VILNIAUS UNIVERSITETAS

Bronius Skūpas

PUSIAU AUTOMATINIO PROGRAMAVIMO UŽDUOČIŲ VERTINIMO IR TESTAVIMO METODAS

Daktaro disertacija

Technologijos mokslai, Informatikos inžinerija (07 T)

Vilnius, 2013

Disertacija rengta 2008—2012 metais Vilniaus universiteto Matematikos ir informatikos institute.

Mokslinė vadovė:

prof. dr. Valentina Dagienė. (Vilniaus universitetas, technologijos mokslai, informatikos inžinerija – 07 T).

Padėka

Nuoširdžiai dėkoju mokslinei vadovei prof. dr. Valentinai Dagienei už vadovavimą, vertingas mokslines konsultacijas, pasitikėjimą manimi, nuolatinį padrąsinimą, kantrybę, neįkainuojamus patarimus ir pagalbą.

Esu labai dėkingas darbo recenzentams prof. dr. Dalei Dzemydienei ir doc. dr. Gintautui Grigui, prof. dr. Albertui Čaplinskui, doc. dr. Linai Kankevičienei už vertingas pastabas, padėjusias tobulinti šį darbą.

Nuoširdus ačiū dr. Tom Verhoeff (Technische Universiteit Eindhoven, Nyderlandų karalystė) už anglų kalbos pastabas ir padrąsinimus.

Taip pat esu dėkingas Sutinen šeimai ir dr. Jarkko Suhonen, nuoširdžiai globojusiems mane ERASMUS programos mainuose Rytų Suomijos universitete, Joensuu.

Dėkoju Vilniaus universiteto Matematikos ir informatikos instituto Informatikos metodologijos skyriaus ir Vilniaus licėjaus kolegoms už paramą.

Esu labai dėkingas Jūratei, mano žmonai, už jos kantrybę tyrimų ir disertacijos rašymo metu, o taip pat ir vertingas dalykines pastabas.

Ačiū šeimai už moralinį palaikymą, skatinimą ir supratimą.

Bronius Skūpas

Santrauka

Šiuolaikinėje visuomenėje auga informacinių technologijų ir informatikos specialistų poreikis. Didelė šių specialistų dalis turi mokėti programuoti. Todėl dauguma mokyklą baigiančių mokinių turi turėti bent minimalias kompiuterių programavimo žinias. Žymi dalis universitetų techninių, gamtos, bei ekonominių specialybių taip pat turi išklausyti programavimo kursą.

Programavimo mokymas yra sudėtingas procesas, nes jis sietinas ir su kūrybišku mąstymu, ir su griežtai formalizuotomis užduotimis, praktiniais darbais. Mokiniai ir studentai turi būti mokomi kurti ne tik veikiančias, bet ir kokybiškai suprojektuotas, patikimas, tvarkingas programas. Mokinių, sudentų sukurtų programų patikrinimas bei įvertinimas reikalauja daug mokytojo ar dėstytojo pastangų ir laiko.

Norint palengvint mokytojų darbą yra kuriamos automatinės programų testavimo sistemos, kurios grindžiamos statine bei dinamine mokinių programų analize. Dauguma jų remiasi *juodosios dėžės* testavimo principu. Nors šis principas kritikuojamas labai seniai, bet gerų alternatyvų nėra. Norint kompensuoti šio principo trūkumus sistemos papildomos įvairiais papildomais moduliais, diegiamas dalinis rankinis vertinimas. Automatinio programų testavimo ir rankinio programų vertinimo sujungimas leidžia panaudoti abiejų metodų gerąsias savybes. Šis programų vertinimo būdas dažnai vadinamas pusiau automatiniu programų vertinimu ir testavimu.

Disertacijos rengimo metu buvo tirti metodai, taikytini abejais vertinimo būdais gautų rezultatų apjungimui, taip pat buvo gilinamasi į pusiau automatinio vertinimo ir testavimo metodo automatinės ir rankinės

dalių sąveiką. Buvo sukurta hipotezė, kad galima sukurti naują, patobulintą pusiau automatinio vertinimo ir testavimo metodą, kuris leistų pasiekti didesnę programų vertinimo spartą neprarandant rankinio vertinimo kokybės.

Darbe pristatomas konstruktyvus pasiūlymas kaip tobulinti pusiau automatinio vertinimo ir testavimo metodą, jį diegti į kitas pusiau automatinio testavimo sistemas, jas papildant naujais komponentais, pakeičiant vertintojo ir sistemos sąveiką. Siūlomi papildomi programos teksto redagavimo, programų pakeitimų identifikavimo komponentai yra nesunkiai adaptuojami ir įterpiami į sistemas, nes analogiški komponentai yra naudojami įvairiuose atvirojo kodo projektuose. Modifikuotas dviejų pakopų programos pateiktų atsakymų teisingumo vertinimo komponentas taip pat nesunkiai įdiegiamas vietoje įprastinio vienos pakopos vertinimo komponento. Pakeistas vertintojo sąveikos su sistema būdas yra natūralus, artimas įprastam mokytojo darbo procesui, taip pat nekeliantis diegimo į kitas sistemas problemų. Žymesni sistemų pakeitimai sietini su testavimo bei vertinimo informacijos išsaugojimu.

