *weak non-robust propagation* [13]. A conventional transition fault test uses the weak non-robust propagation [13]. Therefore, it is meaningful to think of such the rules that allow generating the functional delay test, which enables the weak non-robust propagation of the fault effect. When the fault effect propagates as the signal glitch, there is no signal transition on the output, if we observe it in the static mode.

The combination of several functional delay tests obtained in the different ways allows having the delay test that detects more than 99% of the transition faults [3]. In this paper, we show that the software prototype, which is used for the test generation, could be replaced by the activity vectors obtained from the software prototype. The activity vectors retain the information on the functioning of the device. We introduce the activity vectors in the next section. We present a test generation approach using the activity vectors and we report the results of the experiment in Section 4. We finish with conclusions in Section 5.

3. Activity vectors

Let's say, the software prototype model has n inputs and m outputs. We denote the input stimulus by $P=<p_1, p_2, \ldots, p_i, \ldots, p_n>$, where $p_i= \{0, 1\}$, $i=1, 2, \ldots, n$. The activity vector $P^j=<p^j_1, p^j_2, \ldots, p^j_i, \ldots, p^j_n>$ is associated with output j. A component of the activity vector can take on one of the following values: 0, 1, *N*, *V*. The value *V* shows that the complement of the value 1 on the input i changes the value to the opposite on the output j. The value *N* shows that the complement of the value 0 on the input i changes the value to the opposite on the output j. The activity vectors $P1^j$ set the value 1 on the output j, meanwhile the activity vectors $P0^j$ set the value 0 on the output j. The values *V* and *N* are the active values. The activity vector summarizes $n + 1$ input stimuli that differ only by single value. Let's say, we assign the following input stimulus $<X_1, X_2, X_3, X_4, X_5> = <01011>$ for the benchmark circuit C17 (Figure.1). This input stimulus sets the value 1 on the output *y1*. We complement every value of this stimulus one by one and we derive the following activity vector: <0*VN*11>. The activity vector summarizes the following input stimuli: <01011>, <11011>, <00011>, <01111>, <01001>, <01010>. These input stimuli set the value 1 on the output, except the third one and the fourth one.

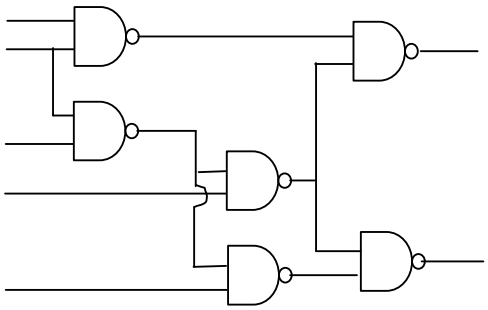


Figure.1. Benchmark circuit *C*17

M activity vectors $P1^j$ or $P0^j$ can be derived for every input stimulus *P* (*M* value denotes the number of outputs). The activity vector P_a can cover the activity vector P_b, and we will denote $P_a > P_b$. The activity vector P_b has the active values only on the same inputs as the activity vector P_a, and the active values of the vector P_b are equal to the active values of the vector P_a on the same inputs. If the active values of the vectors P_a and P_b are the same, then the activity vectors P_a and P_b are equal. The prerequisites of covering the vector P_b by the vector P_a are presented in Table 1.

Table 1. The prerequisites of covering

P_a	V	N	V	V	N	N
P_b	V	N	1	0	0	1

The activity vector P_a covers the activity vector P_b, if at least one of the conditions, which are in the last four columns of Table 1, is satisfied. The vectors, which are not covered by the other vectors, are essential. After analysis of input stimuli, the sets of essential vectors $A1^j$ and $A0^j$ are formed for every output j. The vectors of set $A1^j$ set the value 1 on the output j, while the vectors of set $A0^j$ set the value 0 on the output j.

Let's consider the benchmark C17 presented in Figure.1. The input stimulus $P = <X_1, X_2, X_3, X_4, X_5> = <01110>$ sets the following output values: $<y1, y2> = <00>$. The results of complementing every input value one by one are presented in Table 2. We obtain the following activity vectors according to this table: $P0^{y1} = <N1VV0>$, $P0^{y2} = <01VV0>$. In the same way, for the input stimulus $P = <00110>$ we obtain the following activity vectors according to Table 3: $P0^{y1} = <N0110>$, $P0^{y2} = <00110>$. These vectors are not essential, because they are covered by the previous vectors.

