KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Aidas Knistautas

ALGORITMŲ PRITAIKYMAS GENERUOJANT TESTUS KELIŲ EISMO TAISYKLIŲ TEORIJOS MOKYMUISI

Baigiamasis magistro projektas

Vadovas

Doc. dr. Sigitas Drąsutis

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

ALGORITMŲ PRITAIKYMAS GENERUOJANT TESTUS KELIŲ EISMO TAISYKLIŲ TEORIJOS MOKYMUISI

Baigiamasis magistro projektas

Nuotolinio mokymosi informacinės technologijos (kodas 621E14002)

Vadovas

(parašas) Doc. dr. Sigitas Drąsutis
(data)

Recenzentas

(parašas) Prof. Aleksandras Targamadzė
(data)

Projektą atliko

(parašas) Aidas Knistautas
(data)

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

Informatikos fakultetas

(Fakultetas)

Aidas Knistautas

(Studento vardas, pavardė)

Nuotolinio mokymosi informacinės technologijos (kodas 621E14002)

(Studijų programos pavadinimas, kodas)

„Algoritmų pritaikymas generuojant testus kelių eismo taisyklių teorijos mokymuisi“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ m. _____ ___ d.

Kaunas

Patvirtinu, kad mano, **Aido Knistauto**, baigiamasis projektas tema „Algoritmų pritaikymas generuojant testus kelių eismo taisyklių teorijos mokymuisi“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

Knistautas, Aidas. Algoritmų pritaikymas generuojant testus kelių eismo taisyklių teorijos mokymuisi. Magistro baigiamasis projektas / vadovas doc. dr. Sigitas Drąsutis; Kauno technologijos universitetas, Informatikos fakultetas.

Mokslo kryptis ir sritis: Technologijos mokslai, informatikos inžinerija - 07T

Reikšminiai žodžiai: *algoritmas, testas, kompiuterizuotas, Fisher - Yates, adaptyvus, Kelių eismo taisyklės.*

Kaunas, 2016. 52 p.

SANTRAUKA

Kiekviena vairavimo mokykla turi savo arba naudoja kitų gamintojų Kelių eismo taisyklių teorijos testų sprendimo sistemas, kurių pagalba galima mokytis vairavimo teorijos mokyklose arba, naudojant prisijungimo kodus, spręsti testus nuotoliniu būdu.

Viena pagrindinių tokių sistemų problema – klausimų pasikartojimas, kai testai pakartotinai sprendžiami kelias savaites. Klausimams kartojantis, mokiniai galiausiai mintinai įsimena klausimų atsakymus. Taip nukenčia mokinių žinios ir testų sprendimas neparodo mokinio žinių lygio. Atlikus rinkoje esamų Kelių eismo taisyklių testų sprendimo sistemų analizę, pasitvirtino, kad, sprendžiant testus pakartotinai, klausimai kartojasi. Vieni klausimai buvo pateikiami dažniau, nei kiti.

Išanalizavus, kokios testų rūšys gali būti pritaikytos optimizuojant testo klausimų pateikimą, buvo pasirinktas automatizuotas testo surinkimo tipas, nes, pritaikius optimizuotą algoritmą, galima sudaryti didelius skirtingų testų rinkinius. Egzaminavimo testams buvo pasirinktas kompiuterinio adaptyvaus testo tipas, nes yra pažangesnis, nei kiti ir parodo tikslius naudotojo gabumus.

Palyginus algoritmus, kurie gali būti panaudoti testų generavimui, sužinota, kad daugelis atsitiktinių skaičių generavimo algoritmų turi sisteminę paklaidą, dėl kurios vieni klausimai pateikiami dažniau, nei kiti. Išsiaiškinta, kad Fisher – Yates algoritmas neturi sisteminės paklaidos.

Realizavus 4 pasirinktus algoritmus, buvo atlikti greitaveikos testai ir palyginti jų rezultatai. Įrodyta, kad Fisher – Yates algoritmas neturi sisteminės paklaidos, nes visi klausimai pateikiami vienodu dažnumu. Adaptyvaus testo tipas buvo pritaikytas galutiniuose mokinio žinių įvertinimo testuose.

Knistautas, Aidas. The Adaptation of Algorithms for Generating Tests for Learning Theory of Road Traffic Regulations: Master‘s thesis in Information Technologies of Distance Education / supervisor assoc. Doc. dr. Sigitas Drąsutis. The Faculty of Informatics, Kaunas University of Technology.

Research area and field: Technological Sciences, Informatics Engineering - 07T

Key words: *algorithm, test, computerized, Fisher – Yates, adaptive, road traffic regulations.*

Kaunas, 2016. 52 p.

SUMMARY

Each driving school has its own or uses test solution systems for road traffic regulations created by other producers which helps to learn driving theory in schools or solve tests using access codes from distance

The main problem of these systems is repeated questions, when tests are solved several weeks. When the questions are repeated learners memorize the answers. Thus knowledge of learners gets worse and solution of the test do not show the real level of knowledge. The market analysis of test solutions systems of traffic regulation showed, that questions repeat when tests are solved repeatedly. Some questions were presented more often than others. It was analysed which type of the tests can be used to optimize presentation of test questions the automated test collection type was chosen, because various, vast collections can be created. For exam tests- computer adaptive test type was chosen, because of its progressiveness and real learners skills can be revealed.

It was found out by comparing the algorithms, which can be used for generating test, that most of random number generating algorithms have systematical error. Due to this error some questions are presented more often than others. It was found out, that Fisher-Yates algorithm do not have systematical error. Four algorithms were implemented. Speed tests were performed and results were compared. It was proved, that Fisher-Yates algorithm does not have systematical error, because all questions are presented in the same frequency. Adaptive test type was applied for final skill tests.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNAS

Alsket	Kelių eismo taisyklių testavimo sistema.
Autotestai	Kelių eismo taisyklių testavimo sistema.
K.E.T programa	Kelių eismo taisyklių testavimo sistema.
KET	Kelių eismo taisyklės.
Laravel 5	Internetinių aplikacijų kūrimo karkasas.
MVC	Programinės įrangos architektūros principas (angl. Model View Controller).
Blade	Internetinių puslapių šablonų kūrimo karkasas naudojamas „Laravel 5“
HTML5	Dokumentų žymėjimo kalba, skirta aprašyti interneto svetainės puslapio struktūrą (angl. Hyper Text Markup Language).
CSS	Kalba, skirta formatuoti ir keisti dokumentų žymėjimo kalba parašytų puslapių atvaizdavimą.
CSS3	CSS kalbos naujausias standartas.
JavaScript	Objektinė skriptų programavimo kalba.
PHP	Dinaminė interpretuojama programavimo kalba, pritaikyta svetainių kūrimui (angl. Hypertext Preprocessor).
„Mozilla Firefox“	Interneto naršyklė.
„Google Chrome“	Interneto naršyklė.
MySQL	Duomenų bazių valdymo sistema.
PDO	PHP sąsaja skirta pasiekti duomenų bazę (angl. PHP Data Objects).
Mcrypt	Biblioteka skirta informacijos kodavimui.
Apache	HTTP serveris.
HTTP	Protokolas skirtas perduoti informaciją internetu (angl. Hyper Text Transfer Protocol).
UML	Specifikacijų kūrimo ir modeliavimo kalba (angl. Unified Modeling Language).
ATA	Kompiuterinio testavimo algoritmo tipas (angl. Automated Test Assembly).
CAT	Kompiuterinio testavimo algoritmo tipas (angl. Computerized Adaptive Test).
LOFT	Kompiuterinio testavimo algoritmo tipas (angl. Linear On the Fly Test).
CCT	Kompiuterinio testavimo algoritmo tipas (angl. Computerized Classification Test).
CFT	Kompiuterinio testavimo algoritmo tipas (angl. Computerized Fixed Test).

ĮVADAS

Projektas „Algoritmų pritaikymas generuojant testus Kelių eismo taisyklų teorijos mokymuisi“ yra baigiamasis magistro projektas.

Darbo aktualumas ir problematika

Kiekviena vairavimo mokykla turi ar naudoja kitų gamintojų Kelių eismo taisyklių (toliau KET) teorijos testų sistemas, kurioms padedant galima mokytis spręsti testus vairavimo mokyklose arba naudojant prisijungimo kodus spręsti testus internetu.

Sistemos būna mokamos, tačiau dažna tokių sistemų problema – klausimų pasikartojimas, sprendžiant testus ilgą laiką. Mokytis vairavimo teorijos mokykloje trunka nuo kelių savaičių iki kelių mėnesių, todėl, sprendžiant testus, labai daug klausimų kartojasi, kol mokiniai galiausiai mintinai įsimena klausimų atsakymus. Taip nukenčia mokinių žinios ir testų sprendimas neparodo mokinio žinių lygio.

Darbo tikslas ir uždaviniai

Pagrindinis projekto tikslas – optimizuoti klausimų pateikimą interaktyvioje KET testų sprendimo sistemoje pritaikant pažangesnius maišos algoritmus bei panaudojant darbo rezultatus pakeisti mokyklos testų sprendimo sistemą nauja nemokama sistema.

Pagrindiniai projekto uždaviniai:

1. Atlikti rinkoje esamų KET testų sprendimo sistemų, funkcionalumo analizę ir jas palyginti;
2. Išanalizuoti galimas technologijas, kurias būtų galima pritaikyti kuriamai testavimo sistemai;
3. Išanalizuoti testo klausimyno sudarymo algoritmus ir palyginti jų rezultatus bei greitaveiką.
4. Suprojektuoti ir realizuoti sistemą pagal reikalavimų specifikaciją ir realizuoti algoritmus klausimyno sudarymui;
5. Atlikti visus reikiamus sistemos funkcionalumo testus, sudarius įvairias galimas situacijas su skirtingais duomenimis.

Darbo struktūra

Magistro baigiamąjį darbą „Algoritmų pritaikymas generuojant testus Kelių eismo taisyklių teorijos mokymuisi“ sudaro įvadas, 4 pagrindiniai skyriai, išvados, literatūros sąrašas, terminų ir santrumpų žodynas. Pagrindinę darbo dalį sudaro – 40 puslapių. Pagrindinėje dalyje pateikti 26 paveikslai ir 3 lentelės.

Pirmame skyriuje aprašoma analitinė dalis, kurioje apžvelgtos testavimo sistemos ir algoritmai.

Antrame skyriuje - projektinės dalies aprašymas.

Trečiame skyriuje aprašoma, kaip yra realizuoti algoritmai.

Ketvirtame skyriuje pateikiami algoritmų testų rezultatai.

Penktame skyriuje apibendrinamas atliktas darbas, pateikiamos išvados.

Šeštame skyriuje pateikiamas naudotos literatūros sąrašas.

1. ANALITINĖ DALIS

1.1. KET testų sprendimo sistemų analizė

Testuojant ir besimokant vis dažniau pasitelkiami kompiuteriniai testavimo būdai: apklausos, testai, viktorinos. Tai efektyvus būdas tikrinti besimokančiųjų žinias. Tokie testavimo būdai ne tik taupo laiką ir resursus, bet suteikia galimybę rinkti statistinius duomenis ir analizuoti rezultatus. Kompiuterinis testavimas suteikia galimybę rengti nuotolinio mokymo kursus ar galimybę spręsti mokomuosius testus internetu [1].Ne išimtis ir KET testai. Paieškojus šių testų internete randama begalė rezultatų. Analizavimui pasirinktos 3 KET testų sprendimo sistemos.

1.1.1. K.E.T programa

1.1 pav. KET sistemos prisijungimo langas

Tai sistema skirta spręsti KET testus. Norint naudotis pilna sistemos versija, reikia nusipirkti norimą valandų kiekį. Kiekvienam naudotojui, nusipirkusiam atitinkamą valandų kiekį, suteikiamas kodas, kuriuo galima prisijungti sistemoje. Pavyko rasti ir nemokamą sistemos versiją (http://www.ketprograma.lt/) [2] norintiems ją išbandyti. Bandomojoje versijoje nemokamai galima spręsti testus ir matyti jų rezultatus.

Privalumai:

- veikia greitai;
- naudotojo sąsaja paprasta naudotis;
- testai paruošti dvejomis skirtingomis kalbomis;
- naudotojo sąsaja pritaikyta mažesniam ekrano dydžiui.

Trūkumai:

- nėra bendros informacijos apie sistemą;
- nusipirkti testo sprendimo valandas galima tik keliuose vairavimo mokyklų puslapiuose;
- nėra informacijos apie testų atnaujinimą;
- klausimų pasikartojimas;
- nėra pilno tekstinio paaiškinimo apie KET punktą, kai klausimas yra atsakomas neteisingai.

1.1.2. Autotestai

1.2 pav. Autotestai klausimo pateikimo langas

Buvo rasta ir išbandyta sistema „Autotestai“. Norint naudotis pilna sistemos versija taip pat reikalingas prisijungimo kodas, kurį galima įsigyti (http://www.autotestai.lt) internetinėje svetainėje [3].

Privalumai:

- vartotojo sąsaja patraukli;
- pritaikyta prie įvairaus dydžio monitorių bei telefonų;
- pateikiama informacija apie testų atnaujinimą;
- galima pasirinkti norimos kategorijos testus.

Trūkumai:

- klausimų pasikartojimas.

1.1.3. Alsket

1.3 pav. Alsket sistemos prisijungimo langas

Sistema buvo išbandyta naudojant prisijungimo raktą. Ją galima rasti adresu (http://alsket.lt/) [4]. Prisijungus buvo galima spręsti KET testus, tačiau daugelis funkcijų buvo negalimos. Norint naudotis visomis sistemos funkcijomis, reikalingas registruotas naudotojas.

Privalumai:

- pateikiama informacija apie testų atnaujinimą;
- nuolat atnaujinama;
- vartotojo sąsaja dvejomis kalbomis;
- galimybė pasirinkti norimą kategoriją;
- testų metu sistema veikia labai greitai.

Trūkumai:

- nėra bandomosios versijos;
- vartotojo sąsajos išvaizda nepatraukli;
- keletas funkcijų užrakintos neregistruotam vartotojui;
- klausimų pasikartojimas.

Testavimo sistemos buvo išbandytos pasinaudojant pateikiamomis bandomosiomis versijomis, todėl duomenys gali neatitikti tikrosios versijos.

Lyginant sistemas, buvo atsižvelgta į keletą pasirinktų pagrindinių kriterijų. Visos sistemos buvo lyginamos iš eilės. Buvo bandomos demonstracinės testavimo sistemų internetinės versijos. Sistemų lyginimo lentelė pateikta žemiau (1.1 lentelė).

Vartotojo sąsaja – buvo žiūrėta, kaip atvaizduojama naudotojo sąsaja kompiuteryje, planšetiniame kompiuteryje ir telefone. Atkreiptas dėmesys į sąsajos patogumą, šriftą, mygtukų išdėstymą.

Klausimų pateikimo algoritmas – buvo ieškoma informacijos, kokiu būdu pateikiami klausimai. Tačiau nei vienoje sistemoje nebuvo nurodytas algoritmas, pagal kurį pateikiami klausimai.

Klaidų analizė ir teorijos pagrindimas – pažiūrima kaip sistema pateikia rezultatus naudotojui. Atsižvelgta į testo atsakymų analizę. Buvo žiūrima, ar, išsprendus testą, parodomi klaidingi ir teisingi atsakymai. Tikimasi, kad klaidingo atsakymo atveju sistema nurodys KET teorijos punktą, pagal kurį buvo sudarytas klausimas.

Pilnas funkcionalumas – tikrinta, kokios funkcijos suteikiamos naudotojui. Žiūrima, ar naudotojui suteikiamos papildomos funkcijos, jei užsiregistruoja sistemoje arba nusiperka testo sprendimo laiko.

Informacija apie klausimų atnaujinimą – tikrinama, kada paskutinį kartą buvo atnaujinta klausimų bazė pagal naujausius Kelių eismo taisyklių pasikeitimus.

Bendras klausimų skaičius – kiekviena testų sistema turi savo klausimų bazę. Buvo žiūrima, koks kiekvienos iš analizuojamų sistemų pateikiamas dydis. Kuo didesnis klausimynas, tuo patikimesnis testavimas, nes klausimai dažnai nesikartoja.

1.1 lentelė. KET sistemų palyginimas

Kriterijai	Alsket	K.E.T programa	Autotestai
Vartotojo sąsaja	Ne visiškai pritaikyta prie mažesnio ekrano dydžio įrenginių. Mažesniame telefono vaizduoklyje sistemos elementai pasislepia.	Pritaikyta atvaizduoti tiek kompiuteryje, tiek telefone.	Pritaikyta atvaizduoti tiek kompiuteryje, tiek telefone.
Greitaveika	Pirmas klausimas pateikiamas mažiau nei per sekundę.	Pirmas klausimas pateikiamas mažiau nei per sekundę.	Pirmas klausimas pateikiamas mažiau nei per sekundę.