Disertacijoje pristatomos šiuolaikinių automatinių ir pusiau automatinių programų testavimo sistemų ypatybės, pristatomas patobulintas pusiau automatinio testavimo sistemos metodas, demonstruojami jo tyrimo konstravimu rezultatai – pagrindžiamas naujojoj metodo efektyvumas ir jo pagalba gautų įvertinimų kokybė. Metodas išbandytas jį taikant realiame gyvenime – Lietuvos informacinių technologijų valstybinio brandos egzamino (IT VBE) praktinių užduočių vertinime.

Turinys

Iliustracijų sąrašas

Lentelių sąrašas

Terminų ir santrumpų žodynėlis

„Viskas arba nieko grupėms“ agregavimo funkcija Testavimo rezultatų agregavimo būdas, kai tik įveikus visus testus iš vienos grupės gaunamas grupei priskirtas ne nulinis įvertis. Grupių įverčiai juos agreguojant yra sumuojami.

„Viskas arba nieko“ agregavimo funkcija Testavimo rezultatų agregavimo būdas, kai tik įveikus *visus* testus gaunamas ne nulinis įvertis.

ACM-ICPC Tarptautinės aukštųjų mokyklų programavimo varžybos. ACM organizuojamos programavimo varžybos universitetų komandoms. Komanda naudojasi vienu kompiuteriu, rezultatams agreguoti naudojama „viskas arba nieko“ agregavimo funkcija.

Agregavimo funkcija Matematinė funkcija apibendrinanti duomenų rinkinį į vieną rezultatą. Disertacijoje agregavimo funkcijos taikomos testavimo rezultatams ir kitiems įvertinimams agreguoti. Žr. *testavimo rezultatų agregavimo funkcija*.

Automatinis programų testavimas Mokinių parašytų programų teisingumo tikrinimas pagal iš anksto parinktus vertinimo kriterijus bei vykdant programą su parinktais duomenų rinkiniais.

Dalinio agregavimo funkcija Testavimo rezultatų agregavimo būdas, kai kiekvienam testui priskirtas atskiras įvertis. Šie įverčiai juos agreguojant yra sumuojami.

Dinaminė analizė Programos analizė, atliekama vykdant programoje užrašytas komandas. Vienas labiausiai paplitusių dinaminės analizės pavyzdžių – programų testavimas juodosios dėžės metodu. Alternatyva dinaminei analizei – žr. *statinė analizė*

Efektyvumo testai Testiniai pradinių duomenų ir teisingų atsakymų rinkiniai, skirti įvertinti testuojamos mokinio programos tinkamumą atlikti užduoties sąlygoje suformuluotus veiksmus neviršijant nurodytų vykdymo laiko bei naudojamos atmintinės apribojimų.

Informacinių technologijų valstybinis brandos egzaminas (IT VBE) Laisvai pasirenkamas abitūros egzaminas. Jo viena sudedamųjų dalių yra praktinės programavimo užduotys.

IOI Žr. *Tarptautinė informatikos olimpiada*.

IT VBE Žr. *Informacinių technologijų valstybinis brandos egzaminas*.

IT VBE praktinių užduočių vertinimo sistema (IT VBE PUVS) Pusiau automatiniu testavimu grįsta Lietuvos informacinių technologijų valstybinio brandos egzamino praktinių užduočių vertinimo sistema. Pradėta kurti 2005 m. Nuosekliai tobulinama.

IT VBE PUVS Žr. *IT VBE praktinių užduočių vertinimo sistema.*

Juodosios dėžės principas Principas, kuriuo remiantis įrenginio ar programos veikimas analizuojamas tik iš jos veikimo rezultatų, nežiūrint į vidinę sandarą ar veikimo principą.

Kriterinis vertinimas Vertinimas, kurio pagrindas – tam tikri kriterijai (pvz., standartai), su kuriais lyginami užduotį sprendusiųjų pasiekimai.

Lietuvos mokinių informatikos olimpiada (LMIO) Kasmetinės nacionalinės individualios mokinių programavimo varžybos, vykstančios pagal panašias į tarptautinių informatikos olimpadų taisykles. Vyksta keliais turais mokyklose. Olimpiados metu naudojamos saityno technologija paremtos programavimo varžybų aptarnavimo sistemos (PVAS).

LMIO Žr. *Lietuvos mokinių informatikos olimpiada.*

Mokinio programos efektyvumas Programos tinkamumas atlikti užduoties sąlygoje nurodytus veiksmus neviršijant leistų operatyviosios atmintinės bei vykdymo laiko ribų. Efektyvumas gali būti vertinamas panaudojant efektyvumo testus.

Nevieši testai Testiniai pradinių duomenų ir teisingų atsakymų rinkiniai, kurie naudojami mokinio programų testavimui, tačiau jų turinys atskleidžiamas mokiniui tik pasibaigus pamokai, varžyboms ar egzaminui.

Norminis vertinimas Vertinimas, kuris sudaro galimybes palyginti užduotis sprendusiųjų pasiekimus.

OS Kompiuterio operacinė sistema. Operacinės sistemos turi įtakos nepabaigtų programų vykdymo klaidoms.

Paketinė užduotis Užduotis, kurios sprendimas yra programa, skaitanti iš anksto paruoštus duomenis bei suformuojanti nurodyto formato rezultatus.