Table 2. The complement of input values of stimulus <01110>

p_1	p_2	p_3	p_4	p_5	y_1	y_2
0	1	1	1	0	0	0
1	1	1	1	0	1	0
0	0	1	1	0	0	0
0	1	0	1	0	1	1
0	1	1	0	0	1	1
0	1	1	1	1	0	0

Table 3. The complement of input values of stimulus <00110>

p_1	p_2	p_3	p_4	p_5	y_1	y_2
0	0	1	1	0	0	0
1	0	1	1	0	1	0
0	1	1	1	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	1	0	0

After analysis of all the possible input stimuli, we obtain the following essential activity vectors:

$A0^{y1}$ = {< $N1VV0$>, <$1NN10$>, <$NN100$>}

$A1^{y1}$ = {< $V0V10$>, <$0V1N1$>, <$0VN10$>}

$A0^{y2}$ = {< $1N01N$>, <$11VV1$>}

$A1^{y2}$ = {< $101NV$>, <$10N1V$>, <$1V1N0$>, <$0VN10$>}

We notice that the active values of the essential activity vectors correspond to the variables of the terms of the direct and inverse logical functions. Then, we obtain the following logical functions:

$$y1 = X_1X_3 + X_2\overline{X}_4 + X_2\overline{X}_3$$

$$\overline{y1} = \overline{X}_1X_3X_4 + \overline{X}_2\overline{X}_3 + \overline{X}_1\overline{X}_2$$

$$y2 = \overline{X}_4X_5 + \overline{X}_3X_5 + X_2\overline{X}_4 + X_2\overline{X}_3$$

$$\overline{y2} = \overline{X}_2\overline{X}_5 + X_3X_4$$

But the complete correspondence not always exists between the values of the essential activity vectors and the variables of the terms of the logical function. It is possible to think of the example when the active values of the activity vectors are subset of the variables of the terms of the logical function. Let's consider the logical function TTF of three logical variables:

$$Y = X_1X_2X_3 + \overline{X}_1\overline{X}_2\overline{X}_3$$

$$\overline{Y} = X_1\overline{X}_2 + X_1\overline{X}_3 + \overline{X}_1X_2 + X_2\overline{X}_3 + \overline{X}_1X_3 + \overline{X}_2X_3$$

After analysis of all the possible input stimuli, we obtain the following essential activity vectors:

$A1$ = {< VVV>, <NNN>}

$A0$ = {< $V00$>, <$0V0$>, <$00V$>, <$N11$>, <$1N1$>, <$11N$>}.

That corresponds to the following terms of the logical function:

$$Y = X_1X_2X_3 + \overline{X}_1\overline{X}_2\overline{X}_3$$

$\overline{Y} = X_1 + X_2 + X_3 + \overline{X}_1 + \overline{X}_2 + \overline{X}_3$

We notice that the active values of the activity vectors of the set $A0$ form the incomplete terms of the inverse logical function. The active values of the activity vectors can produce the incomplete terms if all the possible input stimuli are not considered. Such a situation arises for the large circuits. But that it is not the case for the example of the logical function TTF.

Conjecture. The active values of the essential activity vectors of the sets $A1^j$ and $A0^j$ of the output j correspond to the complete or incomplete terms of the direct and inverse logical function.

We can not prove this conjecture, but, on the other hand, we were unable to find the example that would contradict to our conjecture. The investigation is difficult because the logical function of the output can be expressed in many different ways. The logical function is not obligatory minimal in all the cases.

Because the essential activity vectors correspond to the complete or incomplete terms of the logical function of the outputs, there is a possibility to check whether the output responses of the synthesized circuit not contradict to the existence of the term of the logical function. Let's consider how it is possible to determine the membership of the term in the logical function of the output. The term consists of the input logical variables. The variable of the term can be in complemented or uncomplemented form. The term is completely defined if the input stimulus assigns to the uncomplemented variables the value 1, whereas the input stimulus assigns the value 0 to the complemented variables. The term $X_1, \overline{X}_2, X_3$ will be completely defined, if the value 1 will be assigned to the variables X_1 and X_3, and the value 0 – to the variable X_2. The input stimulus, which completely defines the term of the direct (inverse) logical function, sets the value 1 (0) on the output. If the term $X_1, \overline{X}_2, X_3$ belongs to the direct logical function, then any input stimulus, which completely defines the considered term, sets the value 1 on the output. If the term $X_1, \overline{X}_2, X_3$ belongs to the inverse logical function, then any input stimulus, which completely defines the considered term, sets the value 0 on the output. Generally, the term of the logical function determines the input stimuli that set the same value on the output.