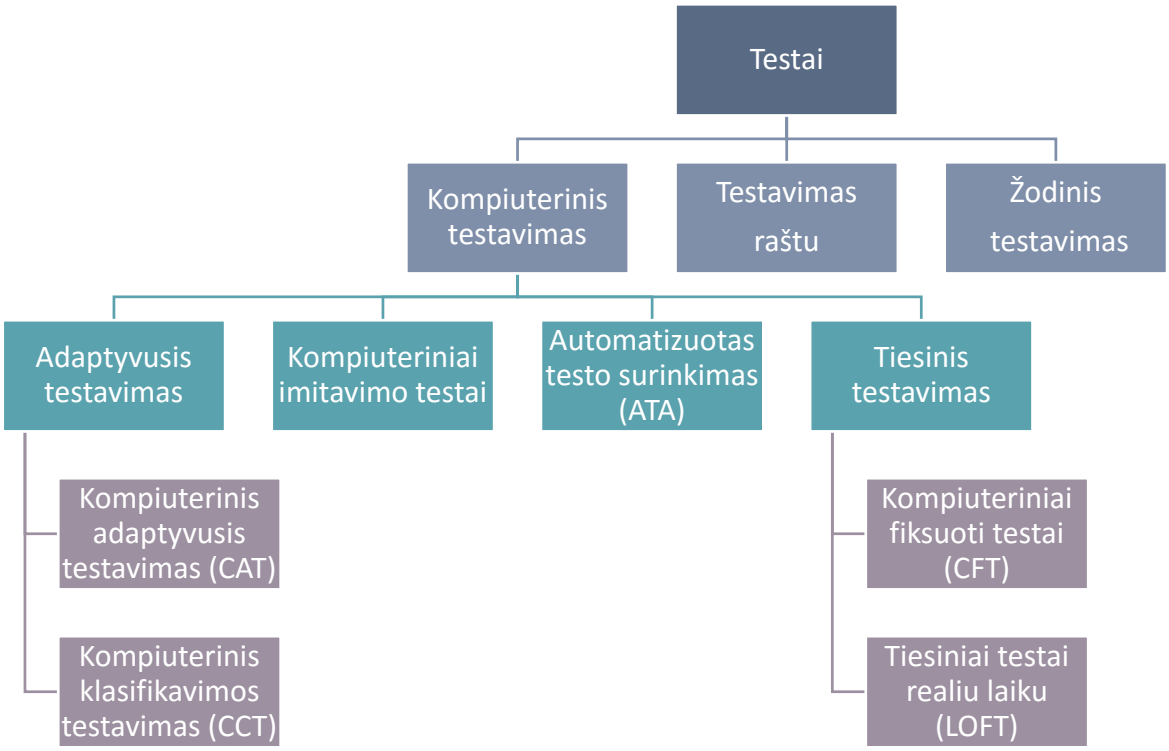
Klausimų pateikimo algoritmas	Nežinomas	Nežinomas	Nežinomas
Klaidų analizė ir teorijos pagrindimas	Parodomos klaidos ir atitinkamas teorijos straipsnis.	Nurodomi tik KET punktų numeriai ir teisingi bei neteisingi atsakymai.	Parodomos klaidos ir atitinkamas teorijos punktas.
Pilnas funkcionalumas	Reikalingas mokyklos suteikiamas raktas arba registruotas	Reikalingas mokamas testo valandų skaičius	Reikalingas mokamas testo valandų pirkimas
Informacija apie klausimų atnaujinimą	yra	yra	yra
Bendras klausimų skaičius	~2000	nenurodytas	1500

1.2. Testai ir jų sudarymo algoritmai

Dėl savo universalumo, testuojant besimokančiuosius, vis dažniau yra pasitelkiami kompiuteriai. Ženkliai sutaupomas tiek dėstytojų, tiek studentų darbo laikas, nes testai automatiškai patikrinami. Naudojant testavimui kompiuterius, taupomas popierius. Taip tausojami gamtos resursai.

Tobulėjant testavimo sistemoms, keitėsi klausimų pateikimas. Pradėta naudoti vaizdinę medžiagą. Tobulėjo ir plėtėsi klausimų tipai nuo „taip“ – „ne“ iki žodžių sujungimo ar net atviro klausimo. Nors testai tobulėjo, tačiau testavimo principas išliko toks pats. Pateikiamas statinis atsitiktinai sugeneruotų klausimų skaičiaus. Testuojamasis turi atsakyti į visus klausimus, norint surinkti maksimalų teisingų atsakymų skaičių. Testai išlaikomi arba neišlaikomi, surinkus atitinkamą teisingų ir neteisingų atsakymų santykį. Toks testavimo metodas naudojamas iki šiol. KET testui atlikti dažniausiai skiriamas fiksuotas laiko kiekis. Sprendžiant KET testus, vienam klausimui atsakyti suteikiama 1 minutė. Todėl visam testui, kurį sudaro 30-40 klausimų, išspręsti skiriama 30-40 minučių laiko. Tai nemažas laiko tarpas. Esant dideliam testuojamųjų kiekiui, testavimo procesas gali trukti net kelias ar keliolika valandų.

Testavimo sistemose dažnai pasirenkamas algoritmas, atitinkantis tam tikrus tos srities reikalavimus. Sukurta įvairiausių testavimo algoritmų. Pagrindiniais išskiriami tik keli, kurie gali būti suskirstyti taip, kaip parodyta paveiksle (žr. 1.4 pav.):



1.4 pav. Pagrindinės testų rūšys

Viena iš pagrindinių KET testavimo sistemų ypatybių yra ta, jog, sprendžiant testus, pakartotinai vis dažniau pasitaiko klausimų, kurie jau buvo spręsti. Todėl, norint sukurti geresnę ir efektyvesnę mokymosi priemonę būsimiems vairuotojams mokyti, buvo nuspręsta pritaikyti vieną iš galimų klausimyno sudarymo algoritmų. Algoritmas turi atitikti šiuos kriterijus:

- gebėti generuoti skaičių aibes;
- sudarytoje skaičių aibėje skaičiai negali kartotis;
- algoritmas turi būti pakankamai greitas, kad sudarytų aibę iš 30 – 40 skaitmenų;
- kaskart, sudarant naują aibę, skaičiai negali kartotis prieš tai buvusioje;
- išsekus klausimų aibei, klausimai kartojasi kuo mažesniu dažniu;
- galimybė pritaikyti testo sudėtingumą individualiai kiekvienam testuojamajam.

Vienas iš klausimyno sudarymo kriterijų bei privalumų yra tai, kad klausimyne turi būti pateikti įvairaus pobūdžio klausimai. Dažniausiai pateikiami klausimai iš bendrų KET taisyklių ir keletas klausimų iš kategorijos, kurią norima laikyti VĮ „Regitra“. Vidutiniškai pateikiami [5]:

- 8 klausimai apie eismą sankryžose;
- 2 klausimai apie važiavimo greitį;

- 2 klausimai apie sankryžas su reguliuotoju;
- 2 klausimai apie kelio ženklus ir jų galiojimo zonas;
- 1 klausimas apie žiedines sankryžas;
- 1 klausimas apie krovinio gabenimą;
- 1 klausimas apie važiavimą gyvenamojoje zonoje;
- 1 klausimas apie manevravimą kelyje;
- 1 klausimas apie saugų vairavimą.

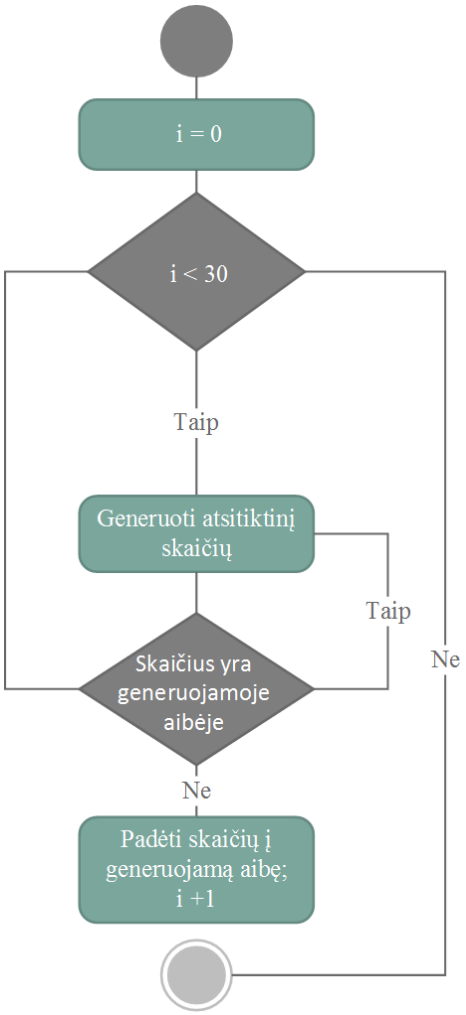
1.2.1. Aibės generavimas tikrinant

Vienas iš paprastų būdų sugeneruoti unikalių skaičių aibę yra matomas žemiau pateiktoje schemoje (žr. 1.5 pav.). Pavyzdžiui, parodyta, kai generuojamas B kategorijos KET testas iš 30 unikalių klausimų. Kiekvienas sugeneruotos testo aibės skaičius atitinka vieną klausimą iš visos klausimų aibės.

Atlikus esamų sistemų analizę, išsiaiškinta, kad vidutiniškai sistema turi nuo 1500 iki 2000 ir daugiau klausimų duomenų bazę.

Tai vienas iš būdų sugeneruoti unikalią skaičių aibę. Tačiau tai nėra efektyvus būdas aibei sudaryti. Viena iš priežasčių yra tai, kad, jei skaičius yra aibėje ir naujai sugeneruotas skaičius atitiks prieš tai sugeneruotą, algoritmas gali iš naujo generuoti kitą skaičių. Taip papildomai bus prarastas laikas skaičių generacijai. Taip pat neišsprendžiama problema dėl sugeneruotos aibės unikalumo nei prieš tai buvusiose.

Generuojant aibes ir tikrinant jas su prieš tai buvusiomis, prarandama daug laiko. Toks generavimo būdas, prieš pradedant testą, nėra priimtinas sistemos greitaveikai. Todėl viena iš galimybių yra iš anksto sugeneruoti visas galimas unikalias testo aibes. Tai padėtų pagreitinti testavimą, nes tada reikėtų tik paimti vieną iš jau sugeneruotų aibių ir pateikti besimokančiajam. Vidutiniškai paėmus klausimų duomenų bazės dydį a = 1650, unikalių testo klausimų skaičių t = 30, gauname, kad 1650/ 30 = 55 unikalius testus. Vieno testo sprendimo laikas atitinka klausimų skaičių. Suskaičiavus gauname 27,5 testų sprendimo valandų.



1.5 pav. Aibės generavimas atsitiktinio skaičiaus generavimo metodu

1.2.2. Aibės generavimas paprastuoju maišymo būdu

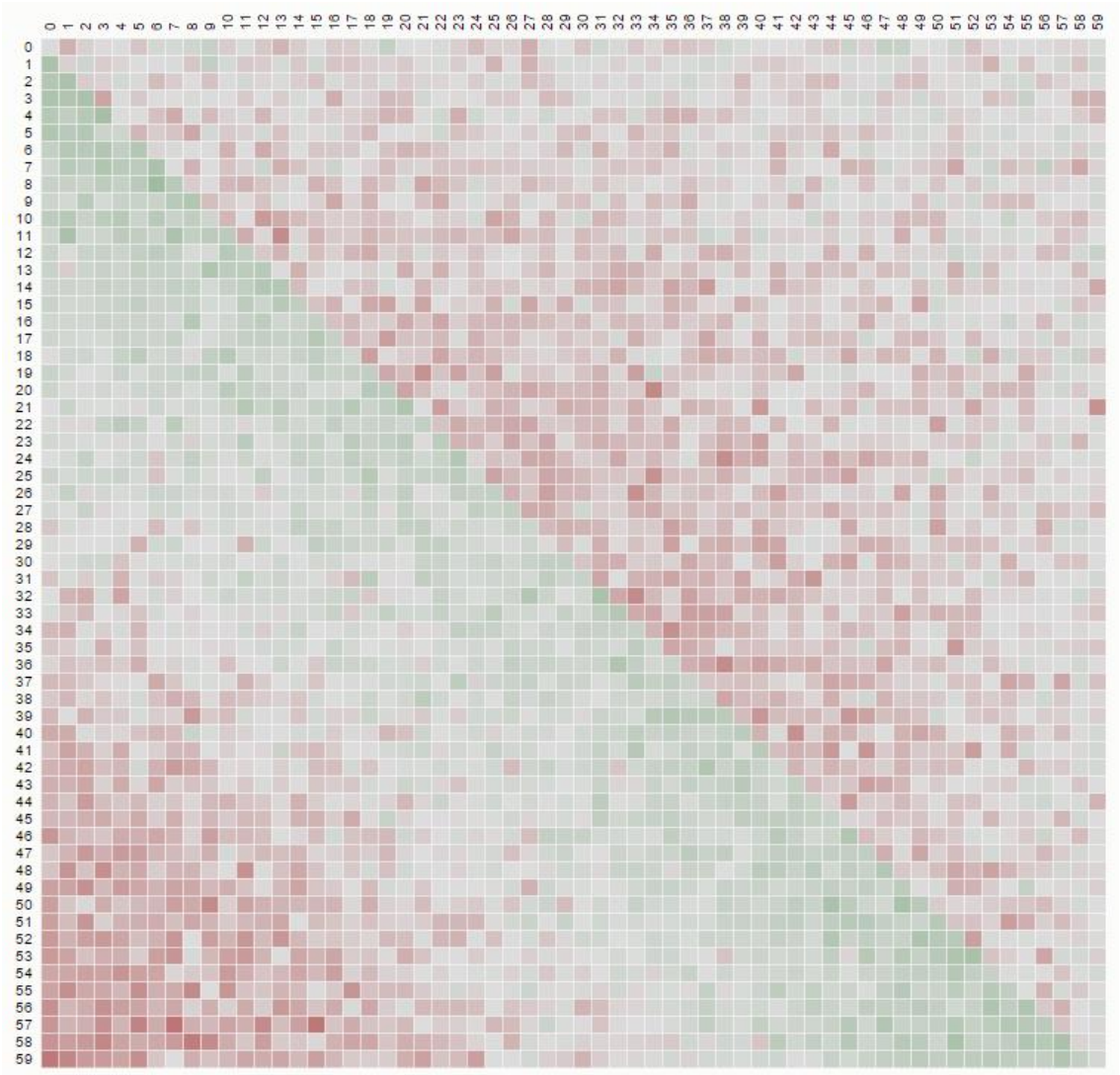
Algoritmas veikia, eidamas per skaičių aibę ir kiekvieną elementą sukeičia atsitiktiniu tos pačios aibės elementu. Vėliau iš sumaišytos aibės paimamas reikalingas elementų kiekis nuo pradžios. Nors algoritmo veikimo principas gana paprastas, tačiau, išanalizavus pateikiamus rezultatus, paaiškėjo, jog algoritmas turi sisteminę paklaidą [6]. Algoritmas netolygiai išmaišo pateiktą aibę.

Tyrime pasinaudota 2012 metais Miko Bostocko atliktu darbu [7]. Savo darbe jis vizualiai realizavo ir palygino kelis algoritmus. Jo pateiktose vizualizacijose galima pasirinkti jau įgyvendintus aibės maišymo algoritmus arba sukurti savo ir išbandyti sistemoje. Algoritmų veikimai vaizduojami 59 langelių aukščio ir 59 langelių pločio matricoje.

Žemiau pateikiama algoritmo vizualizacija matricoje (žr. 1.6 pav.). Iš pateiktų rezultatų matome, jog algoritmas vienus elementus pasirenka dažniau nei kitus. Elemento sisteminė paklaida žymima spalvų gama nuo žalios iki raudonos. Kuo langelio spalva raudonesnė, tuo didesnė neigiama sisteminė paklaida, kuo langelis žalesnis, tuo didesnė teigiama paklaida. Kaip matome, atspalviai keičiasi

gradientu nuo neigiamos link teigiamos reikšmės. Tai patvirtina, kad algoritmas turi sisteminę paklaidą.

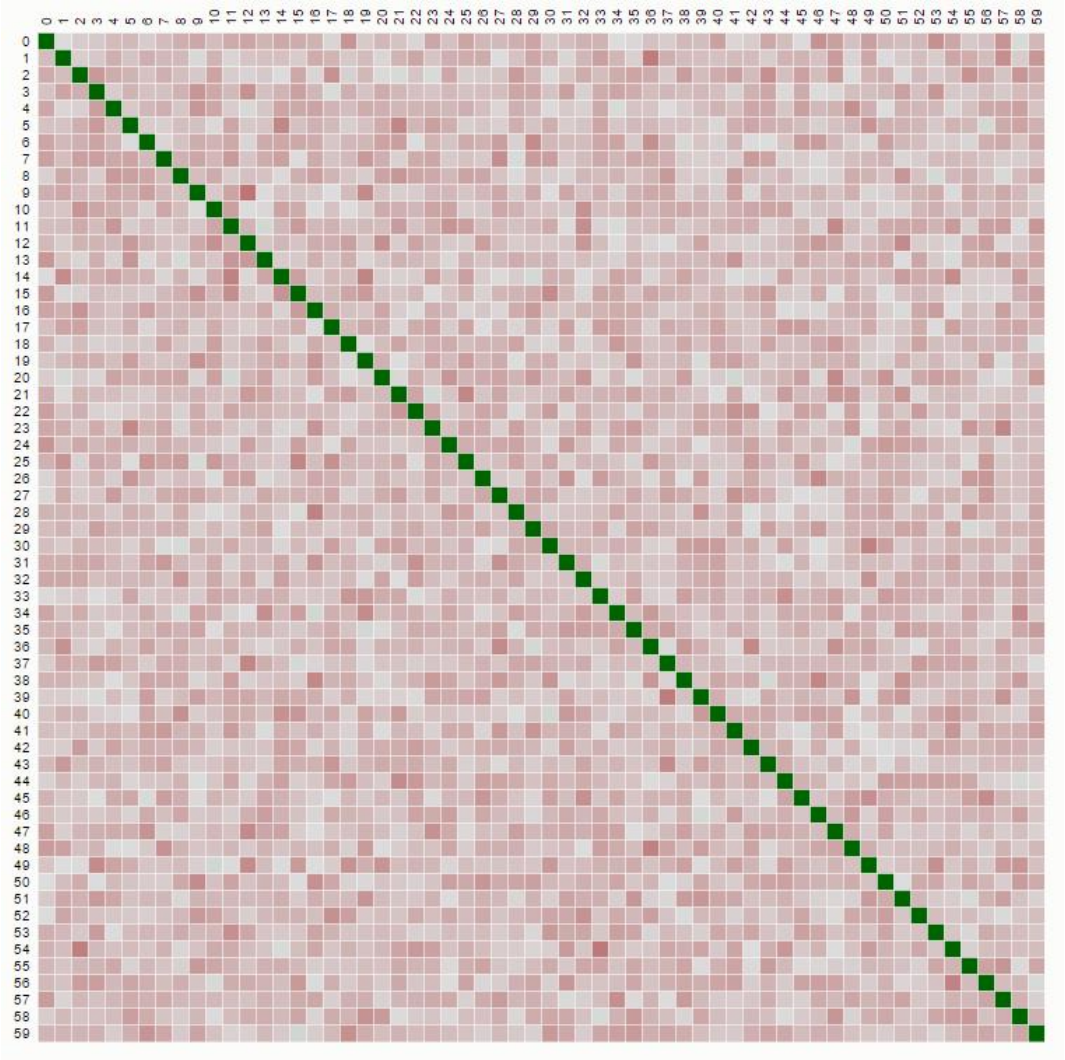
Toliau buvo pastebėta, kad algoritmas visais atvejais eina per visą elementų aibę. Sukeisdamas elementą su atsitiktiniu, algoritmas gali atsitiktinai grąžinti elementus į savo vietas. Taip pat šis algoritmas neefektyvus, nes sukeičia jau sukeistus aibės narius. Atsitiktiniam sukeitimui algoritmas naudota visą turimą aibę. Tokiu būdu laiko ir resursų sąnaudos išauga atitinkamai nuo turimos aibės dydžio.