Pateikta programa Mokinio, studento ar varžybų dalyvio sukurta programa, kuri buvo sukurta atsižvelgiant į užduoties reikalavimus. Programos tekste gali būti ir kitos informacijos nei vien programavimo kalbos komandos. Pavyzdžiui programos tekste gali būti komentarai, o komentaruose užrašyta sprendimo idėja, informacija apie autorių, naudotą programavimo kalbą ir pan.

Pavyzdiniai testai Testiniai pradinių duomenų ir teisingų atsakymų rinkiniai, kurie pateikiami mokiniui programos kūrimo metu, kad programos autorius galėtų įsitikinti, kad teisingai suprato užduoties sąlygą, kad programa funkcionuoja su nesunkiai patikrinamais duomenimis.

Programavimo užduoties sprendimo įvertinimas Programavimo užduočių vertinimo proceso rezultatas, konkretus sprendimas apie užduotį sprendusiojo pademonstruotus pasiekimus bei palyginimas su kitų sprendusiųjų pasiekimais.

Programavimo užduočių vertinimas Informacijos apie programavimo užduoties sprendimų atitikimą keliamiems reikalavimams kaupimas, interpretavimas ir apibendrinimas.

Programavimo varžybų aptarnavimo sistema (PVAS) Įvairių serverinių programų ir modulių rinkinys, teikiantis paslaugas varžybų dalyviams, vertintojams ir administratoriams. Viena dažniausiai teikiamų PVAS paslaugų – automatinis programų testavimas

Programų testavimas juodosios dėžės metodu Programos testavimas su iš anksto parinktais duomenimis, tikintis gauti galimus patikrinti rezultatus. Išvados apie programos kokybę daromos tik iš išoriškai pamatuojamų vykdymo laiko, panaudotos OS atmintinės bei rezultatų teisingumo. Tai yra vienas iš juodosios dėžės principo panaudojimo būdų.

Pusiau automatinis programų testavimas Testavimas, kurio metu panaudojamos tiek automatinio programų testavimo, tiek rankinio programų vertinimo galimybės.

PUVS Praktinių (programavimo) užduočių vertinimo sistema.

PVAS Programavimo varžybų aptarnavimo sistema.

Rankinis programų vertinimas Vertinimas, kurį atlieka žmogus (mokytojas).

Statinė analizė Programos analizė atliekama analizuojant programoje užrašytas komandas, bet jų nevykdant. Populiariausi statinės analizės pavyzdžiai – sintaksės tikrinimas, plagiato paieška, teksto išdėstymo analizė ir jo pertvarkymas. Alternatyva statinei analizei – žr. *dinaminė analizė*.

Tarptautinė informatikos olimpiada (IOI) UNESCO iniciatyva vykstančios tarptautinės individualios mokinių programavimo varžybos, vykstančios kasmet akivaizdiniu būdu. Mokinių programos testuojamos juodosios dėžės metodu.

Teisingumo testai Testiniai pradinių duomenų ir teisingų atsakymų rinkiniai, skirti įvertinti testuojamos programos tinkamumą atlikti užduoties sąlygoje suformuluotus veiksmus

Testas Programai parengtas pradinių duomenų ir galimų teisingų atsakymų rinkinys. Dažniausiai yra užtikrinamas pradinių duomenų formato atitikimas užduoties sąlygoje nurodytam formatui.

Testavimo rezultatų agregavimo funkcija Matematinė funkcija, kuri pritaikoma apjungiant testavimo rezultatus. Apibrėžiant šią funkciją nurodomi testinių rinkinių svoriai, jų įtaka galutiniam balui.

Testų grupė Testų aibė, kurios tikslas patikrinti specifinį užduoties reikalavimą programai ar dažnai pasitaikančią programavimo klaidą.

Užduotis Tiksliai suformuluota programavimo užduotis, kurioje yra aiškiai apibrėžtas santykis tarp pradinių duomenų ir rezultatų. Dažniausiai užduoties sprendimas remiasi programos sukūrimu ir aprašymu. Mokymui skirtose užduotyse gali būti nurodomas algoritmas, kuriuo reikia pasinaudoti, tačiau varžybose algoritmą bei sprendimui reikalingas duomenų struktūras dažniausiai turi pasirinkti pats mokinys.

Vertinimo kokybė Vertinimas turi tenkinti patikimumo ir pilno vertinimo skalės išnaudojimo reikalavimus, t.y. vertinant didesnį kiekį užduoties sprendimų turi būti gaunami visi galimi skirtingi įvertinimai.

Vertinimo patikimumas Vertinimo tikrumas (neprieštaringumas), vertinimo rezultatų pastovumas: kai tas pats užduoties sprendimas tomis pačiomis sąlygomis gauna tokį pat įvertinimą; kai galima patikimai palyginti užduotį sprendusiųjų pasiekimus tarpusavyje (norminis vertinimas) arba užduotį sprendusiųjų pasiekimus su nustatytais kriterijais (kriterinis vertinimas).

Vertinimo schema Užduoties autoriaus paqrinktas testų bei vertinimo kriterijų rinkinys, bei jų įverčių agregavimo funkcija.