Condition 1. All the input stimuli, which completely define the term, always set the same value on the output.

This condition is mandatory, but not sufficient. Any two terms of the logical function, the variables of which do not contradict each other, will satisfy Condition 1. The Condition 2 defines the prerequisites for the single term.

Condition 2. Every variable of the term has the corresponding input stimulus that the value change assigned to the variable of the term invokes the value change on the output.

Let's assume that the accordance to the Condition 1 and Condition 2 confirms the existence of the term of the logical function. The activity vector does not support completely the both conditions of the existence of the term. Firstly, the only n-k input stimuli, which completely define the term, are considered, where n – the number of inputs, k – the number of the active inputs. Additionally, the accordance to the Condition 2 is proved only for a single input stimulus. For example, the Condition 2 for the term $X_1 \overline{X}_2$ of the logical function TTF of three variables is satisfied by two input stimuli 100 and 101. Therefore, when we consider the single input stimulus, we can not obtain a minterm, but only the term $X_1 \overline{X}_2$. Based on this observation, we will introduce the Rule 1 specifying how to construct the minterms from the terms.

Rule 1. The two terms defined by the activity vectors can be combined into a single term, if the combined terms have the different variables and the values of the inactive inputs are the same as the values of the active inputs.

The constructed term has to satisfy the Condition 1 and Condition 2. Based on the Rule 1 for the logical function TTF, we obtain the following essential activity vectors: $A0$ = {<V00>, <0V0>, <00V>, <N11>, <1N1>, <11N>}. The combination of <V00> and <1N1> allows obtaining the term $X_1 \overline{X}_2$, and the combination of <V00> and <11N> allows obtaining the term $X_1 \overline{X}_3$. In such a way, we can obtain all the terms of inverse function: $\overline{Y} = X_1\overline{X}_2 + X_1\overline{X}_3 + \overline{X}_1X_2 + X_2\overline{X}_3 + \overline{X}_1X_3 + \overline{X}_2X_3$. This logical function is not minimized. If we combine the activity vector with the single other activity vector, we could obtain the minimal logical function: $\overline{Y} = X_1\overline{X}_2 + X_2\overline{X}_3 + \overline{X}_1X_3$.

The size of the sets of the activity vectors $A1^j$ and $A0^j$ directly depends on the size of the set of the input stimuli considered. The number of the activity vectors is directly proportional to the number of the terms of the logical function of the output. Therefore, after finding the appropriate number of the activity vectors, this number does not increase more. We could use this feature for the test generation. The input stimuli are generated randomly and selected only those ones that increase the number of the essential activity vectors of the set $A1^j$ or the set $A0^j$. The selected input stimuli according to this rule can be used

as the test for the device. The generation of the random input stimuli becomes ineffective, when the generation process does not lead to the selection of the new input stimuli. If the generation does not lead to the increase of the sets $A1^j$ or $A0^j$ during the predefined time limit, the generation is stopped. Based on this simple algorithm, the delay tests were generated for the ISCAS'85 benchmark circuits. The results are presented in Section 4.

The random generation is not the most effective way to find the activity vectors. We noticed that the process of finding the activity vectors is more effective, if we use the adjacent generation of the stimuli for the selected ones [1], [2]. The adjacent activity vector differs from the selected one by a single value only. The change of the active value allows obtaining the activity vector, which sets the opposite value on the output, whereas the change of the inactive value allows obtaining the new activity vector in some cases. The probability that the randomly generated stimuli will differ by a single value is small. Therefore, the generation of the adjacent stimuli to the selected ones allows to enrich the random search and to speed up the process of finding the new activity vectors [1], [2].

Additionally, the active values of the activity vectors can be considered as the terms of the logical function of the output and they could be used for the generation of the new input stimuli. In order to obtain the activity vector of the set $A1^j$ ($A0^j$) that has the most possible number of the active values, it needs to define as much as possible of the values of the activity vector of the opposite set $A0^j$ ($A1^j$) that has the single complemented active value only. This feature allows creating the different deterministic input stimuli generation methods that enrich ineffective random search.