1.6 pav. Paprastas aibės maišymas pakeičiant kiekvieną aibės elementą atsitiktiniu kitu elementu [7]

Kita šio algoritmo atmaina yra, kai einama per visą turimą elementų aibę ir sukeičiami atsitiktiniai aibės elementai su atsitiktiniu kitu tos pačios aibės elementu. Vizualizuotą algoritmo veikimą matome paveiksle (žr. 1.7 pav.).

Išanalizavus rezultatą matome, kad algoritmas beveik tolygiai išrenka atsitiktinius elementus. Tačiau gan ryškus raudonas langelių atspalvis leidžia teigti, kad elementai atsitiktinai yra išrenkami keletą kartų. Šis algoritmas taip pat veikia gana lėtai, nes einama per visus aibės elementus ir atsitiktinai išrenkamas elementas taip pat yra iš visos turimos aibės.

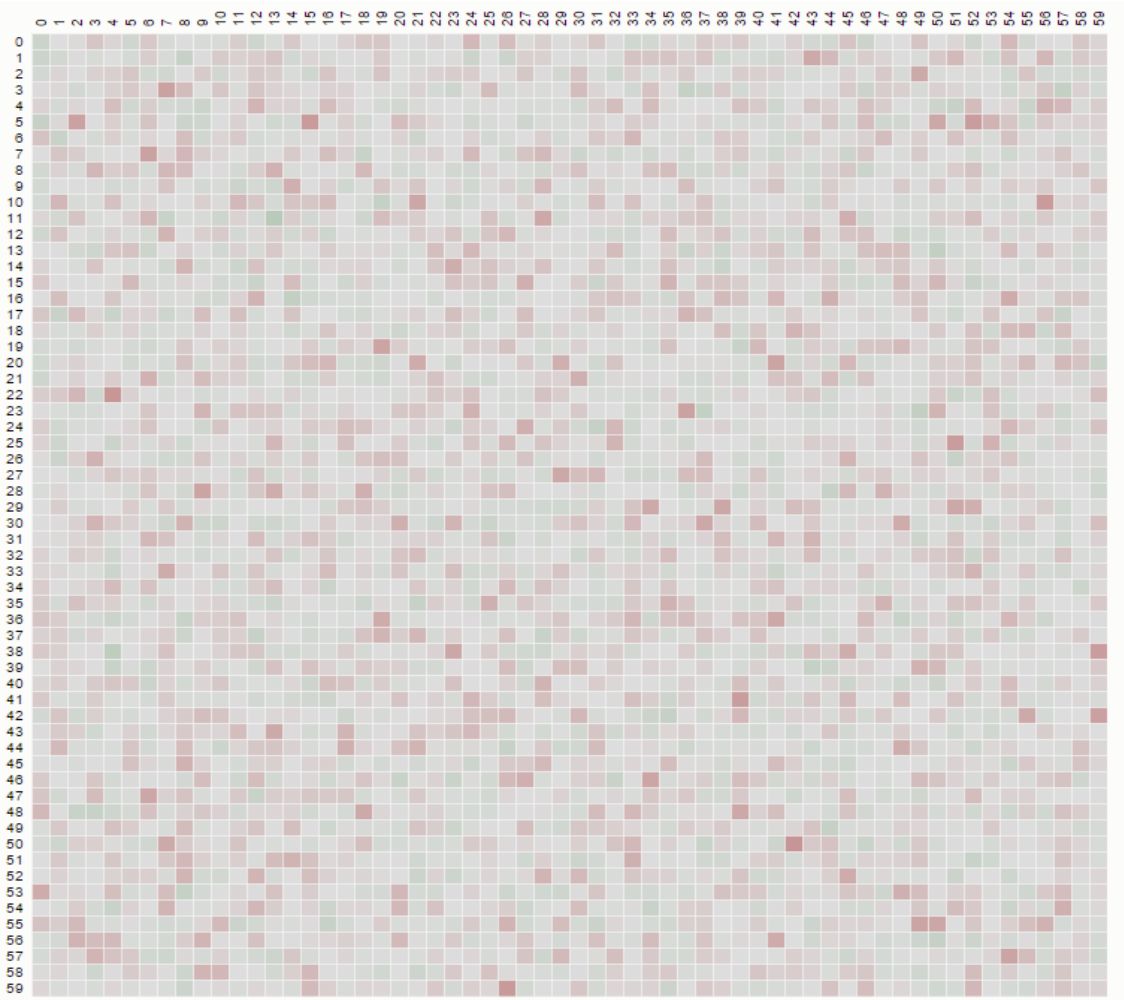


1.7 pav. Einama per visus aibės elementus ir sukeičiamas atsitiktinis aibės elementas su kitu atsitiktiniu aibės elementu [7]

1.2.3. Aibės generavimas Fisher - Yates (Knuth) algoritmu

Daug greitesnis ir efektyvesnis aibės generavimo algoritmas yra Fisher - Yates. Pavadinimas kilęs iš jį sugalvojusių ir praktiškai pritaikiusių mokslininkų pavardžių [8]. Algoritmas išpopuliarėjo, kai buvo pradėtas naudoti kompiuterijoje.

Algoritmas veikia iteraciniu metodu. Iš eilės einama per visą elementų aibę ir esamas elementas sukeičiamas su atsitiktiniu tos pačios aibės elementu. Nors šis algoritmas beveik nesiskiria nuo paprasto prieš tai analizuoto algoritmo, tačiau turi vieną esminį privalumą – po kiekvienos iteracijos galimų atsitiktinių narių aibė sumažinama vienetu. Taip po kiekvienos iteracijos algoritmas turi vis mažesnę galimybių aibę. Todėl veikia greičiau, nei prieš tai aprašyti algoritmai. Iš žemiau pateikto algoritmo vizualizavimo paveikslo matome, kad dalies langelių atspalvis rausvas, matomi žalsvi langeliai parodo, kad skaičius buvo išrinktas vos keletą kartų. Matematiškai įrodyta, kad šis algoritmas neturi sisteminių paklaidų.



1.8 pav. Fisher-Yates algoritmo vizualizavimas [7]

1.2.4. Kompiuteriniai adaptyvūs testai (angl. Computerized Adaptive Tests)

Adaptyvių testų tikslas yra maksimaliai išnaudoti testo galimybes, nustatant, ar testuojamasis išlaiko testą, pateikiant kuo mažiau klausimų. Metodo veikimas paprastas. Testuojamajam, jei apie jo gebėjimus nėra žinoma, pateikiamas vidutinio ar lengvo sudėtingumo klausimas. Jei testuojamasis atsako į klausimą gerai – jam pateikiami sunkesni klausimai. Jei klausimas atsakytas neteisingai – pateikiami lengvesni klausimai. Metodas naudoja IRT (Item Response Theory) [9], kuris, pagal atitinkamus besimokančiojo ir klausimo parametrus, parenka sekantį klausimą. Testuojamojo

parametras – žinių lygis. Nei karto neatlikus testo, nustatomas vidutinis lygis. Klausimo parametrai: klausimo sunkumas, didžiausia galima paklaida, atspėjimo tikimybė, jei klausimas turi kelis pasirenkamus variantus. Žemiau pateikiama IRT formulė, pagal kurią paskaičiuojama teisingo atsakymo galimybė:

$$p_i(\theta) = c_i + \frac{1 - c_i}{1 + e^{-a_i(\theta - b_i)}}; \quad (1)$$

Tikimybė, kad testuojamasis atsakys į pateiktą klausimą, kurio gebėjimus žymi ženklas θ, žymima $p_i(\theta)$. Formulėje klausimo sudėtingumas žymimas b. Klausimo diskriminantas, kuris parodo, kiek tikslus yra klausimo sudėtingumas, žymimas a. Ženklas c žymi klausimo atsitiktinį spėjimą [1].

Metodo sudedamosios dalys:

- ištestuota ir sukalibruota klausimų bazė;
- pradinis testuojamojo lygis;
- algoritmas, kuris parenka klausimus;
- testuojamojo įvertinimo algoritmas;
- testo pabaigos sąlyga arba riba, kai testas yra išlaikomas arba ne.

Pagrindiniai CAT metodo privalumai:

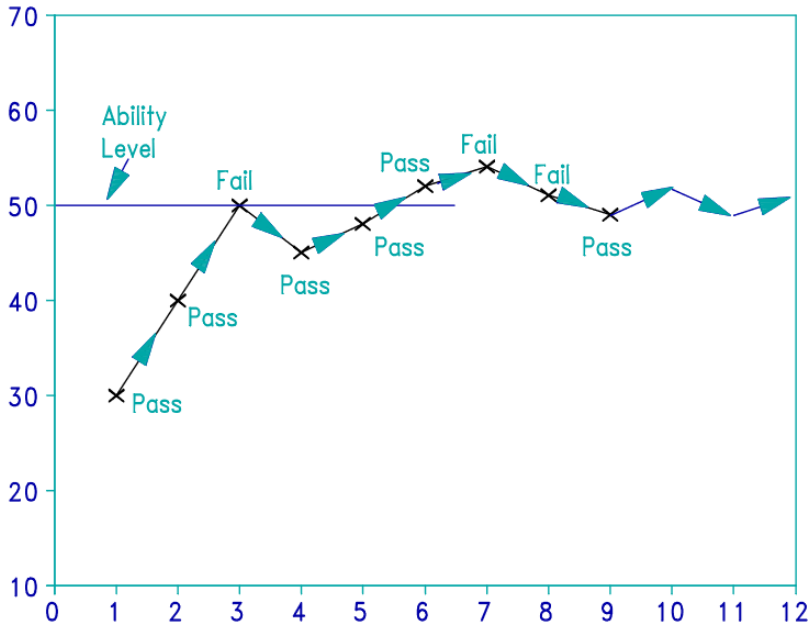
1. Panaikinami nereikšmingi klausimai.
   Klausimai, kurie naudotojui yra per lengvi arba per sunkūs, nebus pateikiami. Taip pat testuojamieji negali atsakinėti į klausimus pagal iš anksto paruoštus atsakymo variantus. Atsakymų spėliojimas taip pat yra neefektyvus, sprendžiant tokio tipo testus.
2. Sutrumpinamas testo ilgis.
   Adaptyvūs testai yra trumpesni, tačiau nedaug. Norint testuoti mokinius dideliu tikslumu, didėja ir bendras klausimų skaičius. Optimaliausiu atveju testas stabdomas, kai pasiekiama testo išlaikymo riba. Kartais, parinkus tinkamus testo nustatymus, adaptyvūs testai testo ilgį sumažina net 50% [10].
3. Saugesnis testas.
   Tokio tipo testai pateikia po vieną klausimą. Kaskart gali būti pateiktas vis kitas klausimas priklausomai nuo atsakymo, todėl testas nesuteikia galimybės nusirašyti [11].

Metodo trūkumai:

- testo sistemos paruošimas reikalauja daug laiko;
- didesni sistemos sukūrimo kaštai;
- klausimyno sudarymas ir paruošimas yra ilgas ir sunkus procesas;

- klausimynas turi būti išbandomas keletą ar kelis tūkstančius kartų;
- būtini pilotiniai sprendimai klausimų kalibravimui;
- testai supanašėja – tokie patys klausimai panašiai atsakiusiems;
- nėra galimybės grįžti į ankstesnį klausimą.

Algoritmo veikimo diagrama parodyta paveiksle (žr. 1.9 pav.) Diagramos apatinėje (X) ašyje pažymėtas klausimo numeris. Kairėje (Y) ašyje pažymėtas klausimo sudėtingumas. Kaip matome iš diagramos, su kiekvienu teisingai atsakytu klausimu, sekantis klausimas sudėtingėja.



1.9 pav. Adaptyvaus testo veikimo diagrama [11]

Taip klausimai yra sunkinami, kol klausimas atsakomas neteisingai. Toliau pateikiamas lengvesnis klausimas, tačiau sunkesnis, nei prieš tai atsakyti klausimai. Algoritmas, parinkdamas sekantį klausimą, vis mažina galimo sudėtingumo rėžius kol bus tenkinama viena iš testo pabaigos sąlygų.

Testo pabaigos sąlygos:

- Nėra galimybės pateikti tokio sudėtingumo klausimo;
- Naudotojas atsakė į didžiausią leistiną testo klausimų skaičių;
- Naudotojo rezultato tikslumas pakankamai tikslus, kad būtų baigtas testas;
- Baigėsi testui skirtas laikas.

1.2.5. Kompiuteriniai klasifikavimo testai (angl. Computerized Classification Tests)

Klasifikavimo testai naudojami ne įvertinti testuojamųjų žinias, o juos suskirstyti į grupes. Dažniausiai šis testo tipas taikomas, grupuojant į išlaikiusiųjų ir neišlaikiusiųjų egzaminuojamųjų grupes. Šių testų pagalba galima suskirstyti į daugiau kaip dvi grupes. Šio tipo testuose naudojama IRT teorija. Todėl šį testo tipą galime priskirti prie adaptyvių testų. Klasifikavimo testai dažniausiai

būna kintamo ilgio. Tai viena iš adaptyvių testų gerųjų savybių. Testo įvertinimas skaičiuojamas testo pabaigoje. Įvertinama pagal skalę.

Šiems testams naudojama vidutinio dydžio klausimų bazė. Tačiau, jei naudojama IRT teorija, patartina kuo didesnė klausimų bazė, nes taip užtikrinamas testo tikslumas ir klausimai nesikartoja. Tokio tipo testai priskiriami prie didesnio saugumo.

Kompiuteriniai klasifikavimo testai reikalauja vidutinių resursų ir laiko sąnaudų, nes klausimų bazė yra vidutinio dydžio. Jei naudojama IRT teorija, reikalingas klausimų kalibravimas.

Testuojamos grupės dydis vidutinis, tačiau, jei naudojama IRT teorija, galima testuoti ir labai dideles žmonių grupes.

1.2.6. Fiksuoti testai (angl. Computer Fixed Tests)

Tokio tipo testai yra tradicinio popierinio testo atitikmuo kompiuteriniuose testuose. Fiksuoti testai yra sudaromi paprasčiausio generavimo algoritmo. Testai yra fiksuoto ilgio. Algoritmas tik išmaišo klausimų ir (arba) atsakymų tvarką teste. Kartais klausimai pateikiami taip pat fiksuota tvarka.

Klausimų bazė dažiausiai būna nedidelė, todėl klausimų saugumas mažas. Papildžius klausimų bazę naujais klausimais, iš karto galima juos panaudoti sudarant klausimynus.

Tokio tipo testavimo sistemai išlaikyti reikalingi nedideli resursai ir kaštai. Klausimų bazės sudarymui nereikalaujama pilotinio testavimo.

Šio tipo testais testuojamos nedidelės testuojamųjų grupės arba jie naudojami specifinės srities žinių patikrinimui.

1.2.7. Automatizuotas testo surinkimas (angl. Automated Test Assembly)

Algoritmo veikimo principas paprastas. Iš didelės klausimų bazės atsitiktine tvarka parenkami klausimai. Taip sudaroma didesnė testų įvairovė. Tokio tipo testai gali veikti tiek tinkle, tiek atsijungus nuo tinklo.

Automatizuoto testo surinkimo testuose klausimų bazė daug didesnė nei fiksuoto tipo testuose, todėl padidinamas testo saugumas.

Šio tipo testams išlaikyti nereikia daug resursų. Priežiūra nesudėtinga. Pridėjus klausimą į bendrą bazę, iškarto galima jį panaudoti sudarant klausimyną.

ATA testai gali būti naudojami didelių testuojamų žmonių grupių žinioms tikrinti.

1.2.8. Tiesiniai testai realiu laiku (angl. Linear On-The-Fly Tests)

Šio tipo testo veikimo principas labai panašus į automatizuoto testo surinkimo metodą. Tačiau šio tipo testo klausimų aibė yra sugeneruojama atsitiktinai ar pagal tam tikrus kriterijus, kiekvieną

kartą, kai testas yra pradedamas. Todėl, kiekvieną kartą, pradėjus testą, gali būti sugeneruotas unikalus klausimynas. Tokio tipo testų generavimui pasitelkiamos klasikinė ar IRT teorijos. Šios teorijos pagalba testo saugumo ir tikslumo galimybės yra praplečiamos. Klausimai gali būti pateikiami, atsižvelgiant į testuojamojo atsakymus. Jei naudojama viena iš teorijų, šio tipo testą galima priskirti prie adaptyvių.

Reikalinga didesnė klausimų bazė. Tai užtikrina didesnį saugumą, nes klausimai rečiau kartojasi.

Reikalingi didesni sistemos išlaikymo kaštai bei resursai, nei prieš tai aprašytuose testų tipuose, nes klausimų bazė didelė. Taip pat, jei naudojama kuri nors ir klausimo sudėtingumo, ir pateikimo teorijų, reikalingi skaičiavimai ir papildomas tikrinimas kiekvienam klausimui.

Labai didelės testuojamų žmonių grupės testavimo galimybės. Tačiau reikėtų atsižvelgti į klausimų bazės dydį. Didesniam testuojamųjų skaičiui reikalinga daug didesnė klausimų bazė.