1 Įvadas

1.1 Darbo aktualumas

Kompiuteriai ir kita technologinė įranga naudojama visose visuomenės veiklos srityse. Programinės įrangos kūrimo tempai spartėja ir tam reikalingi vis nauji informacinių technologijų (IT) profesionalai: projektuotojai, programuotojai, testuotojai. Šie būsimi darbuotojai pradedami rengti mokyklose, kur jie supažindinami su IT pradmenimis, programavimo pagrindais. Jie taip pat mokosi savarankiškai, tęsia mokslus aukštosiose mokyklose.

Programavimo mokymas – sudėtingas procesas. Jis sietinas ir su kūrybiniu mąstymu, ir su griežtai formalizuojamomis užduotimis, praktiniais programavimo darbais. Mokiniai ir studentai (toliau mokiniai) turi būti mokomi kurti ne tik veikiančias, bet ir kokybiškai suprojektuotas, patikimas, teisingai veikiančias programas. Mokinių sukurtų programų patikrinimas bei įvertinimas reikalauja daug mokytojo ar dėstytojo pastangų ir laiko.

Siekiant galimybės daliai vertinimo kriterijų taikyti automatinį programų testavimą programavimo užduotys yra dažnai griežčiau specifikuojamos, tiksliai nurodoma, iš kokių duomenų ir kokius rezultaus reikia gauti, dažnai nurodomi darbo metodai ar dalis naudotinų algoritmų. Tai leidžia vertinimo metu taikyti faktų tikrinimo metodą. Tiksliai identifikuotas galimas patikrinti faktas vadinamas vertinimo kriterijumi.

Vertinimo kriterijų, tinkamų automatiniam testavimui, atpažinimas yra svarbus, nes jis leidžia vertinti darbus efektyviau. Taip pat tai leidžia sukurti mokymo metodus, kuriuose didžioji techninio darbo dalis atliekama automatizuotai ir mokytojas gali koncentruotis į svarbiausias mokymo

problemas, kurios reikalauja tiesioginio mokinio bendravimo su mokytoju [5].

Automatinis programavimo užduočių sprendimų testavimas aktualus ne vien tik mokymo proceso metu, bet ir per programavimo egzaminus, profesinius programavimo žinių patikrinimus, priimant į darbą, organizuojant programavimo varžybas.

Daug autorių pastebi, kad visiškai automatinis testavimas, besiremiantis statine bei dinamine analize, negali būti pilnai objektyvus. Todėl pasitelkiamas pusiau automatinis vertinimo ir testavimo metodas, kuris leidžia lanksčiau naudotis automatinio testavimo privalumais.

Darbe tiriamos galimybės naudoti pusiau automatino programavimo užduočių vertinimo ir testavimo metodą mokomojoje veikloje, egzaminuose. Analizuotos pasitaikančios vertinimo klaidos. Ieškota būdų pasinaudojant žinomais vertinimo procesais sukurti naują patobulintą metodą kokybiškesniam ir efektyvesniam vertinimui, aiškesniam vertinimo pagrindimui sukurti.

1.2 Tyrimų objektas

Automatinės ir pusiau automatinės programavimo užduočių testavimo sistemos, jų architektūra, veikimo metodai ir algoritmai, sąveika su vartotojais.

1.3 Darbo tikslas

Išanalizavus esamas automatines ir pusiau automatines programavimo užduočių vertinimo sistemas susisteminti architektūrinius sprendimus ir pasiūlyti sprendimus kaip patobulinti mokinio–sistemos ir sistemos–vertintojo sąveiką bei padidinti vertinimo efektyvumą ir kokybę.

1.4 Darbo uždaviniai

1. Atlikti automatinių ir pusiau automatinių programavimo užduočių testavimo sistemų teorinių ir eksperimentinių pasiekimų analizę.

2. Ištirti automatinių ir pusiau automatinių programavimo užduočių testavimo sistemų savybes, jų sąsajų su vartotojais ypatumus bei teikiamas paslaugas, išskirti ir susisteminti šių programų sistemų metodines ir architektūrines problemas.

3. Pasiūlyti patobulintą metodą pusiau automatiniam programavimo užduočių vertinimui ir testavimui, kuriuo remiantis būtų galima pagerinti darbų vertinimo kokybę bei efektyvumą.

4. Atlikti pasiūlyto metodo tyrimą konstravimu ir įvertinti metodo teikiamas naujas galimybes.

1.5 Tyrimų metodai

Disertacijoje naudojamos tyrimų metodikos pagrindą sudaro analitiniai, apibendrinamieji, konstruktyvieji ir vertinamieji metodai.

Darbe buvo susidurta su realaus pasaulio problema – buvo nežinoma, koks programavimo užduočių vertinimo metodas galėtų padėti įvertinti didelį kiekį pateiktų programavimo užduočių sprendimų per tenkinantį laiką ir su pakankama vertinimo kokybe. Tokio tipo problemoms dažnai rekomenduojamas tyrimas konstravimu.

Kasanen ir kt. [44] šiam metodui nurodo šešis žingsnius: (1) pasirenkama problema, reikalaujanti praktinio sprendimo; (2) įsigilinama į problemos mokslinę kryptį ir kontekstą, susisteminami stochastiški duomenys; (3) sukuriama aibė idėjų tobulinimui bei atnaujinimui – remiamasi euristine sprendimų paieška per daug nesigilinant į teorinį jų pagrindą; (4) eksperimentiškai įsitikinama, kad sukurtas sprendimas veikia; (5) ištiriamos teorinės patobulinimo pasekmės ir (6) išsiaiškinama galimybė taikyti naujus patobulinimus kitose srityse.