4. Test generation approach

The number of the activity vectors selected for complex devices can be large; therefore, the number of the activity vectors has to be limited. Any imposed restrictions would lead to the loss of the information retained in the activity vectors. Nevertheless, the practical considerations of the used resources compel to apply the restrictions. Therefore, the minimum amount $min_i^j = S^j/S_i$ of the values V and N was calculated for the activity vectors of the sets $A0^j$ and $A1^j$, where S^j – the number of inputs related to the output j, S_i – the number of outputs related to the input i. The obtained value of the amount was rounded to the nearest integer value bigger than S^j/S_i. This ratio has to be kept larger than 1/3. The generated activity vector is not included to the set $A0^j$ or $A1^j$, if the number of the active values V or N is not increased on any of the inputs, which have the ratio of active values less than min_i^j. In such a way, the number of the activity vectors is controlled for the sets $A0^j$ and $A1^j$.

The number of the active values V and N for the activity vectors can be minimized. For every activity vector P^j from the set $A0^j$ ($A1^j$), the duplicating active values and the opposite active values are removed if the remaining active values do not contradict to the values of the activity vector P^j.

The simplest way to generate the pair of the delay test patterns is to use the activity vector as the second pattern of the pair, and the first pattern of the pair is obtained by changing single active value to the opposite one. For every activity vector, the number of test pattern pairs formed is equal to the number of the active values of the activity vector. The constructed test is SIT. But such a test is quite long; therefore, the minimization has to be applied.

The results of the experiment on the benchmark circuits ISCAS'85 are presented in Table 4. This table has two parts: before minimization and after minimization. The minimization was based on the minimizing active values V and N. We presented the following attributes: the number of the activity vectors (No AV), the number of the active values (No AS), the length of the SIT test (L), and the transition fault coverage (FC%). During the minimization, some activity vectors loose all the active values. Such activity vectors are removed from the further consideration. We see that after minimization the number of the activity vectors decreased twice, the number of the active values decreased 1.2 times, the length of the SIT test diminished more than 3 times, but the transition fault coverage degraded 0,4 per cent on the average, as well.

Table 4. SIT test

Circuit	Before minimization				After minimization			
	No AV	No AS	L	FC%	No AV	No AS	L	FC%
C432	1073		9288	99.78	800		2987	95.40
C499	4174		30903	94.40	1444		10197	94.40
C880	2370		22016	99.83	1479		5670	99.33
C1355	4147		30462	97.13	1997		10113	97.13
C1908	7721		115132	96.47	5178		43117	98.23
C2670	5195		40928	100	4453		29564	99.98
C3540	4095		46090	99.98	2711		8263	96.86
C5315	43909		508672	99.90	39144		190919	99.88
C6288	1408		20800	100	9593		10592	100
C7552	69881		932026	99.45	17565		253576	99.29
Average	14398		175632	98.70	7576		56500	98.30

We know from our experience that the complete coverage of the transition faults for some circuits can be obtained by using MIT tests only. The simplest way to obtain MIT test is to change all the active values to the opposite ones in first pattern of the pair. We name such a method as AVI. Let's consider the other possible ways of the MIT test construction.

The pair of test patterns formed from any of the activity vector from the set $A0^j$ ($A1^j$) and from any of the activity vector from the set $A1^j$ ($A0^j$) after change the active values *V* and *N* to the values 1 and 0 respectively detects transition faults. The pair of the test patterns formed from the activity vectors of the same set $A0^j$ or $A1^j$ detects also the transition faults but the signal glitch can be observed only. The four cases of the combination of the activity vectors can be considered;

- V1. For every activity vector from the set $A0^j$ ($A1^j$), the first vector of the pair is selected from the set $A1^j$ ($A0^j$) such a one, which has the most of the opposite values to the active values of the activity vector;
- V2. For every activity vector from the set $A0^j$ ($A1^j$), the first vector of the pair is selected from the set $A0^j$ ($A1^j$) such a one, which has the most of the opposite values to the active values of the activity vector;
- V3. For every activity vector from the set $A0^j$ ($A1^j$), the first vector of the pair is selected from the set $A1^j$ ($A0^j$) such a one, which has the least of the opposite values to the active values of the activity vector, but not less than 1;
- V4. For every activity vector from the set $A0^j$ ($A1^j$), the first vector of the pair is selected from the set $A0^j$ ($A1^j$) such a one, which has the least of the opposite values to the active values of the activity vector, but not less than 1;

Let's consider the circuit C17 in Figure.1. After consideration of all the possible input stimuli, we obtain the following sets of the activity vectors for the output *y*1:

$A0^{y1}$ = {< N1VV0>,<1NN10>, <NN100>}, and

$A1^{y1}$ = {< V0V10>,<0V1N1>, <0VN10>}; and

for the output *y*2:

$A0^{y2}$ = {< 1N01N>,<11VV1>}, and

$A1^{y2}$ = {< 101NV>,<10N1V>, <1V1N0>,<0VN10> }.