1.3. Generavimo metodų palyginimas

Išanalizuotus testų generavimo algoritmus ir atrinktus pastebėtus privalumus bei trūkumus matome žemiau pateiktoje lentelėje (1.2 lentelė).

1.2 lentelė. Generavimo metodų palyginimas

Metodas	Metodo privalumai	Metodo trūkumai
Fiksuoti testai	Testai kompiuterizuoti. Testo ilgis ir trukmė fiksuota. Testas vertinamas pagal teisingų atsakymų skaičių. Testo klausimai pateikiami atsitiktine tvarka.	Labai mažai testo variantų. Mažas saugumas. Paprasčiausias metodas.
Automatizuotas testo surinkimas	Testo ilgis ir trukmė fiksuota. Daug didesnis testo variantų skaičius. Testų variantai gali būti sukurti iš anksto prieš testą. Testas gali būti kuriamas pagal reikalavimus.	Testas gali būti kuriamas pagal reikalavimus, tačiau, tuo atveju, jei kažkuris reikalavimas neįvykdomas arba vienas kitam prieštarauja – testo generavimas neįvykdomas.

Tiesiniai testai realiu laiku	Didesnis saugumo lygis. Testas generuojamas kiekvienam testuojamajam atskirai. Fiksuotas ilgis.	Nepakankamai variantų, nors generuojama kiekvienam atskirai, todėl saugumo lygis nedidelis.
Kompiuteriniai klasifikavimo testai	Testo ilgis ir trukmė priklauso nuo atsakymų į testo klausimus. Testo rezultatai tikslūs. Testo saugumas žymiai didesnis.	Skirti grupavimui į dvi ir daugiau grupių. Reikalinga didelė klausimų bazė.
Kompiuteriniai adaptyvūs testai	Testo ilgis ir trukmė priklauso nuo atsakymų į testo klausimus. Pateikiami klausimai labiau atitinka testuojamojo žinias. Testo rezultatai tikslūs. Testo saugumas žymiai didesnis, nes pateikiamas minimalus klausimų kiekis, testo klausimynas unikalus.	Reikalinga didelė klausimų bazė. Sudėtingas klausimų bazės paruošimas. Nėra galimybės grįžti į ankstesnį klausimą, kas sukelia testuojamųjų nepasitenkinimą.

1.4. Technologijų analizė

Pradedant analizuoti galimas technologijas, pirmiausia išsiaiškinama ir atsižvelgiama į funkcinius reikalavimus, kuriais remiantis bus pasirinktos sistemos kūrimo technologijos.

Užsakovas aiškiai išskiria, kad kuriama sistema turi palaikyti ne vieną nutolusią mokymo klasę. Todėl sistemą buvo nuspręsta kurti, pasinaudojant naujausiomis žiniatinklio technologijomis, kurios gali perduoti informaciją internetu. Tokiu būdu mokomoji priemonė bus pasiekiama iš mokinių asmeninių kompiuterių ir mokymo klasėse.

1.4.1. HTML5 (Hyper Text Markup Language)

Tai pagrindinė kalba, kurią naudojant atvaizduojamas internetinio puslapio turinys moderniose interneto naršyklėse. HTML kalbos pagalba aprašoma puslapio struktūra, kurią interneto naršyklė interpretuoja pagal HTML elementų žymas. Naujausias HTML5 standartas praplėtė ir papildė esamas HTML kalbos galimybes, tokias kaip vaizdo ir garso įrašų įterpimas.

1.4.2. CSS3

Tai kalba, skirta nusakyti elementų išvaizdą ir formatuoti dokumentų ženklinimo kalba parašytus dokumentus. Svetainės išvaizda aprašyta naujausiu CSS3 standartu. Kuriant sistemą, buvo panaudotos šios funkcijos [12]:

- Box Sizing funkcija;
- Media Queries. Stilių pritaikymas prie skirtingų ekrano dydžių ar medijos tipo;
- CSS stulpeliai. Teksto skaidymui į stulpelius;
- CSS 3D Transforms. Elementų 3D transformavimui. Meniu elementų apsukimas.

1.4.3. JavaScript

JavaScript yra viena pagrindinių programavimo kalbų, naudojamų tinklapių kūrime, kai reikia puslapiui ar jo elementams suteikti papildomo interaktyvumo galimybių. Praplečia galimybę valdyti interneto naršyklę bei atlikti veiksmus kliento pusėje. Realizuojant projekto interaktyvumo funkcijas, taip pat buvo naudojamasi JavaScript pagalba.

1.4.4. PHP

Tai viena iš labiausiai paplitusių programavimo kalbų, kuri buvo sukurta 1995 metais. Pati kalba buvo kurta internetinių svetainių programavimui, tačiau neretai naudojama ir paprastų aplikacijų kūrime. PHP kalbos sintaksė labai panaši į kitų kalbų, tačiau turi savų ypatumų. Pagrindiniai PHP kalbos privalumai :

- veikia daugelyje esamų operacinių sistemų;
- nesunkiai gali būti suderinama su HTML kalba;
- nemokamas atviro kodo produktas;
- palaiko daugelį pagrindinių duomenų bazių: MySQL, Oracle, MySqli, MSSql;
- gali būti suderinama su pagrindiniais internetiniais serveriais.

Dėl savo veikimo serverio pusėje PHP programos kodas nėra matomas kliento pusėje, todėl tai yra vienas iš pranašumų, kuriant dinaminius internetinius puslapius. PHP suteikia papildomų galimybių vykdyti skaičiavimus, nepasiekiant programinio kodo iš kliento pusės.

1.4.5. Laravel 5

Sistemos kūrimui buvo pasirinktas PHP pagrindu parašytas tinklalapių kūrimo karkasas „Laravel 5“. Tai – iš esmės atnaujintas „Laravel 4“. Pakeista sisteminių failų aplankų struktūra. Nauja maršrutų (angl. Routes) kešavimo funkcija ženkliai padidina aplikacijos greitaveiką. Taip pat vienas iš

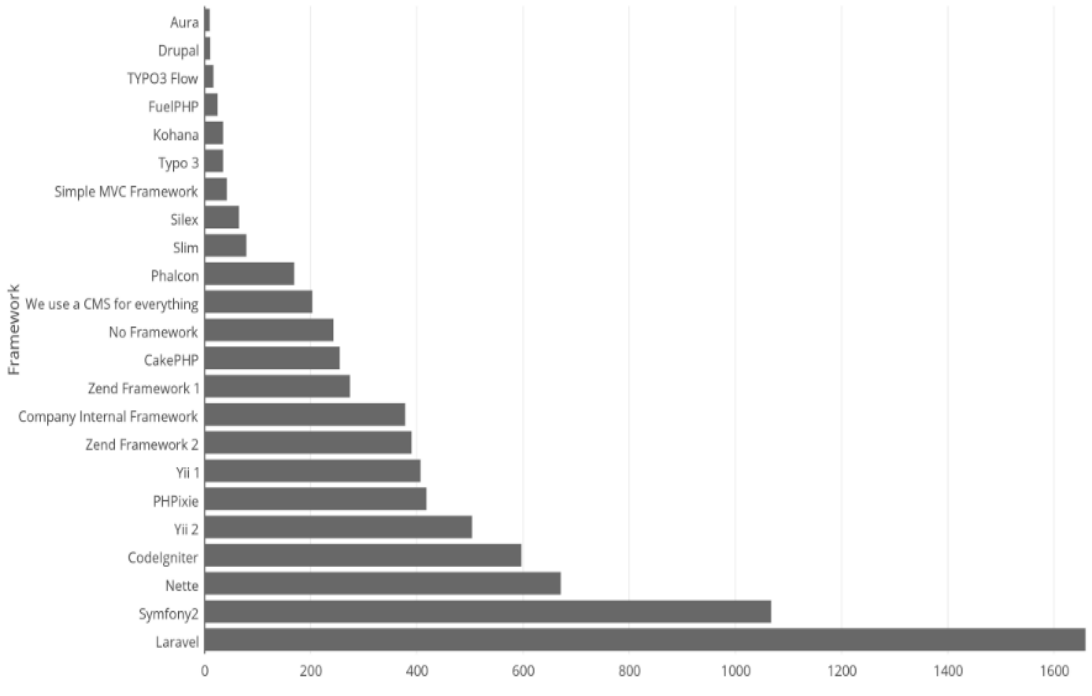
privalumų yra tai, kad bazinėje sudėtyje turi naudotojų prisijungimo funkcijas, naudotojų sesijų valdymą ir duomenų kešavimą (angl. Cache).

Vienas iš pagrindinių Laravel įrankių yra „Eloquent“ sąsaja tarp duomenų bazės ir karkaso. Šio įrankiu programuotojas vos keliais programos kodo sakiniais gali gauti reikiamus duomenis arba sudėtingos SQL komandos tampa kelių žodžių junginiu.

Kitas patogus „Laravel“ įrankis tai – „Blade“ šablonai, kurie padeda atvaizduoti reikiamus duomenis ir internetinės aplikacijos puslapius, minimaliai panaudojant HTML [13]. Šiuo įrankiu kuriami puslapio šablonai gali būti apjungiami, naudojami begales kartų ir papildantys vienas kitą.

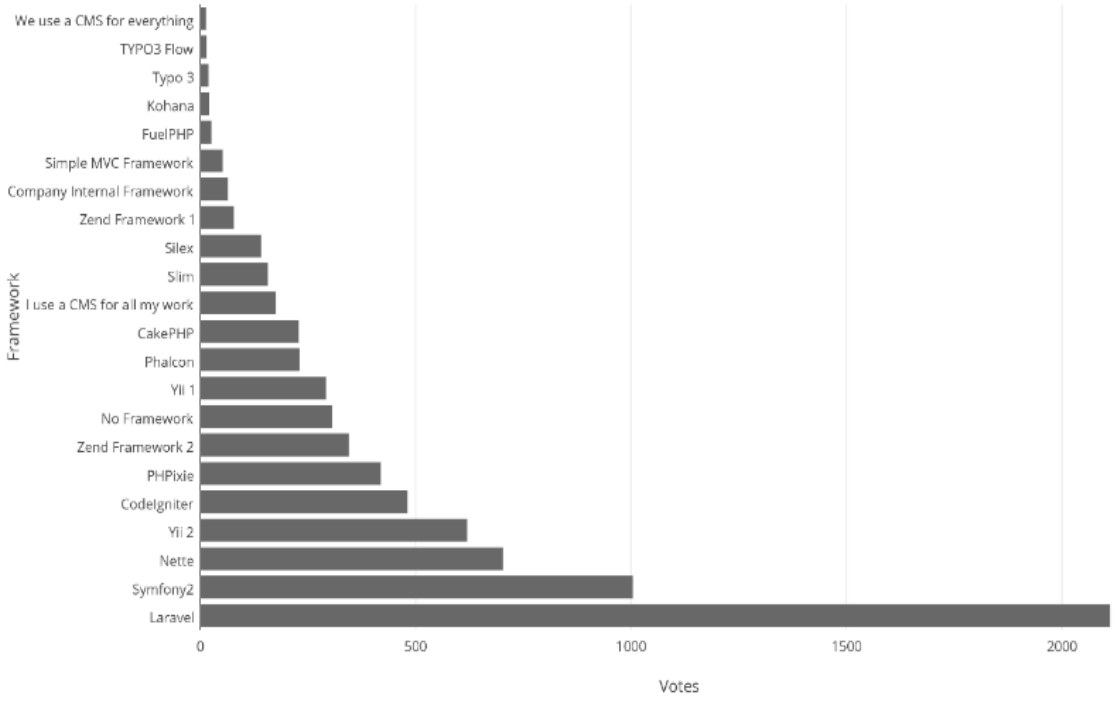
Šis karkasas pasirinktas dėl savo paprastumo programuojant, konfigūravimo galimybių įvairumo, diegimo į esamą svetainės talpyklą galimybės. Taip pat viena iš priežasčių, kodėl buvo pasirinktas šis karkasas, yra tai, jog su 4 versija buvo sukurta keletas svetainių ar bandomųjų variantų ir rezultatas tenkino tiek užsakovus, tiek mane.

Vadovaujantis Remiantis Bruno Skvorc [14] atlikta apklausa, „Laravel 5“ taip pat labai populiarus tarp svetainių kūrėjų. Apklausos rezultatai pateikiami žemiau esančioje diagramoje. Ši diagrama parodo karkaso pasirinkimą, kuriant svetainės projektus darbe. Diagramoje matyti, jog Laravel ženkliai, kode dvigubai lenkia kitus svetainių kūrimo karkasus (žr. 1.10 pav.).



1.10 pav. Karkasų pasirinkimas kuriant projektus darbe (17)

Apklausoje taip pat matoma (žr. 1.11 pav.), kad asmeniniams projektams Laravel karkasas ženkliai pirmauja tarp svetainių kūrimo karkasų. Antroje vietoje lieka Symfony2, atsilikdamas daugiau nei dvigubai.



1.11 pav. Karkasų pasirinkimas kuriant asmeninius projektus [14]

Atlikus rinkoje esamų Kelių eismo taisyklių testų sprendimo sistemų analizę, pasitvirtino, kad sprendžiant testus pakartotinai, kelis ar keliasdešimt kartų, klausimai kartojasi. Vieni klausimai pateikiami dažniau nei kiti.

Atlikus testų analizę, paaiškėjo kokios testų rūšys, gali būti pritaikytos optimizuojant testo klausimų pateikimą naudotojams. Mokomiesiems testams buvo pasirinktas automatizuotas testo surinkimo tipas, nes galima sudaryti didelius skirtingų testų rinkinius. Egzaminavimo testams – kompiuterinio adaptyvaus testo tipas, nes yra daug saugesnis, nei kiti ir parodo tikslius naudotojo gabumus.

Suradus ir palyginus algoritmus, kurie gali būti panaudoti testų generavimui, sužinota, kad daugelis atsitiktinių skaičių generavimo algoritmų turi sisteminę paklaidą. Dėl jos, vieni klausimai pateikiami dažniau nei kiti.

Atlikus galimų technologijų analizę, pasirinktas, patogus ir daugelio kitų programuotojų įvertintas internetinių aplikacijų kūrimo karkasas -„Laravel 5“.

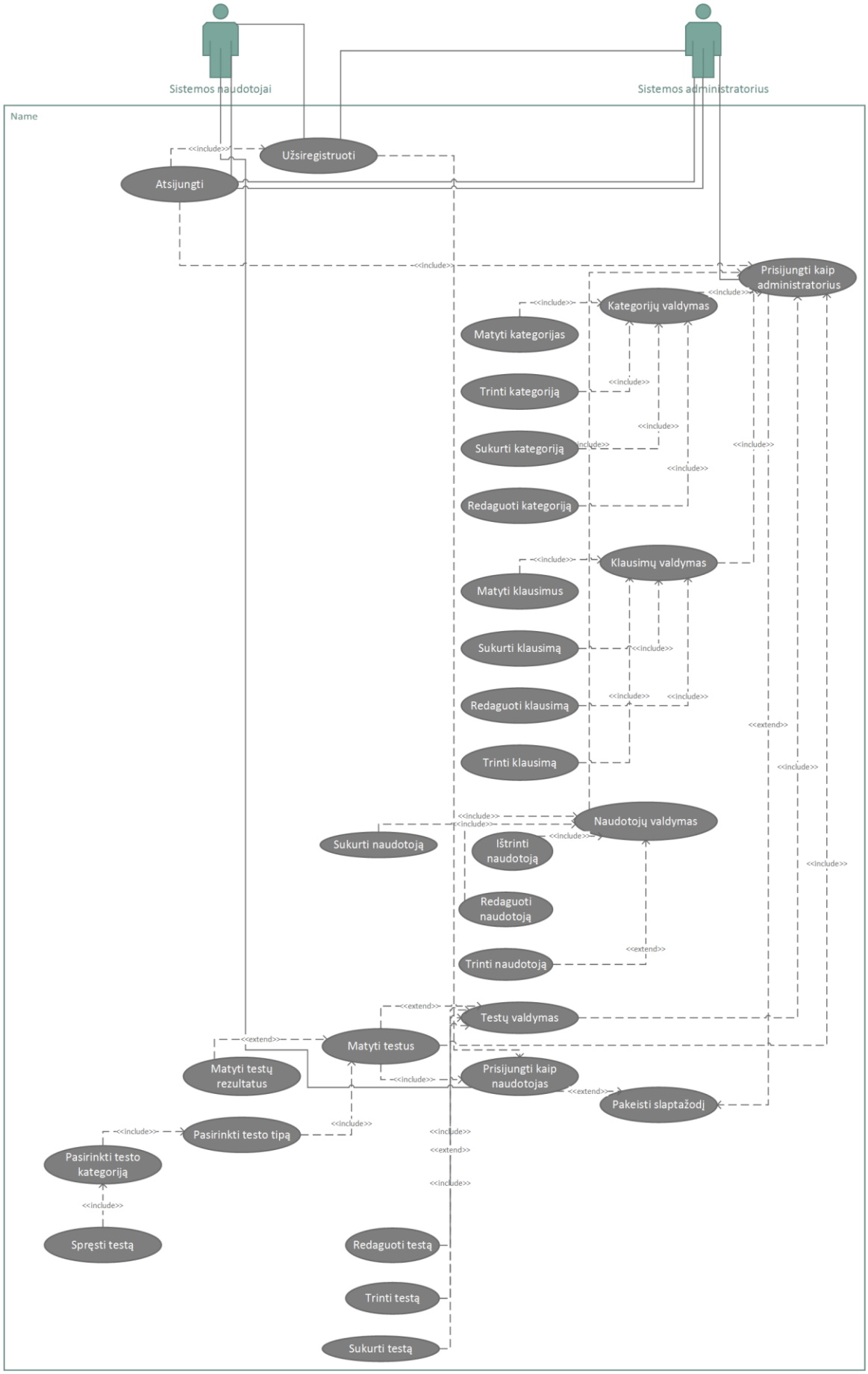
2. PROJEKTINĖ DALIS

2.1. Funkciniai reikalavimai

Bendri funkciniai reikalavimai projektuojamai sistemai:

- sistemos administratorius, dėstytojo ir testuojamųjų prisijungimo teisės;
- sistema turi palaikyti kelis nuotolinius mokymo taškus;
- naudotojų registracija arba prisijungimas per esamą vairavimo mokyklos sistemą;
- viskas saugoma duomenų bazėje;
- galimybė spręsti testus internetu;
- testo kūrimas;
- testo redagavimas;
- testo panaikinimas;
- klausimo kūrimas;
- klausimo redagavimas;
- klausimo panaikinimas;
- kategorijos kūrimas;
- kategorijos redagavimas;
- kategorijos panaikinimas;
- galimybė pasirinkti testo tipą;
- galimybė pasirinkti testo generavimo algoritmus;
- optimizuotas klausimų generavimas;
- klausimų pateikimo algoritmas, prisitaikantis prie kiekvieno testuojamojo individualiai;
- mokinių rezultatų peržiūra;
- testas turi pateikti kuo tikslesnius rezultatus.