Tyrime buvo laikomasi šių žingsnių, tik ne visi jie sekė nurodyta tvarka. Darbo procesas buvo iteratyvus ir rekursyvus. Iš dalies tai susiję su

tuo, kad darbai buvo pradėti prieš septynis metus. Per šiuos metus keitėsi reikalavimai vertinimo metodui bei tyrimų kryptys.

Analitinėje disertacijos dalyje pristatoma atlikta literatūros, programinės įrangos, programavimo užduočių analizė. Ji buvo vykdoma kokybinio tyrimo metodais – taikyta informacijos paieška, sukaupta medžiaga apdorota lyginamosios analizės, apibendrinimo metodais. Šių tyrimų rezultatai leido įsigilinti į problemą, sukaupti galimų sprendimų, įrankių bei metodų rinkinį. Ši darbo dalis atitinką konstruktyviojo tyrimo (2) žingsnį.

Teorinėje disertacijos dalyje pateikiamas procesas, kaip buvo konstruojamas metodas – pristatytos idėjos ir jų eksperimentinių realizacijų prototipe pasekmės. Šioje disertacijos dalyje persipynė (3) ir (4) konstruktyviojo tyrimo žingsniai. Buvo tikrinama idėja, kad interaktyvus pusiau automatinis programavimo užduočių vertinimo metodas gali padėti išspręsti neefektyvaus, bet kokybiško rankinio vertinimo problemą – pakelti vertinimo efektyvumą. Atlikti pažingsniniai tobulinimai, todėl buvo svarbus eksperimentinis prielaidų tikrinimas, kuris rėmėsi interviu principu su vertinimo ekspertais.

Eksperimentinėje dalyje pateikiamas suformuluotų hipotezių tikrinimas (5 žingsnis) tiek kiekybiniais, tiek kokybiniais metodais. Kiekybiniai eksperimentiniai tyrimai buvo atliekami pasinaudojant turimais metodo taikymo realiam gyvenime duomenimis – metodas pritaikytas Lietuvos valstybinio brandos egzamino praktinių užduočių sprendimų vertinimui. Tačiau iškeltų hipotezių patikrinimui reikėjo palyginimo su etalonine vertinimo sparta ir vertinimo kokybe. Tam pagal suformuluotas taisykles atrinkti ekspertai-vertintojai, bei atrinkti užduočių sprendimai, pasirinktas eksperimentinio vertinimo laikas. Įvykdytas eksperimentas, atlikta ekspertų- vertintojų kokybinė apklausa. Šiuo būdu gauti etaloniniai rankinio vertinimo balai, laiko sąnaudų įverčiai. Duomenys leido įvertinti

plačiai įdiegto naujo vertinimo metodo efektyvumą ir kokybę. Vertinimo kokybei tirti taip pat pritaikytas ekspertinis kokybinis atvejų tyrimas daliai atrinktų ir jau įvertintų darbų. Buvo analizuojami pasitaikantys vertinimo netikslumai bei jų priežastys. Taip pat ištirta, kiek tolygiai pasiskirsto vertinimo metu gauti balai naudojant įprastą ir patobulintą pusiau automatinius programavimo užduočių vertinimo metodus.

Paskutinis disertacijos tyrimas buvo skirtas (6) žingsniui įgyvendinti – buvo tiriamos galimybės įdiegti metodo patobulinimus į specialiai tam nepritaikytą, EduJudge projekte kurtą, pusiau automatinio vertinimo įskiepį Moodle virtualiai mokymo aplinkai. Tyrimui taikyti sistemų analizės, sistemų projektavimo metodai.

1.6 Mokslinis darbo naujumas ir jo reikšmė

1. Pateiktas naujas programavimo užduočių pusiau automatinio vertinimo ir testavimo metodas. Jo interaktyvus ciklišKas vertinimo procesas leidžia tiksliai diagnozuoti užduotį sprendusiojo padarytas klaidas.

2. Pasiūlyti sprendimai, leidžiantys vertintojui gauti žymiai prasmingesnius grįžtamojo ryšio pranešimus iš automatinio testavimo sistemos.

3. Pateiktas programavimo užduočių pusiau automatinio vertinimo ir testavimo metodas yra efektyvesnis už kitus žinomus pusiau automatinio vertinimo ir testavimo metodus.

1.7 Ginamieji teiginiai

1. Naujas pusiau automatinio programų vertinimo ir testavimo metodas leidžia efektyviau (greičiau) įvertinti programavimo užduotis nei tai būtų galima atlikti rankinio vertinimo metodu.

2. Nauju pusiau automatinio programų testavimo ir vertinimo metodu paremtų sistemų formuojami įvertinimai objektyvumu nenusileidžia įprastiniam rankiniam vertinimui bei automatiniam testavimui.

1.8 Darbo rezultatų aprobavimas

1.8.1 Darbo rezultatų praktinė reikšmė

Sukurtas metodas pritaikytas Lietuvos valstybinio informacinių technologijų brandos egzamino praktinės dalies vertinimo sistemoje. Jos naudojimas padidina egzamino vertinimo efektyvumą keletą kartų.