The test pattern pairs constructed in all four modes for the circuit C17 are shown in Table 5. The second pattern of all the pairs corresponds to the above enumerated activity vectors, when the active values *V* and *N* are changed to 1 and 0, respectively. The names of the columns show the mode of the construction of the first pattern of the pair. The second pattern of the pair is taken from the set $A0^{y1}$ in all the construction modes. The last line of the table shows the transition fault coverage for every mode of the construction. We see that the MIT test does not cover the transition faults completely, but it could be used in order to augment the SIT test.

Table 5. MIT test generation modes for C17

V1	V2	V3	V4
10110	01010	10010	00100
01110	01110	01110	01110
01101	10110	01110	01110
10010	10010	10010	10010
01010	10110	10010	01110
00100	00100	00100	00100

01110	10010	01101	01010
10110	10110	10110	10110
10010	01110	10110	01010
01101	01101	01101	01101
00100	01110	10110	10110
01010	01010	01010	01010
10101	10011	11111	11111
10010	10010	10010	10010
01010	10101	10010	10010
11111	11111	11111	11111
10010	11111	01010	10011
10101	10101	10101	10101
11111	10010	11100	10101
10011	10011	10011	10011
10010	11111	10011	10101
11100	11100	11100	11100
11111	10010	10101	10011
01010	01010	01010	01010
96%	92%	94%	80%

The introduced modes of the delay test construction were applied to the benchmark circuits ISCAS'85. The results are presented in Table 6. The activity vectors were not minimized, because the experimental investigation revealed that the minimization of the active values has no positive influence to the final results.

The analysis of the results in Table 6 reveals that the test length is equal to the number of the activity vectors, and the length on average is less about 4 times than the length of minimized SIT test (Table 4). The transition fault coverage on average of the mode V1 is higher almost 1 per cent than the transition fault coverage of the SIT test. The transition fault coverage on average for all the other modes is almost the same but less than in the mode V1.

Table 6. MIT test generation modes for the ISCAS'85

Circuit	No of pairs	V1 FC %	V2 FC %	V3 FC %	V4 FC %	AVI FC%
C432	1073	99.93	99.35	99.35	99.20	96.86
C499	4174	99.83	98.13	93.53	93.47	92.54
C880	2370	100	100	99.96	99.42	99.83
C1355	4147	99.91	95.19	94.39	92.33	93.31
C1908	7721	98.45	98.00	97.96	97.79	97.79
C2670	5195	99.89	96.48	99.98	96.42	96.71
C3540	4095	99.00	99.50	98.69	99.41	99.08
C5315	43909	100	100	100	100	99.95
C6288	1408	100	100	100	100	100
C7552	69881	99.14	99.13	98.73	98.77	96.99
Average	14398	99.60	98.58	98.25	97.68	97.31

The constructed test sets were combined together in order to achieve the higher transition fault coverage (Table 7). The names of the columns indicate the combined test sets. The combination of two test sets enlarges the length of the final test set twice, but the obtained fault coverage is higher only 0,18 per cent than in the mode V1. This result leads to the conclusion that the combination of two test sets is not reasonable. In order to obtain the higher transition fault coverage the restrictions should be relaxed on the number of the generated activity vectors.

Table 7. Combinations of MIT tests

Circuits	No of pairs	V1&V2 FC %	V1&V3 FC %	V1&V4 FC %	V1&AVI FC %	V2&V3 FC%
C432	2146	100	99.93	100	99.93	99.93
C499	8348	100	100	100	99.83	99.59
C880	4740	100	100	100	100	100
C1355	8294	100	99.94	100	99.94	97.85
C1908	15442	98.89	98.70	98.66	98.74	98.58
C2670	10390	99.95	100	100	100	99.98
C3540	8190	99.77	99.83	99.91	99.90	99.94
C5315	87818	100	100	100	100	100
C6288	2816	100	100	100	100	100
C7552	139762	99.18	99.23	99.24	99.26	99.24
Average	28795	99.78	99.76	99.78	99.75	99.51

5. Conclusion

The activity vectors obtained using the software prototype can be successfully used for the construction of the functional delay test. The presented experimental research revealed that the functional delay test can cover almost completely the transition faults.