Sudarius funkcinių reikalavimų specifikaciją, buvo sudaryta panaudos atvejų diagrama (žr. 2.1 pav.)



2.1 pav. Sistemos panaudos atvejų diagrama

2.2. Nefunkciniai reikalavimai

Bendri nefunkciniai reikalavimai projektuojamai sistemai:

- sistema turi turėti patogią naudotojo sąsają;
- sistema ir algoritmai turi veikti naudotojui priimtinu greičiu;
- sistema turi turėti kelis testo generavimo algoritmus.

2.3. Realizacijai keliami reikalavimai

Sistema projektuojama, atsižvelgiant į šiuos reikalavimus:

- sistemos realizacijai panaudotas puslapių kūrimo karkasas;
- tik prisijungę naudotojai gali įkelti, pakeisti ar ištrinti sistemoje esančius duomenis;
- sistema programuojama, naudojant PHP, HTML, CSS, JavaScript kalbas;
- duomenys apie naudotojus ir kita sistemai priklausanti informacija saugoma MySQL duomenų bazėje.

2.4. Naudotojo sąsajos specifikacija

Sistemos naudotojo sąsaja yra vienas pagrindinių sistemos komponentų, nes sistema per ją bendrauja su naudotoju. Todėl būtina, kad šis sistemos komponentas būtų suprojektuotas tinkamai ir atitiktų šiuos reikalavimus:

- patogus sistemos navigacijos meniu;
- testai turi veikti naudotojui priimtinu greičiu;
- sistemos išvedami duomenys turi būti formatuoti ir priimtini naudotojui;
- testo išdėstymo struktūra lengvai suprantama.

Kuriant sistemos naudotojo sąsają, naudotos tik CSS3 teikiamos galimybės, tokios, kaip teksto šešėliai, pereinamos elementų fono spalvos, taip maksimaliai padidinat suderinamumą su interneto naršyklėmis.

Judantiems svetainės elementams sukurti buvo panaudotos CSS3 animacijos ir transformacijos funkcijos. Pereinantys elementų fonų atspalviai sukurti, naudojant CSS „gradient“ funkcijas ir „rgba“ spalvos kodą, padarant juos reikiamo ryškumo.

Sistemos interaktyvumui išgauti buvo naudota „:hover“ funkcija ir pažangus elementų priskyrimas, naudotas meniu išskleidimui.

Dinamiškai naudotojo sąsajai pasiekti buvo panaudota JavaScript kalba ir jQuery biblioteka, taip pasiekta, jog testo klausimai nerodomi visi vienu metu ir naudotojas turi galimybę vaikščioti pirmyn ir atgal tarp klausimų. Taip pat jQuery pagalba buvo realizuotos ir kitos sistemos funkcijos.

2.5. Techninė specifikacija

Internetinei svetainei peržiūrėti nereikia daug resursų. Jei aparatinė įranga atitinka turimos interneto naršyklės sisteminius reikalavimus, to turėtų pakakti naudotis šia sistema. Testavimo sistema pasiekiama per mobiliuosius įrenginius, tačiau patartina, kad įrenginio procesoriaus greitis būtų kuo didesnis.

Reikalaujama, kad sistema turi veikti užsakovo turimame serveryje. Todėl sistema derinta ir optimizuota taip, kad naudotų kuo mažiau sistemos resursų. Sistemos veikimui būtina užtikrinti, kad serveryje būtų įdiegtos šios programinės įrangos ir papildiniai [15]:

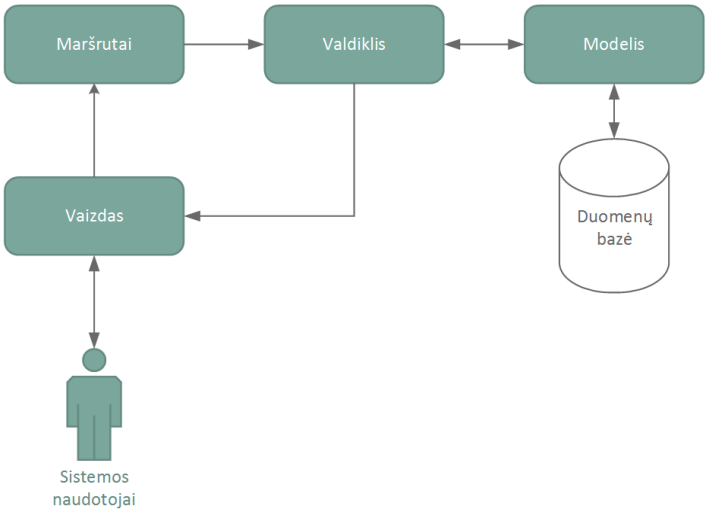
- PHP 5.5.9 ir aukštesnės versijos palaikymas;
- MySQL duomenų bazė, kuri palaikytų PDO;
- Mcrypt biblioteka;
- „Apache“ programinė įranga;
- OpenSSL PHP Extension;
- Mbstring PHP Extension;
- Tokenizer PHP Extension.

2.6. Projektavimo valdymas ir eiga

Dėl aiškių funkcionalumo reikalavimų suformavimo projektavimo pradžioje ir technologijų analizės, sistemos kūrimas vyko nuosekliai, įgyvendinant reikiamas sistemos funkcijas, vėliau jas testuojant ir ištaisant iškilusias problemas.

2.7. Projektavimo technologija

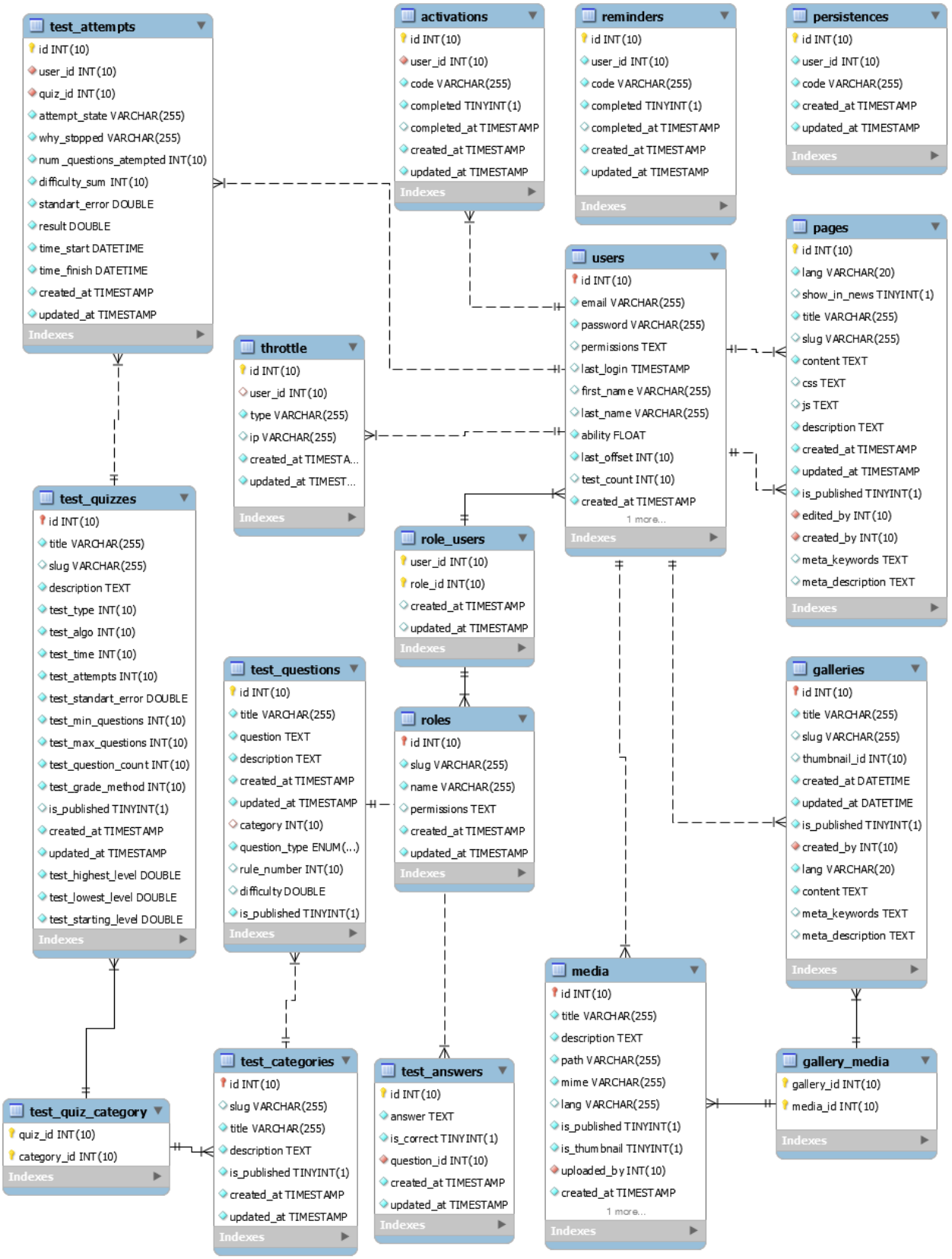
Naudojamas internetinių aplikacijų kūrimo karkasas „Laravel 5“ veikia MVC architektūros principu, matomu paveiksle (žr. 2.2 pav.). Taip supaprastinamas ir sistemos kūrimas. Valdiklis nustato, kokį turinį išvesti naudotojui. Sistema laukia naudotojo veiksmų, ar navigacijos meniu pasirinkimo. Atlikus veiksmą sistema per maršrutų nuorodas nustato, kurį valdiklį ar jo funkciją vykdyti. Valdiklio funkcija kreipiasi į reikiamą modelį, kuris paima reikiamus duomenis iš failo ar duomenų bazės [16]. Modelio grąžinti duomenys per valdiklį pateikiami naudotojui šablonų pagalba.



2.2 pav. MVC principas

2.8. Reliacinė duomenų bazės schema

Reliacinė duomenų bazės schema, kurioje nurodyti duomenų lentelių sąryšiai, pateikta paveiksle (žr. 2.3 pav.).



2.3 pav. Duomenų bazės reliacinė schema

2.9. Sistemos realizavimui naudotos programinės įrangos ir operacinė sistemos

Sistemos kūrimui buvo panaudota:

- „WampServer“ serverio aplinka, kuri turi „Apache“, PHP, MySQL duomenų bazės valdymo modulius. Skirta svetainės talpinimui;
- „Windows 10” kompiuterio operacinė sistema, su kuria buvo atliekamas visas darbo procesas;
- „Android“ kelių versijų mobiliųjų įrenginių operacinės sistemos.
- „Google Chorme“ naršyklė, kurioje buvo bandoma svetainė ir sistema;
- „Mozilla Firefox“ naršyklė, kurioje buvo bandoma svetainė ir sistema;
- „phpStorm 10“ bandomoji versija. Programavimo redaktorius, su kuriuo buvo rašomas sistemos programinis kodas;
- „Microsoft Visio“ programa, su kuria buvo daromos UML diagramos.

2.10. Duomenų kontrolė

Visi duomenys, kurie yra įvedami į testavimo sistemą, pirmiausiai yra tikrinami HTML5 standarto suteikiamais duomenų tikrinimo taisyklių rinkiniais. HTML5 leidžia atsisakyti JavaScript duomenų tikrinimo metodų. Taip pagreitėja formų kūrimas bei sumažėja programinio kodo dydis, norint išgauti tokį patį funkcionalumą.

Kiekvienam sistemoje įvedamo atributo laukui yra nustatytos ribos ir šablonas, kurį turi atitikti. Neatitikus pradinės duomenų patikros, parodomi laukelio reikalavimai. Tik teisingai suvesti duomenys, kurie atitiko pradinį HTML5 taisyklių šabloną, siunčiami į serverį.

Serveryje visi laukai vėl yra tikrinami Laravel 5 karkaso suteikiamu duomenų tikrinimo moduliu. Reikiamiems įvedimo laukams yra sudaromas šablonas, kurį turi atitikti, ir tik tada duomenys yra siunčiami į duomenų bazę. Neatitikus reikiamam šablonui, puslapis, iš kurio buvo įvedami duomenys, yra iš naujo užkraunamas, kartu išvedant klaidos ar perspėjimo pranešimus.

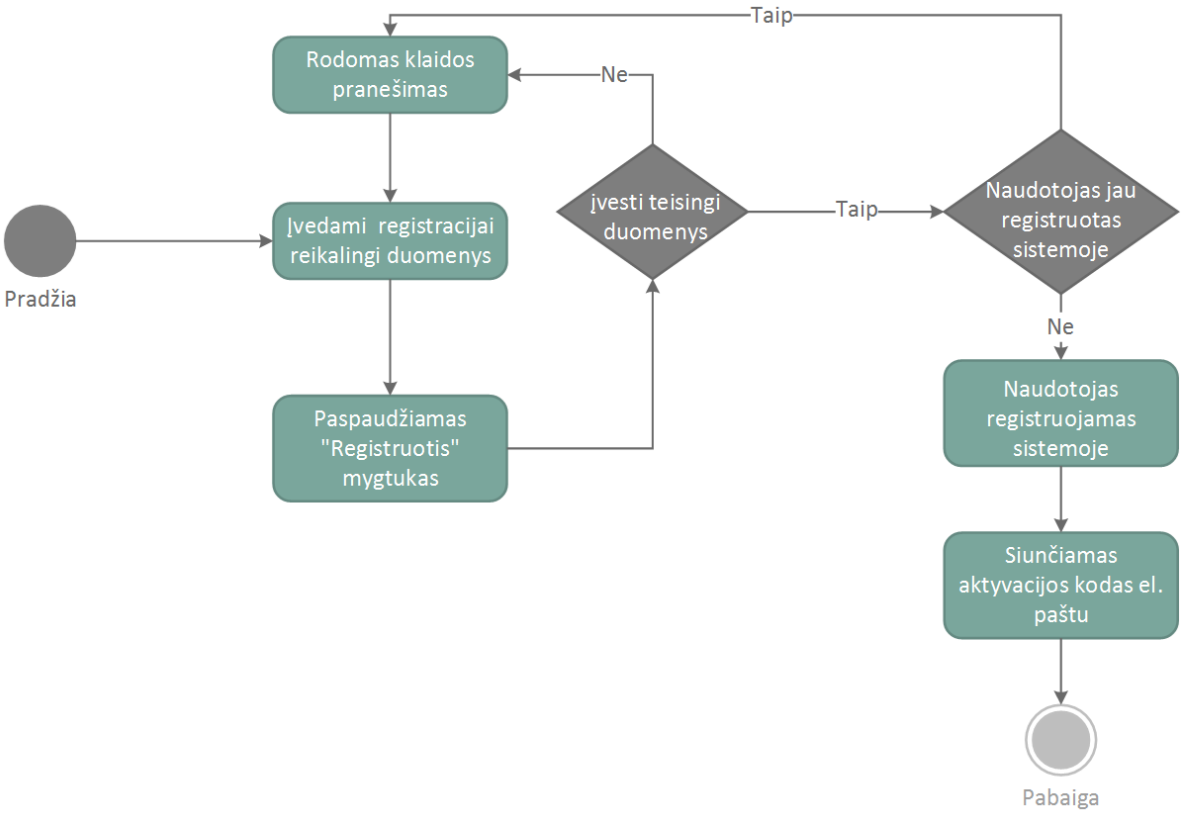
3. REALIZACINĖ DALIS

3.1. Pagrindinių sistemos funkcijų realizacija

Suprojektavus sistemą, pasirinkto internetinių aplikacijų kūrimo karkaso pagalba buvo realizuojamos funkcijos ir pasirinkti algoritmai.

3.1.1. Registracijos sistemoje realizavimas

Naudotojui, užpildžius registracijos formą ir paspaudus mygtuką „Registruotis“, duomenys yra patikrinami lokaliai ir tik tada siunčiami į serverį, kur vėl yra tikrinami. Neatitikus duomenų, yra išvedamas klaidos pranešimas ties kiekvienu įvedimo lauku, kuris neatitiko reikalavimų. Jei suvesti duomenys atitiko visus reikalavimus, jie yra įrašomi duomenų bazėje ir nusiunčiamas elektroninis laiškas naudotojui. Veiklos diagrama pateikiama žemiau (žr. 3.1 pav.)