Metodas taip pat bandytas su virtualios mokymosi sistemos „Moodle“ *Edujudge* įskiepiu, sukurtu pagal ES finansuotą projektą. Įskiepis tobulintas siekiant mokytojo komfortiškesnio vertinimo ir mokiniui aiškesnių vertinimo rezultatų pateikimo. Įsitikinta, kad metodas gali būti nesunkiai diegiamas ir kitose sistemose.

1.8.2 Pranešimai mokslinėse konferencijose

Pranešimai, skaityti Lietuvos ir tarptautinėse konferencijose:

- 12th International Conference on Computer Systems and Technologies, CompSysTech'2011, Vienna, Austrija, Vienna University of Technology.
- 2nd International Conference Social Technologies'2011: ICT for Social Transformations, Vilnius, Lietuva, Mykolo Romerio universitetas, Socialinės informatikos fakultetas.
- XV mokslinė kompiuterininkų konferencija, Kompiuterininkų dienos 2011, Klaipėda, Lietuva, Klaipėdos universitetas.
- 10th International Conference on Computing Education Research, "Koli Calling 2010", Koli National Park, Suomija.
- Lietuvos matematikų draugijos 51-oji konferencija, 2010, Šiauliai, Lietuva, Šiaulių universitetas.
- 14th Annual Conference on Innovation and Technology in Computer Science Education ITiCSE'2009, Paris, Prancūzija, Université Pierre et Marie Curie.

- 14-oji tarptautinė mokslinė kompiuterininkų konferencija, Kompiuterininkų dienos 2009, Kaunas, Lietuva, Kauno technologijos universitetas
- 8th International Conference on Computing Education Research, "Koli Calling 2008", Koli National Park, Suomija.

1.8.3 Publikacijos

Recenzuojamuose periodiniuose leidiniuose atspausdinta:

- **SKŪPAS Bronius** (2011). Pasikeitimų IT valstybiniame brandos egzamine analizė. *Informacijos mokslai*, ISSN 1392-0561, 57, 115–123.
- **SKŪPAS Bronius** (2010). Feedback improvement in automatic program evaluation systems. *Informatics In Education*, ISSN 1648-5831, Vol. 9, N. 2, 229–237.
- **SKŪPAS Bronius** (2010). Automatinio ir pusiau automatinio vertinimo ypatumai IT valstybiniame brandos egzamine. *Lietuvos matematikos rinkinys*, ISSN 0132-2818, 51, 154–159.
- POHL Wolfgang, BURTON Benjamin A., DAGIENE Valentina, **SKŪPAS Bronius**, FAKCHAROENPHOL Jittat, FORIŠEK Michal, HIRON Mathias, OPMANIS Martinš, van der VEGT Willem (2010). Get Involved! The IOI Workshop 2010, Its Goals and Results. *Olympiads In Informatics*, ISSN 1822-7732, Vol. 4, 158-169.
- **SKŪPAS Bronius**, DAGIENĖ Valentina, REVILLA Miguel (2009). Developing Classification Criteria for Programming Tasks. *ACM SIGCSE Bulletin*, ISSN 0097-8418, Vol. 41, N. 3, 373–373
- **SKŪPAS Bronius** (2009). Automatinio mokinių programų vertinimo sistemų lyginamoji analizė. *Informacijos mokslai*, ISSN 1392-0561, 50, 147–152.

Kitos publikacijos:

- DAGIENĖ Valentina, **SKŪPAS Bronius** (2011). Semi-Automatic Testing of Program Codes in the High School Student Maturity Exam. *Proceedings of the 12th International Conference on Computer Systems and Technologies*, (ACM International Conference Proceedings Series Vol. 578), ISBN 978-1-4503-0917-2, 564–569.
- VERDÚ Maria P., VERDÚ Elena, de CASTRO Juan P., PÉREZ María Á., REGUERAS Luisa M., DAGIENĖ Valentina, **SKŪPAS Bronius** (2010). Making a Contest with Edujudge and Questournament. *Edujudge System Handbook: How to Organize Programming Competitions in Moodle Courses*, ISBN 978-84-937580-4-2, 99–128.
- **SKŪPAS Bronius**, DAGIENĖ Valentina (2010). Observations from Semi-Automatic Testing of Program Codes in the High School Student Maturity Exam. *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, ISBN 978-1-4503-0520-4, 31–36.
- DAGIENĖ Valentina, DAGYS Viktoras, **SKŪPAS Bronius** (2009). ..EduJudge“ projektas programavimo kompetencijoms gerinti. *Kompiuterininkų dienos – 2009, Informacinės ir komunikacinės technologijos mokykloje*, 162–164.
- **SKŪPAS Bronius**, DAGIENĖ Valentina (2008). Is Automatic Evaluation Useful for the Maturity Programming Exam? *Proceedings of 8th International Conference on Computing Education Research*, ISBN 978-1-60558-385-3, 117–118.

1.9 Disertacijos struktūra

Darbą sudaro: terminų ir santrumpų žodynėlis, penkios pagrindinės dalys – skyriai, išvados ir rezultatai, naudotos literatūros sąrašas ir priedai.