Several test construction modes using the activity vectors were investigated. The best construction mode is when the first pattern of the pair is taken from the opposite set than the second one, and the first pattern of the pair has the largest number of the values opposite to the active values of the second pattern.

The further improvement of the quality of the functional test can be obtained relaxing the restrictions on the number of the generated activity vectors.

References

[1] **Bareisa E., Jusas V., Motiejunas K., Seinauskas R.** The Realization-Independent Testing Based on the Black Box Models, *Informatica*, Vilnius, Institute of Mathematics and Informatics, Vol. 16, No. 1, pp. 19-36, 2005.

[2] **Bareiša E., Jusas V., Motiejūnas K., Šeinauskas R**. *Functional Digital Systems Testing*, ISBN 9955-25-008-9, *Kaunas, Technologija*, 2006, p. 281.

[3] **Bareiša E., Jusas V., Motiejūnas K., Šeinauskas R**. Functional Delay Test Construction Approaches. *Elektronika ir elektrotechnika = Electronics and electrical engineering*. ISSN 1392-1215. *Kaunas, Technologija*, 2007, No. 2(74), pp. 49 - 54.

[4] **Buonanno G., Ferrandi F.. Ferrandi L.. Fummi F., Sciuto D.** How an "Evolving" Fault Model Improves the Behavioral Test Generation*, Proceedings of Great Lakes Symposium on VLSI*, pp. 124–130, 1997.

[5] **Cho C. H., Armstrong J. R.** B-algorithm: a Behavioral Test Generation Algorithm, *Proceedings of International Test Conference*, pp. 968–979, October 1994.

[6] **Chiusano S., Corno F., Prinetto P.** A Test Pattern Generation Algorithm Exploiting Behavioral Information, *Proceedings of Seventh Asian Test Symposium (ATS'98)*, Singapore, December 1998, pp. 480-485, 1998.

[7] **Corno F., Prinetto P., Sonza Reorda M.** Testability analysis and ATPG on behavioral RT-level VHDL, *Proceedings of IEEE International Test Conference*, pp.753-759, October 1997.

[8] **Ferrandi F., Fummi F., Sciuto D.** Test Generation and Testability Alternatives Exploration of Critical Algorithms for Embedded Applications, *IEEE Transactions on Computers*, Vol. 51, Issue 2, pp.200–215, 2002.

[9] **Kim H., Hayes J.P.** Realization-Independent ATPG for Designs with Unimplemented Blocks, *IEEE Trans. on CAD*, vol. 20, no. 2, pp. 290–306, 2001.

[10] **Michael M., Tragoudas S**. ATPG Tools for Delay Faults at the Functional Level. *ACM Transactions on Design Automation of Electronics Systems*, Vol. 7, No. 1, pp. 33-57, January 2002.

[11] **Pomeranz I., Reddy S. M**. On Testing Delay Faults in Macro-based Combinational Circuits, *Proc. Int. Conf. Computer-Aided Dsign*, San Jose, CA, 1994, pp. 332-339.

[12] **Rudnick E.M., Vietti R., Ellis A., Corno F., Prinetto P., Sonza Reorda M.** Fast Sequential Circuit Test Generation Using High-Level and Gate-Level Techniques, *Proceedings of IEEE Design, Automation and Test in Europe*, pp. 570-576, Feb. 1998.

[13] **Shao Y., Pomeranz I., Reddy S. M.**, On Generating High Quality Tests for Transition Faults, *Proceedings of the 11th Asian Test Symposium (ATS'02)*, pp.1-8, 2002.

[14] **Underwood B., Law W.O., Kang S., Konuk H**. Fastpath: A path-delay test generator for standard scan designs. *In Proceedings of 1994 International Test Conference (1994),* pp.154–163

[15] **Yi J., Hayes J. P.** A Fault Model for Function and Delay Testing, *Proc. of the IEEE European Test Workshop*, ETW'01, pp. 27-34, 2001.

[16] **Yi J., Hayes J. P.** The Coupling Model for Function and Delay Faults, *Journal of Electronic Testing: Theory and Applications*, vol. 21, No. 6, pp.631–649, 2005.

[17] **Yi J., Hayes J. P.** High-Level Delay Test Generation for Modular Circuits, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No. 3, March 2006, pp. 576-590.