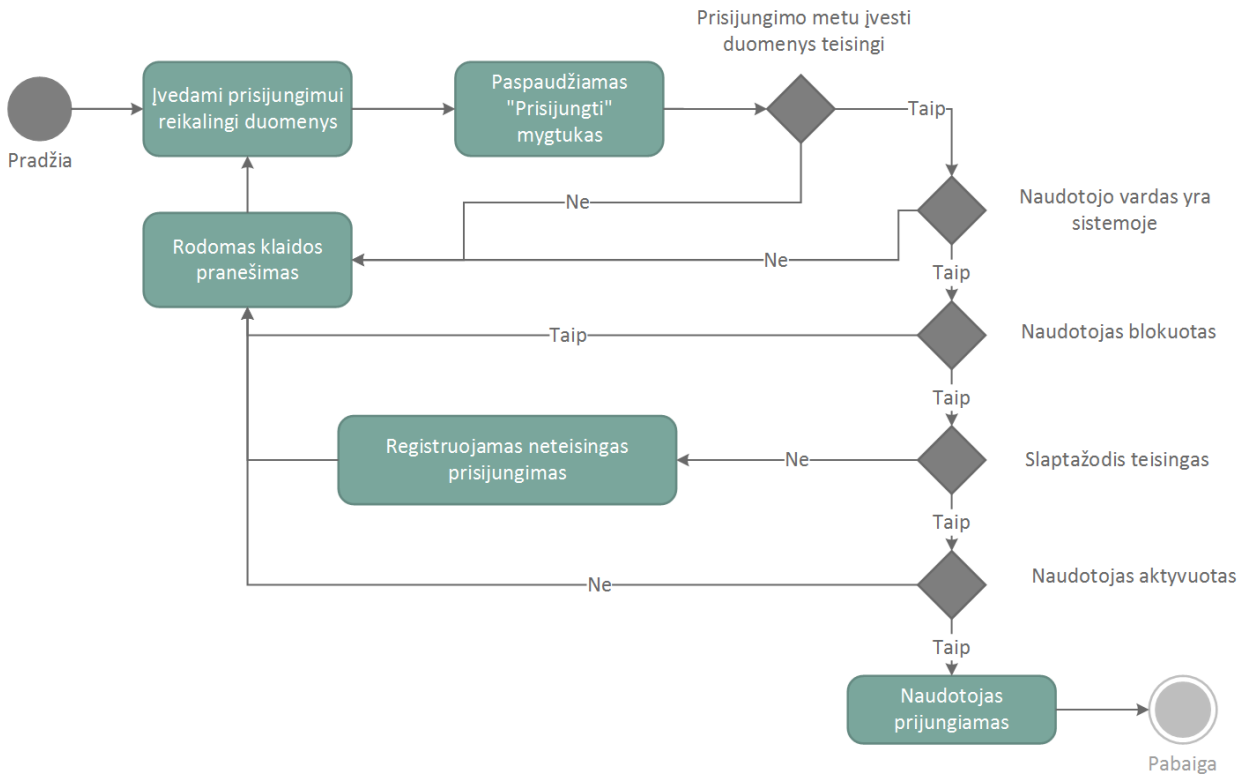


3.1 pav. Naudotojo registracijos sistemoje veiklos diagrama

3.1.2. Prisijungimo sistemoje realizavimas

Naudotojui, užpildžius prisijungimo formą ir paspaudus mygtuką „Prisijungti“, sistema patikrina, ar įvesti duomenys atitinka reikalavimus. Neatitikus reikalavimų, grįžtama į buvusį puslapį, kartu išvedant klaidos pranešimą. Atitikus reikalavimus, tikrinama, ar naudotojo prisijungimo vardas yra duomenų bazėje, radus - toliau tikrinama, ar naudotojas nėra blokuotas. Jei naudotojo vardas nerastas arba blokuojamas, išvedamas klaidos pranešimas ir grąžinama į prisijungimo formos langą. Atitikus

prieš tai minėtus reikalavimus, tikrinama, ar atitinka naudotojo įvestas slaptažodis. Jei taip, patikrinama, ar naudotojas aktyvuotas. Jei slaptažodis neteisingas, registruojama, jog naudotojui nepavyko prisijungti. Jei praeinami visi prisijungimo žingsniai, naudotojas sėkmingai prijungiamas prie sistemos ir nukreipiamas į profilio puslapį, kartu išvedant informacinį pranešimą. Veiklos diagrama pateikiama paveiksle (žr. 3.2 pav.).



3.2 pav. Prisijungimo į sistemą veiklos diagrama

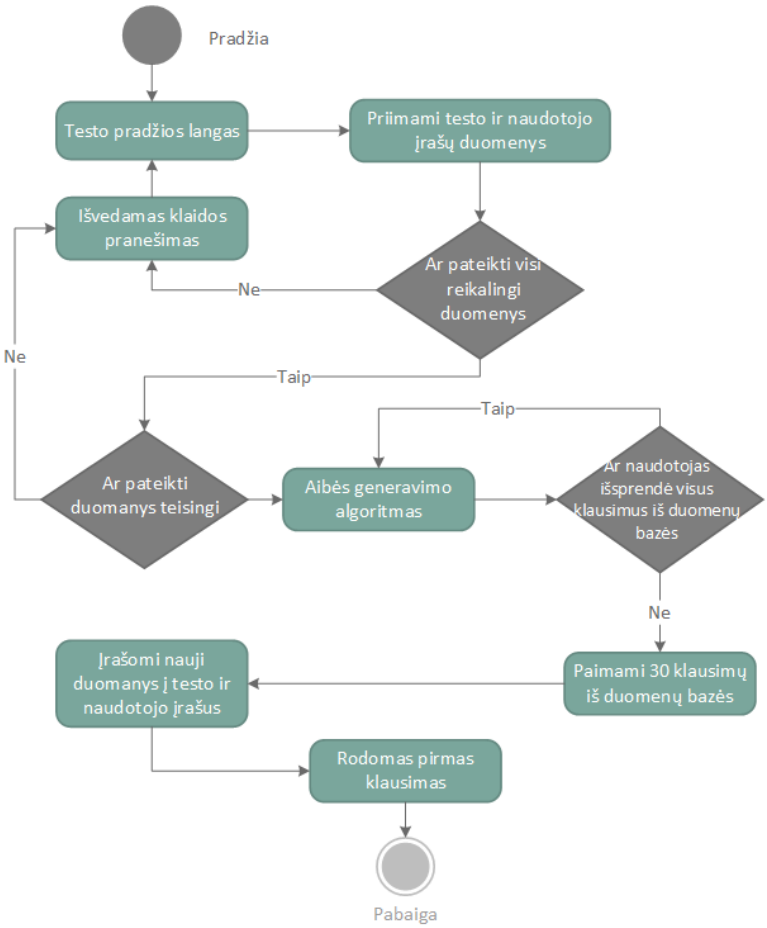
3.1.3. Testo generavimo realizavimas

Pradžioje paimami reikalingi naudotojo ir testo, kurį sprendžia, duomenys. Patikrinama, ar įvesti visi duomenys tinkami generavimui pradėti. Toliau tikrinama, ar naudotojas išsprendė visus duomenų bazėje esančius klausimus. Jei taip - optimizuoto algoritmo pagalba išmaišoma visa klausimų duomenų bazė ir nustatomas pradinis paskutinio klausimo numeris į 1. Toliau paimami pirmi 30 klausimų nuo paskutinio spręsto klausimo, šio atveju 1. Sprendžiant testą, kiekvieną kartą pateikiami sekantys 30 klausimų, kol naudotojas išsprendžia visus galimus klausimus.

3.1.4. Skaičių generavimas naudojant Mersenne Twister algoritmą

Generuojant testą Mersenne Twister algoritmu, atsitiktinai paimamas vienas elementas iš viso klausimyno masyvo. Paimtas elementas dedamas į testui ruošiamą klausimų masyvą. Jei sugeneruotas skaičius jau yra šiame masyve, iš naujo paimamas naujas elementas ir tikrinamas su kuriamu masyvu tol, kol bus paimtas unikalus skaičius. Sugeneravus visą masyvą, klausimai testų sistemos pagalba po

vieną pateikiami naudotojui. Algoritmo realizacijos programinis kodas pateikiamas žemiau paveiksle (žr. 3.3 pav.).



3.3 pav. Testo generavimo veiklos diagrama

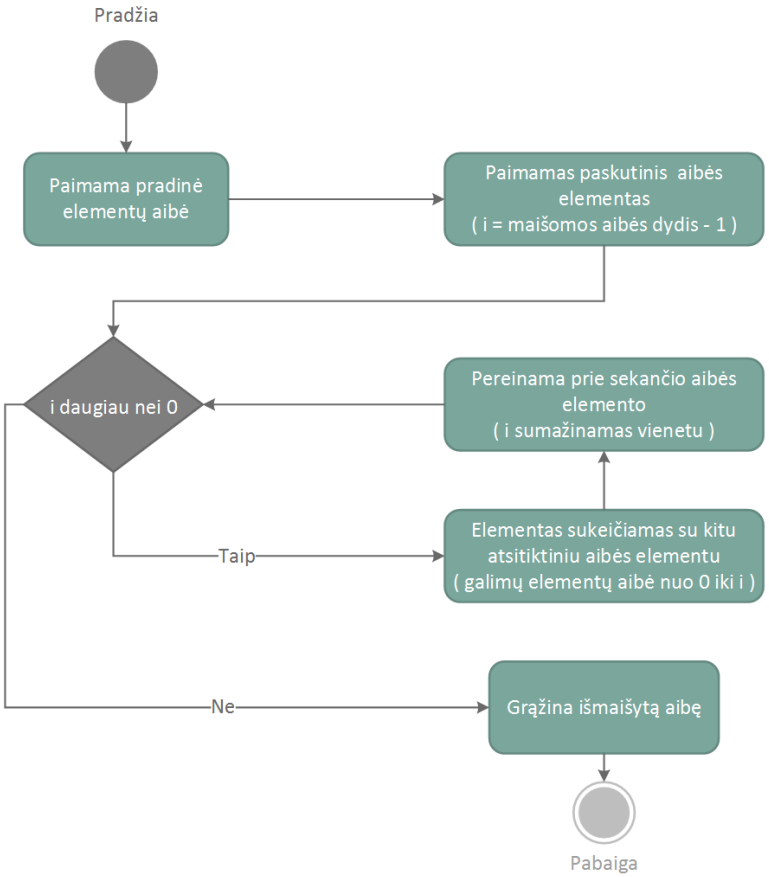
```
$uniques = array();
   for ($s = 0; $s < $questions_count; $s++) {
         $num = mt_rand(1, $database_size);
         if (in_array($num, $uniques)) {
         $num = mt_rand(1, $database_size);
      }
      $uniques[$s] = $num;
}
```

3.1.5. Fisher - Yates algoritmo realizavimas

Algoritmas buvo realizuotas dviem pasirinktais principais. Pirmu atveju algoritmas veikia iteraciniu metodu. Iš eilės einama per visą maišomą elementų aibę nuo galo ir sukeičiamas esamas elementas su atsitiktiniu tos pačios aibės elementu. Nors šis algoritmas beveik nesiskiria nuo paprasto, prieš tai analizuoto algoritmo, tačiau turi vieną esminį privalumą – po kiekvienos iteracijos galimų atsitiktinių elementų skaičius sumažinamas vienetu. Taip po kiekvienos iteracijos algoritmas turi vis mažesnę galimybių aibę. Todėl veikia greičiau, nei prieš tai aprašyti algoritmai. Veiklos diagrama

pateikiama paveiksle (žr. 3.4 pav.). Algoritmo programinio kodo realizacijos fragmentas pateiktas žemiau.

```
function fisherYates1($items)
{
    for ($i = count($items) - 1; $i > 0; $i--) {
        $j = mt_rand(0, $i);
        $tmp = $items[$i];
        $items[$i] = $items[$j];
        $items[$j] = $tmp;
    }
    return $items;
}
```

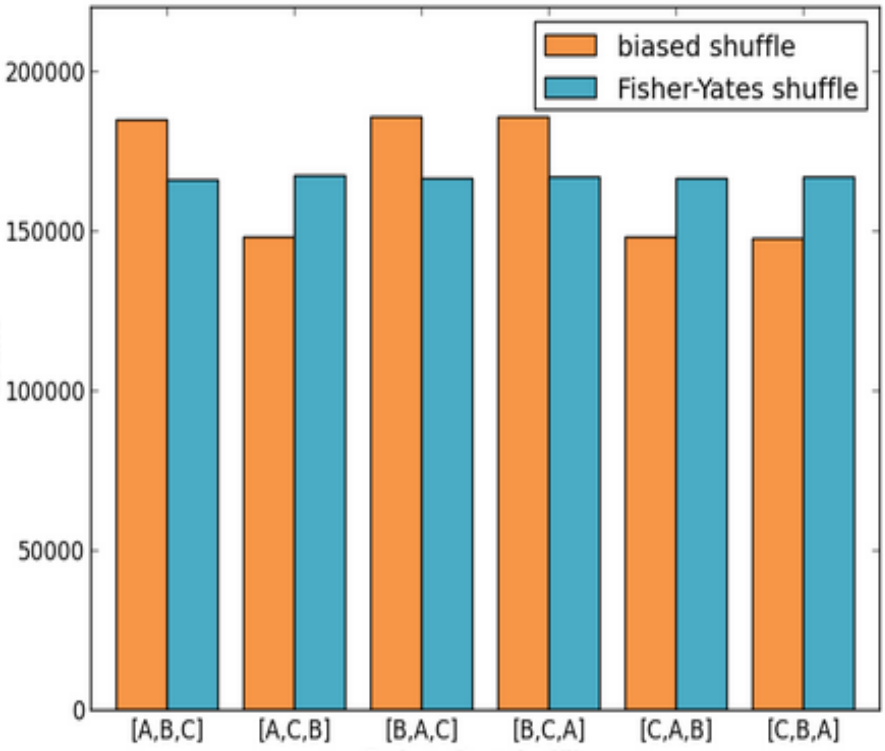


3.4 pav. Fisher - Yates (Knuth) algoritmo veiklos diagrama

Antru atveju algoritmas veikia taip pat iteraciniu metodu. Iš eilės einama per visą maišomą elementų aibę, tik šį kartą nuo pradžios ir sukeičiamas esamas elementas su atsitiktiniu tos pačios aibės elementu. Nors šis algoritmas beveik nesiskiria nuo prieš tai realizuoto algoritmo, tačiau turi vieną esminį skirtumą. Šitaip realizavus algoritmą sulėtėja pačio algoritmo greitaveika.

```
function fisherYates2($items)
{
    for ($i = 0; $i < count($items) - 1; $i++) {
        $j = mt_rand($i, count($items) - 1);
        $tmp = $items[$j];
        $items[$j] = $items[$i];
        $items[$i] = $tmp;
    }
    return $items;
}
```

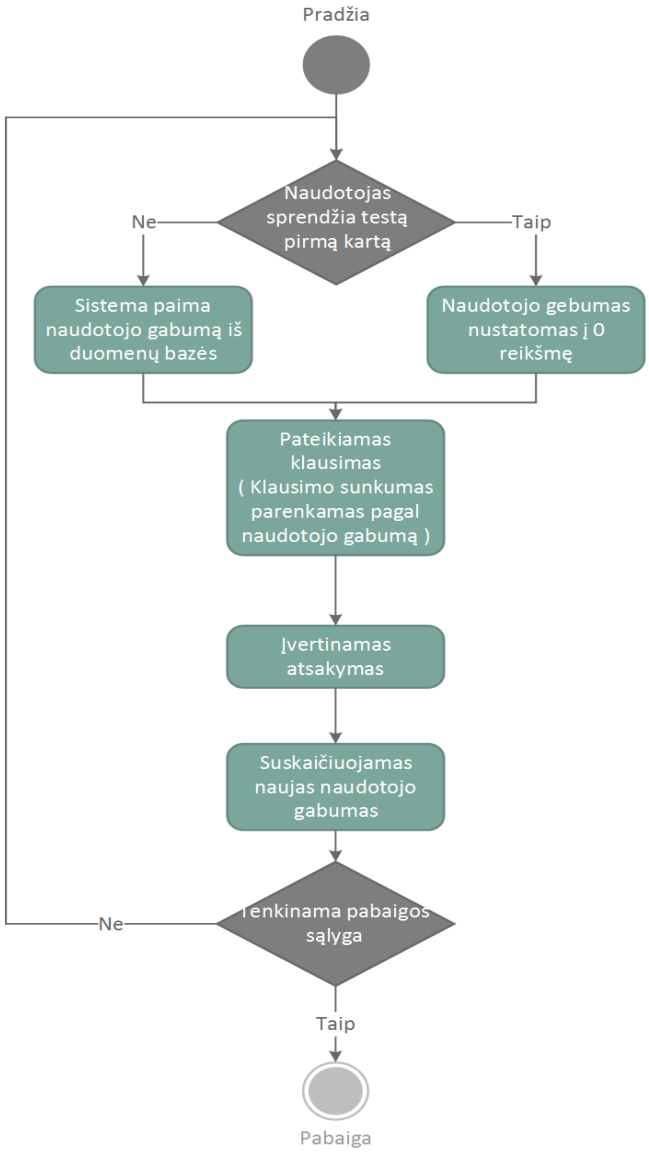
Taip pat, realizuojant algoritmą, galima lengvai sugadinti šio algoritmo veikimo principą ir pridėti sisteminę paklaidą, kaip aprašo Mattas Nedrichas [17] savo darbe, kuriame palygino teisingai ir neteisingai įgyvendinto algoritmo rezultatus. Matome (žr. 3.5 pav.), kad neteisingai realizuotas algoritmas netolygiai išmaišo aibę.



3.5 pav. Teisingai ir neteisingai realizuoto Fisher - Yates algoritmo rezultatų diagrama [17]

3.1.6. Adaptyvaus algoritmo realizavimas

Pagrindinis šio algoritmo tikslas – prisitaikyti ir parinkti optimaliausią klausimą naudotojui. Atliekant testą, po kiekvieno teisingai ar neteisingai atsakyto klausimo, algoritmas paskaičiuoja laikiną naudotojo gabumo lygį. Sekančiam klausimui pagal formulę paskaičiuojami rėžiai tarp galimo lengviausio ir sunkiausio klausimų. Toliau algoritmas atsitikrinai parenka viena klausimą iš galimo sunkumo rėžių. Klausimai pateikiami naudotojui, kol įvykdoma testo pabaigos sąlyga. Supaprastintą algoritmo veiklos diagramą matome paveiksle (žr. 3.6 pav.).



3.6 pav. Adaptyvaus algoritmo veikimo veiklos diagrama

3.1.7. Klausimo pateikimo realizavimas

Generuojant 30 klausimų testus, visi unikalūs klausimai sugeneruojami iš anksto. Sprendžiant testą, eilės tvarka pateikiama po vieną klausimą iš masyvo.