Pirmajame skyriuje pateikiamas darbo įvadas. Pristatomas darbo aktualumas, tyrimų objektas, darbo tikslai ir uždaviniai, tyrimų metodai,

mokslinis naujumas, praktinė darbo reikšmė, ginami teiginiai ir darbo aprobavimas.

Antrajame skyriuje pateikiama informacija apie automatines ir pusiau automatines programavimo užduočių testavimo sistemas, pateikiama medžiaga apie atliktą šias sistemas aprašančių straipsnių analizę. Taip pat pateikiama informacija apie dabartinių sistemų klasifikavimą, jų veikimo principus, pasirinktų sistemų lyginamoji analizė, užduočių, tinkamų automatiniam vertinimui analizė.

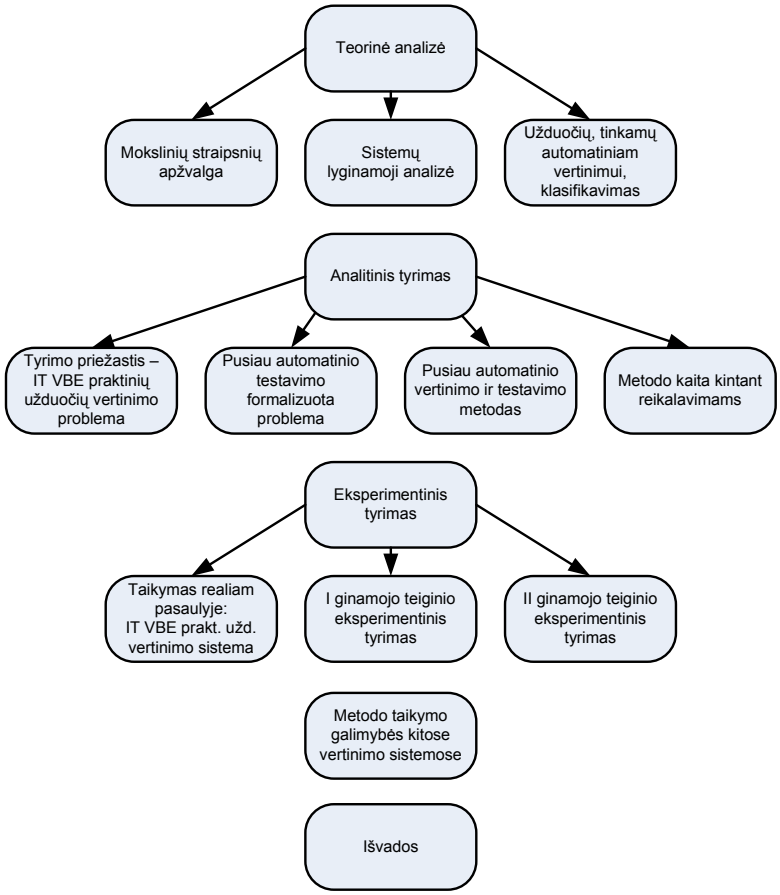
Trečiajame skyriuje detalizuojamas tyrimo konstravimu darbo planas, iškeliamos darbinės hipotezės, abstrahuojami reikalavimai kuriamai sistemai, kurioje bus išbandytas metodas, sistemos prototipo kūrimo eiliškumo tvarka pateikiami projektiniai metodo sprendimai. Taip pat pristatoma, kaip kintatys reikalavimai egzamino vertinimui ir vykdymui įtakoja sistemos modifikavimą ir jų įtaką vertinimo metodui.

Ketvirtajame skyriuje patekiama informacija apie atliktus eksperimentinius sukurto metodo efektyvumo ir jo taikymo teikiamos mokinių programų vertinimo vertinimo kokybės tyrimus.

Penktajame skyriuje aptariamos galimybės metodą taikyti kitose pusiau automatinėse vertinimo sistemose.

Darbo pabaigoje pateikiamas rezultatų apibendrinimas ir išvados.

Prieduose pateikiama: egzamino sąlygų ir vertinimo schemų pavyzdžiai, klausimynas vertintojams.



1.1 pav. **Disertacijos struktūra**

2 Automatinių programavimo užduočių vertinimo sistemų apžvalga

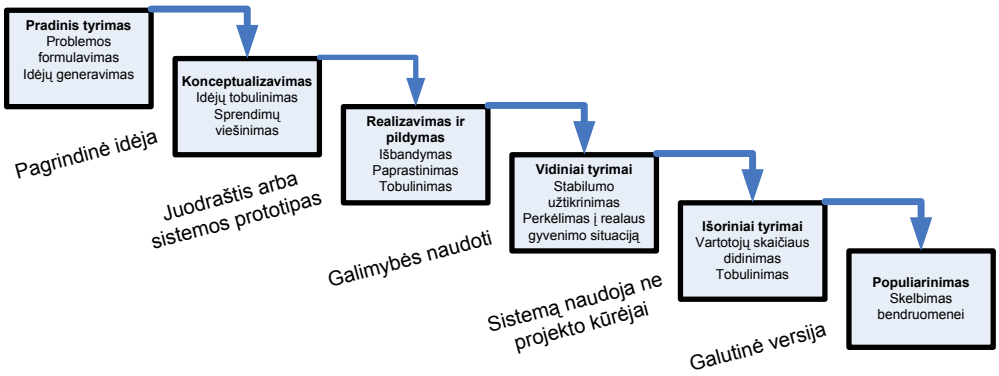
Mokymo srityje jau seniai tiriamos vertinimo pasitelkus kompiuterį (CAA, computer aided assessment) galimybės [12]. Tačiau dažniau analizuojamos galimybės panaudoti kompiuterį teorinėms apklausoms bei testams, nei galimybės jį naudoti praktinių darbų vertinimui. Kompiuteris šiuose darbuose daugiau duomenų kaupimo ir apdorojimo įrankis ir komunikacijos priemonė nei priimanti sprendimus sistema [5].