Pateikiant klausimus adaptyvaus testo metu, pats algoritmas kiekvieną kartą paskaičiuoja sekančio klausimo sunkumą pagal formulę. Pirmiausia paverčiamas sudėtingumo lygis į logitus (angl. logit). Realizacijos programinio kodo fragmentai pateikiami žemiau.

```
$percents = ($level - $min) / ($max - $min);
...
$steps = 1 / ($max - $min);
$percent_step = $steps / 2;
```

```
...
$logit = log( $percents / (1 - $percents) );
```

Toliau, kai nustatomas sudėtingumas, tikrinama, ar teisingai atsakyta į paskutinį pateiktą klausimą. Pagal atsakymą nustatoma, ar sekantis klausimas bus sudėtingesnis ar lengvesnis. Žemiau matomas programinio kodo fragmentas, kuris realizuoja šį parinkimą.

```
 if ($answer_correct) {
        $nextdiff = $ls + 2 / $sum_questions_attempted;
 } else {
        $nextdiff = $ls - 2 / $sum_questions_attempted;
 }
```

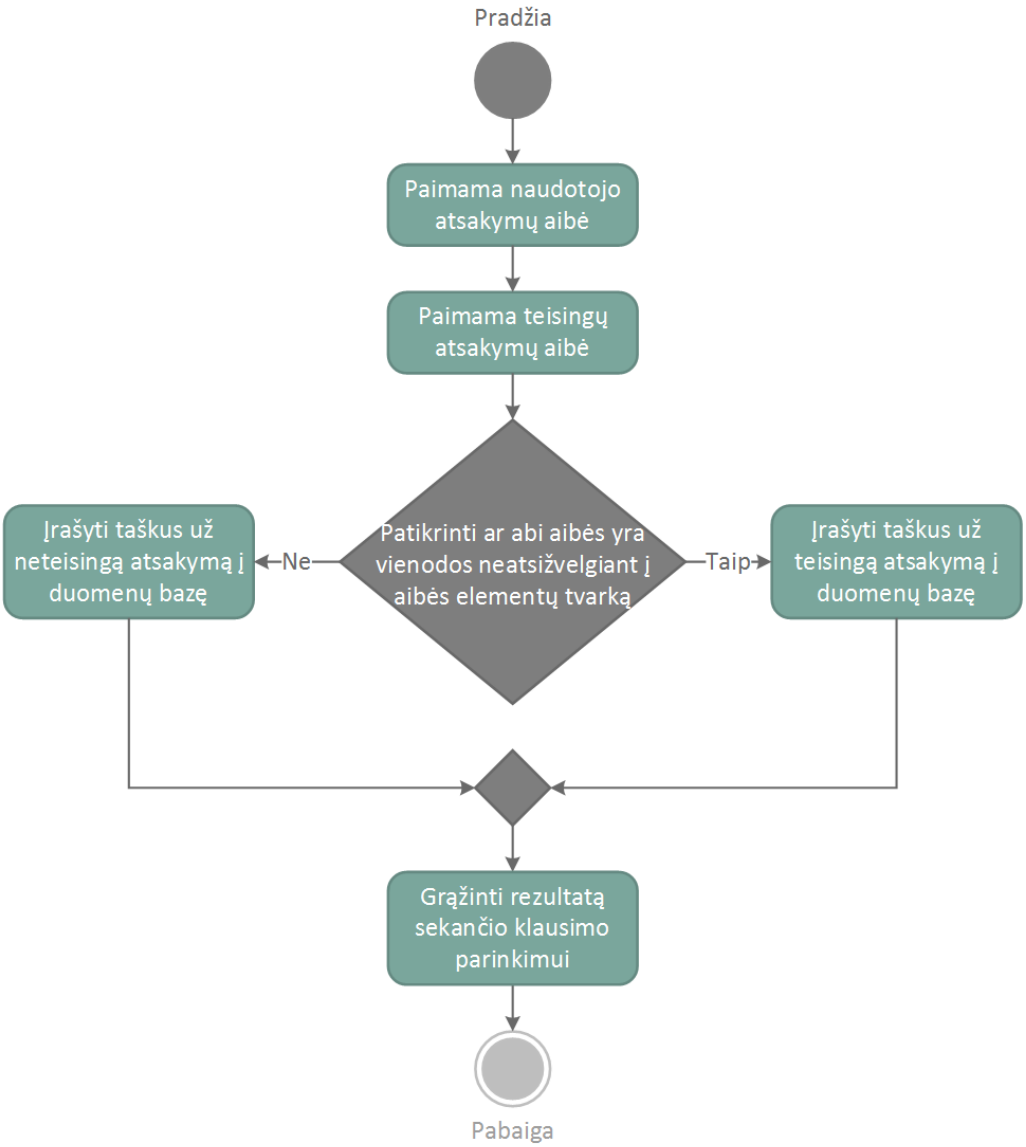
Sekančiame žingsnyje sudėtingumas iš logitų paverčiamas ir suapvalinamas į klausimo sudėtingumo lygį.

```
$inverted = round(1 / ( 1 + exp( (-1 * $nextdiff) ) ), 2);

$difflevel = round($lowestlevel + ( $inverted *
($highestlevel - $lowestlevel) ));
```

3.1.8. Atsakymų tikrinimo realizavimas

Naudotojui pateikus atsakymą ar atsakymus į klausimą, kiekvieną kartą pagal žemiau pateiktą veiklos diagramą patikrinama, ar teisingai atsakyta į klausimą. Teisingi bei neteisingi rezultatai išsaugomi duomenų bazėje. Tikrinimas atliekamas naudotojo rezultatus ir teisingus atsakymus surašius į du atskirus masyvus. Toliau šie masyvai yra palyginami vienas su kitu, neatsižvelgiant į elementų išsidėstymą masyve. Jei, palyginus masyvus, grąžinamas nesutapusių elementų masyvas - atsakymas užskaitomas kaip neteisingas. Jei grąžinamas tuščias elementų masyvas – atsakymas teisingas. Šis tikrinimo metodas taikomas tiek tikrinant paprastus testus, tiek adaptyvius testus.



3.7 pav. Atsakymo tikrinimo veiklos diagrama

Realizuojant algoritmus ir jam reikalingas funkcijas, pastebėti trūkumai buvo ištaisyti jau realizacijos metu. Realizuotos visos reikalingos funkcijos. Testų, kuriuos sudaro 30 atsitiktinių klausimų įgyvendinimas, buvo daug lengvesnis, nei adaptyvaus testo. Norint realizuoti adaptyvaus testo algoritmą prireikė daug papildomų pagalbinių funkcijų, kurios atlieka svarbius skaičiavimus klausimo pateikime ir jo atsakymo vertinime. Pritaikius ir optimizavus algoritmus testo kūrimui, buvo įgyta daug patirties, kuriant ir realizuojant tokio tipo sistemas.

4. EKSPERIMENTINĖ DALIS

4.1. Testavimo planas

Sistemos funkcionalumas buvo įgyvendinamas nuosekliai, todėl testavimas vyko kartu tikrinant, kaip funkcijos reaguoja ir atsako į klaidingus, teisingus ar dalinai teisingus duomenis. Funkcionalumas išbandytas skirtingoms naudotojų grupėms priklausančiomis naudotojų anketomis arba visai neprisijungus prie sistemos.

Testuojant funkcionalumą, buvo patikrinamas duomenų pateikimo naudotojui korektiškumas. Tikrinama, ar duomenys yra atvaizduojami numatytoje naršyklės lango vietoje.

4.2. Algoritmų veikimo testavimas

Testuojant algoritmų realizavimą, paeiliui buvo įgyvendinamos ir testuojamos algoritmui reikalingos funkcijos ir tik po to – bendras algoritmo veikimas.

Algoritmo testavimui buvo kuriamos atskiros funkcijos, kurios patikrintų veikimą ir grąžintų rezultatus lentelėje. Kiekvienas testas buvo sudarytas iš 30 algoritmo sugeneruotų klausimų. Testo generavimas buvo atliktas 800 kartų. Klausimų bazę sudarė 1650 klausimų. Sistemos išvesti duomenys paversti į lentelę (žr. 1.2 lentelė) ir diagramas.

Prieš atliekant algoritmų testus, matematiškai buvo paskaičiuoti numatomi rezultatai. Pirmiausia pagal proporciją buvo paskaičiuota, kiek unikalių testo variantų galima sugeneruoti, jei duomenų bazėje 1650 klausimų, o testą sudaro 30 klausimų. Paskaičiavę pagal formulę (2) gauname 55 unikalius testo variantus.

$$\frac{\text{Klausimyno dydis}}{\text{Testo klausimų skaičius}} = \frac{1650}{30} = 55; \quad (2)$$

Kadangi testas atliekamas 800 kartų, paskaičiuojame pagal formulę (3), kiek kartų turėtų būti pateiktas kiekvienas klausimas. Kai buvo žinoma, kokius rezultatus turėtų grąžinti testuojami algoritmai, buvo pradėtas sistemos algoritmų testavimas.

$$\frac{\text{Unikalūs testai} \times \text{Generojamų testų kiekis}}{\text{Testo klausimų skaičius}} = \frac{55 \times 800}{30} = 14{,}5454; \quad (3)$$

Adaptyvus algoritmas nebuvo įtrauktas į palyginimus, nes jo veikimo principas visiškai skirtingas. Sprendžiant adaptyvius testus, atsižvelgiant į naudotojo gabumų lygį, per lengvi ar per sunkūs klausimai gali būti nepateikti ar pateikti rečiau, nei kiti. Adaptyvaus testavimo algoritmas yra labiau pritaikomas egzamino tipo testams, nei mokomiesiems.

Lentelėje (žr. 4.1 lentelė) pateikiami gauti rezultatai, neįtraukiant adaptyvaus testo algoritmo.

Minimalus skaičius – kiek mažiausiai kartų pateiktas klausimas iš visos klausimų duomenų bazės. Minimali reikšmė turėtų būti kuo artimesnė vidutinei klausimo pateikimo reikšmei.

Maksimalus skaičius – kiek daugiausiai kartų pateiktas klausimas iš visos duomenų bazės. Šis skaičius taip pat turėtų būti kuo artimesnis vidutinei klausimo pateikimo reikšmei.

Vidutinis skaičius – vidutinis klausimo pateikimo skaičius. Tikimasi, kad gautas rezultatas bus panašus į apskaičiuotą pagal formulę (3).

Skirtumas – tai skirtumas tarp mažiausios ir didžiausios reikšmių. Šis skaičius parodo, kaip efektyviai ir (arba) optimizuotai veikia testuojamas algoritmas. Didelė šio skaičiaus reikšmė parodo, kad klausimai pateikiami nevienodai ir algoritmas netinkamas optimizuotam testų generavimui.

Pagal gautus rezultatus (žr. 4.1 lentelė) matome, kad didžiausią skirtumą turi testai, kurie buvo generuoti Mersenne Twister algoritmo pagalba. Mažiausią skirtumą turi testai, kurie buvo generuoti panaudojant Fisher – Yates algoritmą, kurio skirtumas vos 1 klausimas. Palyginus rezultatus galime teigti, kad Fisher – Yates algoritmas yra daug efektyvesnis.

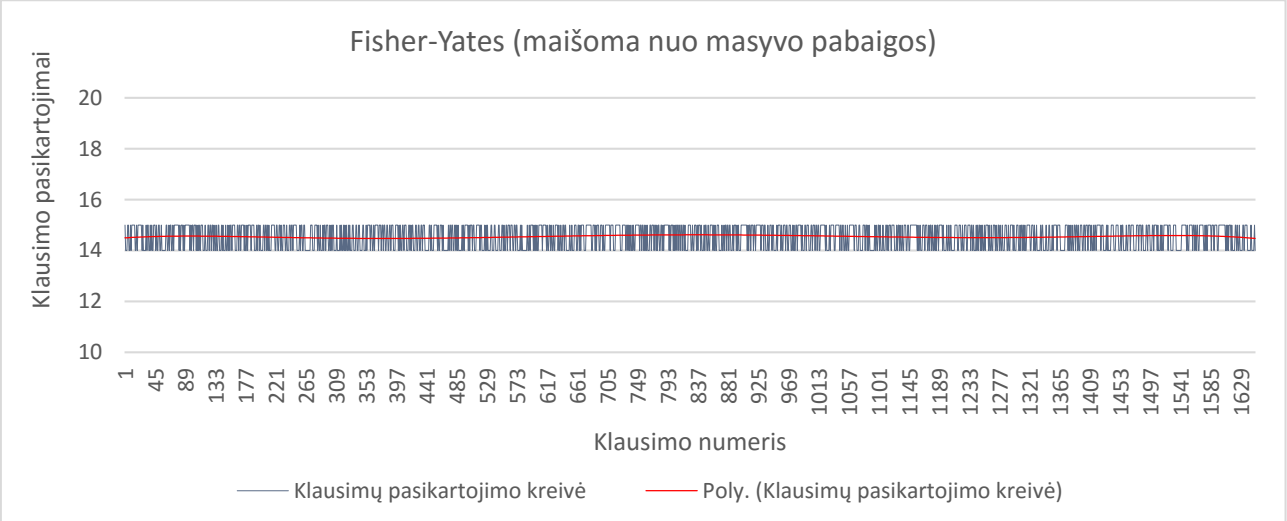
4.1 lentelė. Algoritmo rezultatų palyginimas

	Minimalus skaičius	Maksimalus skaičius	Vidutinis skaičius	Skirtumas
Fisher - Yates (maišoma nuo pabaigos)	14	15	14,54	1
Fisher - Yates (maišoma nuo pradžios)	14	15	14,54	1
Aibės generavimas Mersenne Twister algoritmu	3	30	14,54	27
Aibės generavimas panaudojant PHP funkciją array_rand()	5	29	14,539	24

4.2.1. Testo generavimas Fisher-yates (Knuth) algoritmu

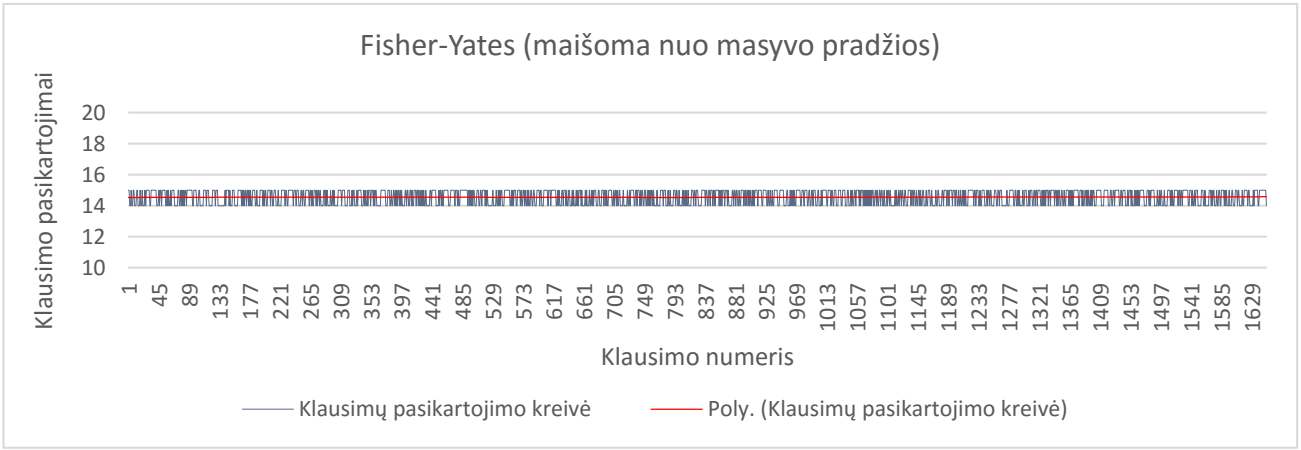
Algoritmas buvo realizuotas dviem variantais. Pirmu atveju pradedama nuo paskutinio pateikto klausimyno masyvo elemento. Sekantys elementai parenkami vienetu mažesni, kol pasiekiamas pirmasis elementas.

Sugeneravus testus su pirmuoju algoritmo variantu, matome (žr. 4.1 pav.), kad kiekvienas klausimas buvo pateiktas 14 arba 15 kartų. Toks rezultatas gautas todėl, kad klausimų duomenų bazės masyvas yra išmaišomas ir klausimai paimami 30 skaičių intervalais. Kaip matome iš raudonos kryptingumo linijos, sisteminės paklaidos nėra, nes linija eina tolygiai.



4.1 pav. Klausimų pasikartojimo dažnis Fisher - Yates algoritmu (maišoma nuo masyvo pabaigos

Antruoju atveju, kai masyvas maišomas nuo pradžios, algoritmas pateikia analogiškus rezultatus, matomus žemiau paveiksle (žr. 4.2 pav.). Kryptingumo linija tolygi, todėl galima teigti, kad algoritmas veikia gerai, tiek maišant nuo pabaigos, tiek maišant nuo masyvo pradžios.

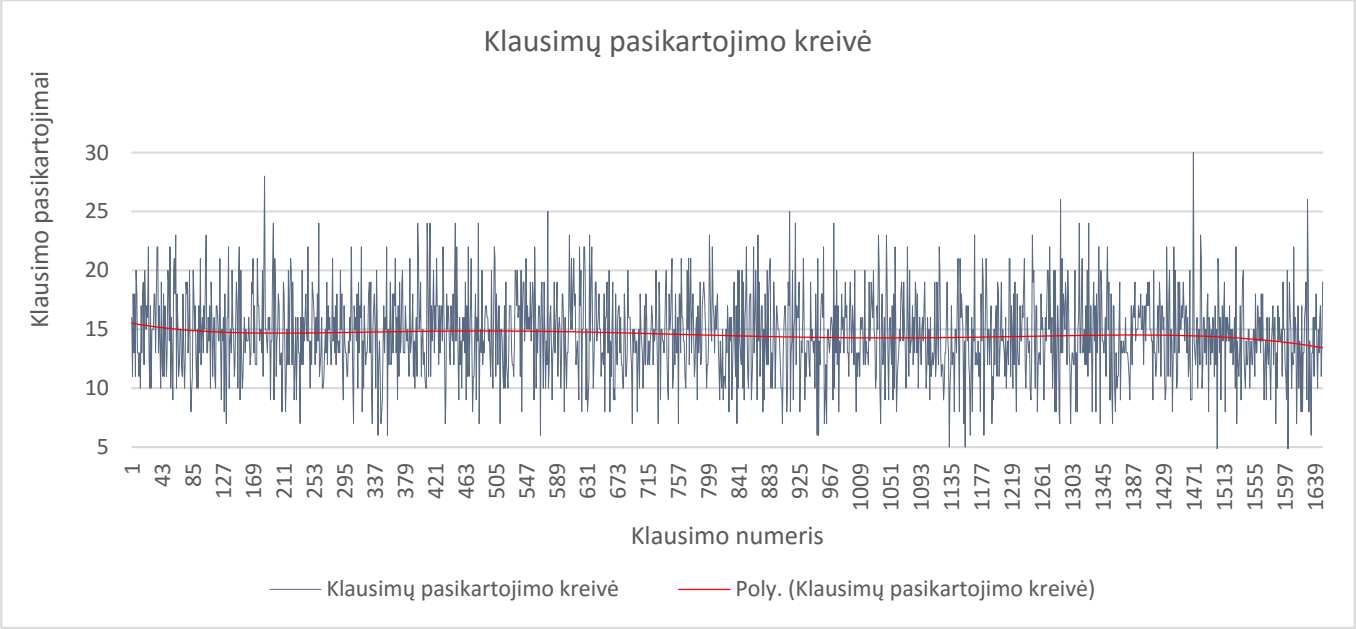


4.2 pav. Klausimų pasikartojimo dažnis Fisher - Yates algoritmu (maišoma nuo masyvo pradžios)

4.2.2. Testo generavimas panaudojant Mersenne Twister sugeneruotą unikalią skaičių aibę

Kaip matoma diagramoje (žr. 4.3 pav.), algoritmui, atsitiktinai parenkant elementus iš viso klausimyno masyvo, sudaromi testai nėra optimizuoti. Vieni klausimai paimami net 30 kartų, o kiti vos 3 kartus. Taip pat, atlikus šiuos testus, galime įrodyti, kad algoritmas turi sisteminę paklaidą. Tai gerai matoma iš kryptingumo linijos. Pirmieji duomenų bazės klausimai vidutiniškai paimami daugiau

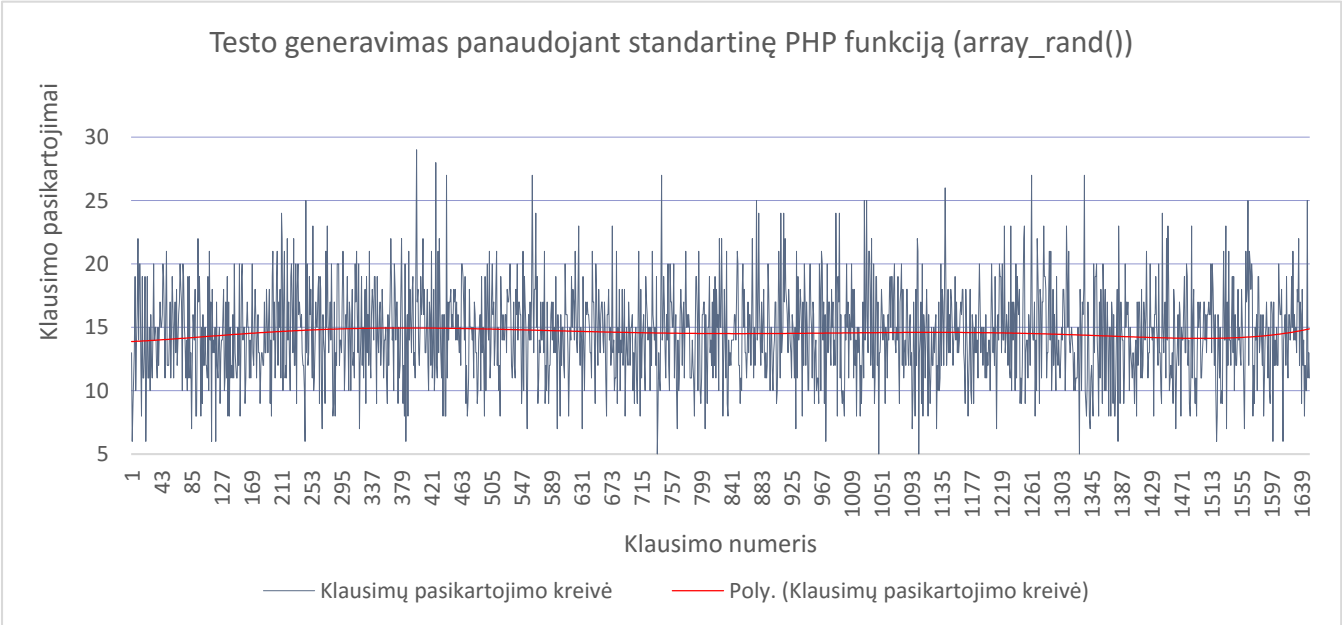
kartų – linija pradžioje pakilusi virš vidutiniškai pateikiamų klausimų reikšmės ir leidžiasi, artėjant link vidurio. Klausimai, esantys duomenų bazės pabaigoje, paimami rečiau. Tai parodo linijos leidimasis žemyn nuo vidutinės klausimo pateikimo reikšmės.



4.3 pav. Klausimų pasikartojimo dažnis Mersenne Twister algoritmu

4.2.3. Testo generavimas panaudojant standartinę PHP funkciją (array_rand())

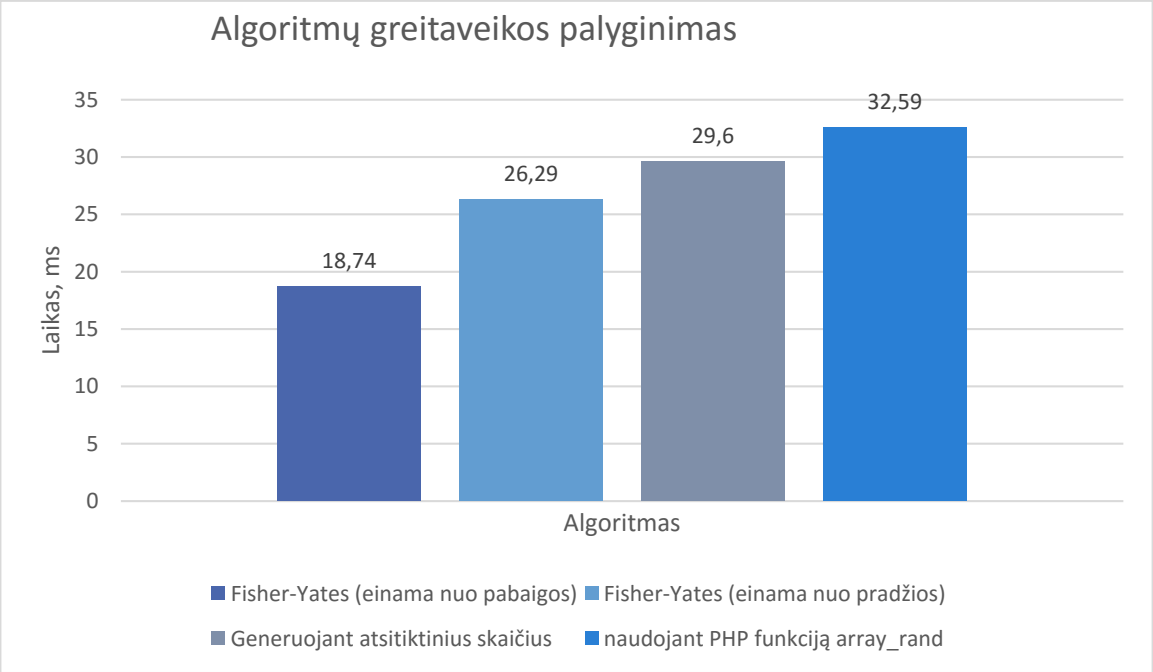
Kaip matoma diagramoje (žr. 4.4 pav.), algoritmui, atsitiktinai parenkant nurodytą elementų skaičių iš viso klausimyno masyvo, sudaromi testai nėra optimizuoti. Vieni klausimai paimami net 29 kartų, o kiti vos 5 kartus. Naudojant šį algoritmą, taip pat pastebima sisteminė paklaida. Tai gerai matoma iš kryptingumo linijos. Pirmieji klausimų duomenų bazės klausimai vidutiniškai paimami mažiau kartų, nei klausimai, esantys duomenų bazės gale.



4.4 pav. Klausimų pasikartojimo dažnis panaudojant PHP funkciją array_rand()

4.2.4. Testo generavimo greitaveikos palyginimas

Realizavus visus 4 prieš tai aprašytus algoritmus, kurie sugeneruoja 30 atsitiktinių testo klausimų, buvo palyginta jų veikimo greitaveika, kurią matome diagramoje (4.5 pav.). Rezultate parodomas vidutinis matuotas laikas, atlikus 10 testų. Vienam testui kiekvienas algoritmas turėjo sugeneruoti 800 sprendžiamų testo variantų. Toks sprendžiamų testų kiekis buvo imtas todėl, kad generuojant didesnį kiekį, skirtumai tarp algoritmų darbo laikų labiau išsiskiria. Kaip matome, greičiausiai veikia Fisher-Yates algoritmas, kai pradedama masyvą maišyti nuo pabaigos. Jis 800 testo variantų sugeneruoja per 18,74 milisekundės. Lėčiausiai veikiantis algoritmas buvo naudojant standartinę PHP funkciją array_rand(). Algoritmas truko 32,59 milisekundės, kol sugeneravo tokį patį variantų skaičių. Fisher – Yates algoritmas net 42,5 % greitesnis ir labiau optimizuotas, nei naudojat lėčiausią array_rand() metodu paremtą algoritmą.



4.5 pav. Algoritmų greitaveikos palyginimas

Atlikus testus ir palyginus rezultatus, buvo nuspręsta, kuris iš algoritmų toliau bus naudojamas testavimo sistemoje. Buvo įrodyta, kad Fisher – Yates algoritmas yra greičiausias ir labiausiai optimizuotas, generuojant klausimų aibes. Taip pat pastebėta, kad algoritmas gali būti pritaikytas ir kitose srityje. Realizuojant tą patį algoritmą keliais variantais, pastebima, kad didelę reikšmę turi pats realizacijos įgyvendinimas ir naudojamos funkcijos.

IŠVADOS

1. Atlikus esamų sistemų analizę sužinota apie bendrą testų sistemų funkcionalumą. Sistemos išbandytos įvairaus dydžio įrenginiuose. Ne visos sistemos buvo pritaikytos atvaizduoti tiek kompiuteryje, tiek telefone. Išbandžius sistemas pakartotinai, buvo patvirtinta, kad klausimai kartojasi. Pastebėta, jog bandomosios sistemų versijos turi labai ribotą klausimyną, todėl klausimai galėjo dažnai kartotis. Pilnos sistemos turi nuo 2 iki 5 kartų daugiau klausimų pagal pateiktus aprašymus, todėl klausimų pasikartojimo tikimybė mažesnė.
2. Išanalizavus rastus algoritmus, kurie tinka klausimyno sudarymui, sužinota, jog kai kurie algoritmai turi blogą savybę – sisteminę paklaidą. Dėl to negalima tinkamai panaudoti šių algoritmų, kuriant unikalias klausimų aibes. Išsiaiškinta, kurie algoritmai galėtų būti pritaikyti klausimyno generavimui. Taip pat, atlikus algoritmų analizę, išsiaiškinta, jog, norint sukurti gerai funkcionuojančią sistemą, reikės apjungti kelis algoritmus į bendrą visumą.
3. Atlikta programavimo kalbų, naudojamų tokio tipo testavimo sistemų kūrimui, apžvalga parodė, kad nauji HTML5 ir CSS3 standartai ir jų galimybės leidžia daug greičiau ir efektyviau įgyvendinti algoritmų realizaciją bei pritaikyti kuriamai sistemai. Išsiaiškinus pagrindinių programavimo kalbų privalumus ir trūkumus, buvo pasirinkta PHP programavimo kalba, PHP internetinių aplikacijų kūrimo karkasas *Laravel 5*.
4. Realizuoti algoritmus pavyko ne iš karto. Realizuojant ir testuojant jų veikimą pastebėti trūkumai ištaisyti realizacijos pradžioje. Vėliau, tikrinant greitaveiką, pastebėta, kad panaudotos programavimo kalbos suteikiamos standartines funkcijos lėtina algoritmų veikimą, todėl buvo reikalingi esminiai realizacijos pakeitimai. Ištaisius šiuos trūkumus, klausimų generavimas tapo greitas ir atitiko suprojektuotą veikimą.
5. Pirmiausia buvo realizuotas ir ištestuotas testų, klausimų, ir kategorijų administravimas. Testavimo procese pastebėti trūkumai ištaisyti. Vėliau testuojami klausimų pateikimo algoritmai ir jų greitaveika. Rezultatai parodė, kad Fisher – Yates algoritmas ne tik efektyviai generuoja klausimų aibes, bet ir 42,5 % veikia greičiau, nei lėčiausias standartinę PHP funkciją naudojantis algoritmas. Nuspręsta, kad, generuojant 30 klausimų testus, klausimų generavimui bus pasirinktas Fisher-Yates algoritmas, o adaptyvus klausimų algoritmas bus pasirinktas egzamino tipo testuose.

LITERATŪRA

1. EUGENIJUS TELEŠIUS, RENATA DANIELIENĖ. *Internetinės ECDL testavimo sistemos inovatyvūs sprendimai.* [Tinkle] 2009 m. http://www.zurnalai.vu.lt/informacijos-mokslai/article/viewFile/3229/2346.

2. UAB "ARS LEIDINIAI". *K.E.T programa*. [Tinkle] 2009-2015 m. [Cituota: 2014 m. 11 21 d.] http://www.ketprograma.lt/.

3. AUTOTESTAI. *AUTOTESTAI.LT*. [Tinkle] 2013 m. [Cituota: 2014 m. 11 20 d.] http://www.autotestai.lt/Demonstracinine-versija.html.

4. UAB "ALS". *KET teorijos mokymosi testai.* [Tinkle] [Cituota: 2014 m. 11 21 d.] http://www.alsket.lt/.

5. DRIVINGED. *Kaip vyksta egzaminai Regitroje*. [Tinkle] DrivingEd, 2008-2015 m. [Cituota: 2014 m. 11 20 d.] http://www.ketbilietai.lt/main/Puslapis/egzaminai_regitroje.

6. DATAGENETICS. *Shuffling.* [Tinkle] DataGenetics, 2014 m. nenurodytas nenurodyta d. [Cituota: 2015 m. 11 20 d.] http://datagenetics.com/blog/november42014/index.html.

7. BOSTOCK, MIKE. *Will It Shuffle?* [Tinkle] 2012 m. 01 21 d. [Cituota: 2015 m. 11 20 d.] https://bost.ocks.org/mike/shuffle/compare.html.

8. INFO WEB S.R.O. *Fisher-Yates shuffle.* [Tinkle] INFO WEB s.r.o., 2015 m. nenurodytas nenurodytas d. [Cituota: 2016 m. 02 26 d.] http://www.programming-algorithms.net/article/43676/Fisher-Yates-shuffle.

9. YU, CHONG HO. *A simple Guide to the Item Response Theory (IRT) and Rasch Modeling.* [Tinkle] 2013 m. 06 20 d. [Cituota: 2016 m. 02 26 d.] http://www.creative-wisdom.com/computer/sas/IRT.pdf.

10. ASSESMENT SYSTEMS. *Computerized Adaptive Testing:The Future of Assessment.* [Tinkle] Assesment systems, nenurodyta m. [Cituota: 2016 m. 02 24 d.] http://www.assess.com/adaptive-testing/.

11. LINACRE, JOHN MICHAEL. *Institute for Objective Measurement.* [Tinkle] [Cituota: 2016 m. 02 15 d.] http://www.rasch.org/memo69.pdf.

12. ANGELOV, MARTIN. *12 Awesome CSS3 Features That You Can Finally Start Using.* [Tinkle] 2013 m. 10 25 d. [Cituota: 2015 m. 04 10 d.] http://tutorialzine.com/2013/10/12-awesome-css3-features-you-can-finally-use/.

13. OTWELL, TAYLOR. *Blade Templates.* [Tinkle] nežinomas m. [Cituota: 2015 m. 12 15 d.] https://laravel.com/docs/5.2/blade.

14. SKVORC, BRUNO. *Best PHP Framework for 2015 – SitePoint Survey Results.* [Tinkle] 2015 m. 03 28 d. [Cituota: 2015 m. 04 05 d.] http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/.

15. LARAVEL. *Laravel 5* . [Tinkle] http://laravel.com/docs/5.1.

16. JeffD, chrisdavidmills, markg, brittanystoroz, kscarfone. *MVC architecture*. [Tinkle] 2015 m. 04 3 d. [Cituota: 2015 m. 11 20 d.] https://developer.mozilla.org/en-US/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture.

17. NEDRICH, MATT. *Fisher-Yates Shuffle – An Algorithm Every Developer Should Know.* [Tinkle] 2014 m. 06 11 d. [Cituota: 2016 m. 04 20 d.] https://spin.atomicobject.com/2014/08/11/fisher-yates-shuffle-randomization-algorithm/.

UAB „SDG akademija“
Draugystės g. 8E, LT- 51290 Kaunas, tel. +370 650 22225, el. p. akademija@sdg.lt, www.sdgakademija.lt
Duomenys kaupiami ir saugomi Juridinių asmenų registre, kodas 136036323, PVM mokėtojo kodas LT100007855918

PAŽYMA DĖL SISTEMOS DIEGIMO

2016 m. gegužės 23 d.

Kaunas

Patvirtinu, kad Aidas Knistautas 2016 m. vasario mėn. pradėjo diegti Kelių eismo taisyklių teorijos testų sprendimo sistemą mūsų vairavimo mokykloje. Sistema ir yra pildoma ir testuojama iki dabar. Numatoma, kad ši sistema ateityje pakeis esamą testų sprendimo sistemą.

Direktorė Karolina Žekaitė