Atvirųjų klausimų, algoritmų, programų vertinimas dažnai reikalauja žymiai daugiau laiko nei testo su uždaraisiais klausimais vertinimas. Todėl turėtų būti skiriama daugiau dėmesio praktinių užduočių bei atvirųjų klausimų automatinio vertinimo problemų sprendimui. Tačiau šioje srityje tyrimų atliekama mažiau. Viena šios tendencijos priežasčių – bendroji vertinimo pasitelkus kompiuterį teorija yra svarbi ne vien tik informatikos mokymui.

2.1 Automatinio programavimo užduočių vertinimo raida

Redwine ir Riddle [67] suformulavo programinės įrangos idėjų brandos modelį, kuriame tvirtino, kad žymiai daliai programinės įrangos novatoriškų idėjų reikia maždaug 50 metų iki visuotinės sklaidos. Šio modelio schema (2.1 pav.) parodo žingsnius nuo pirminės idėjos iki ją realizuojančios programinės įrangos paplitimo. Įdomu tai, kad automatinio programavimo užduočių vertinimo sistemų vystimosi istorija patvirtina šias prielaidas.

Automatinio programavimo užduočių vertinimo sistemų raidos analizei buvo labai naudingos išsamios Ala-Mutka [5], Colton ir kt. [14],



2.1 pav. **Redwine ir Riddle [67] programinės įrangos idėjų brandos modelis**

Douce ir kt. [20], Ihantola ir kt. [37], Liang ir kt. [51], Queirós ir Leal [64], Romli ir kt. [71] automatinio programavimo užduočių vertinimo sistemų apžvalgos.

Ankstyviausias automatinio programavimo užduočių vertinimo sistemos pavyzdys aprašytas Hollingsworth [35]. Sistema tikrino perforuotose kortose pateiktas asemblerio programas. Buvo užsiminta apie galimybę patobulinus sistemą tikrinti FORTRAN programavimo kalba parašytas programas.

Sistemos evoliucionavo gana greitai – apie 1961 metus buvo sukurta sistema, skirta pradedančiųjų Algolo programuotojų mokymui Stanfordo universitete, Jungtinėse Amerikos Valstijose. Užduotys reikalavo parašyti paprogrames, kurios buvo tikrinamos su atsitiktiniu būdu sukurtais testais. Šioje sistemoje jau buvo tikrinamas atsakymo teisingumas [24, 57].

Hext ir Winings [30] aprašyta sistema jau galėjo savarankiškai kompiliuoti ir įvykdyti pateiktas programas su dviem testais be žmogaus įsikišimo. Be to pirmą kartą paminėtas agregavimo funkcijų taikymas įvairiems programos vykdymo vertinimo matams (sėkmingas kompiliavimas, vykdymo laikas, ...) apjungti. Šios kartos sistemų naudojimas reikalavo daug techninių žinių.

Antra automatinio programavimo užduočių vertinimo sistemų karta

rėmėsi esamų OS ir naujai sukurtų įrankių naudojimu [20]. Sistemos dažniausiai buvo paleidžiamos specialia komanda komandinės eilutės interpretatoriuje ir buvo vertinama konkreti nurodyta programa ar programų rinkinys. Tokios sistemos pavyzdį aprašo Isaacson ir Scott [38]. Įdomu tai, kad šiame straipsnyje jau užsimenama apie pusiau automatinio vertinimo poreikį – tvirtinama, kad patikrinus teisingumą reikia peržvelgti programavimo stilių.

Paminėtai antrajai sistemų kartai taip pat priskirtina ir Reek [68] aprašyta TRY sistema, kurioje numatytas daugiavartotojiškumas, galimybė pateikti sprendimą į mokytojo paskyrą UNIX OS, galimybė išsaugoti visų testų rezultatus, galimybė kartoti sprendimo pateikimą keletą kartų. Pirmą kartą paminėta saugumo problema – užsiminta, kad studento programa gali būti kenkėjiška ir ją vykdyti be jokių papildomų apsaugos priemonių gali būti pavojinga.

Taip pat svarbus buvo sprendimas sukurti prieigą prie vertinimo sistemos per kompiuterių tinklą [7, 85]. Ne mažiau svarbūs atsiradę vertinimo ataskaitos, skirtingi testų svoriai.

Jackson ir Usher [40] pristatyta ASSYST sistema viena artimiausių modernioms pusiau automatinėms programavimo užduočių vertinimo sistemoms. Joje numatytas teisingumo, efektyvumo, programavimo stiliaus, studento kurtų testų kokybiškumo vertinimas. Programavimo stilius vertinamas automatiškai, tačiau vertintojui leidžiama šį vertinimą pakeisti. Taip pat svarbus ir sistemos teikiama taškų apibendrinimo į ataskaitas idėja.

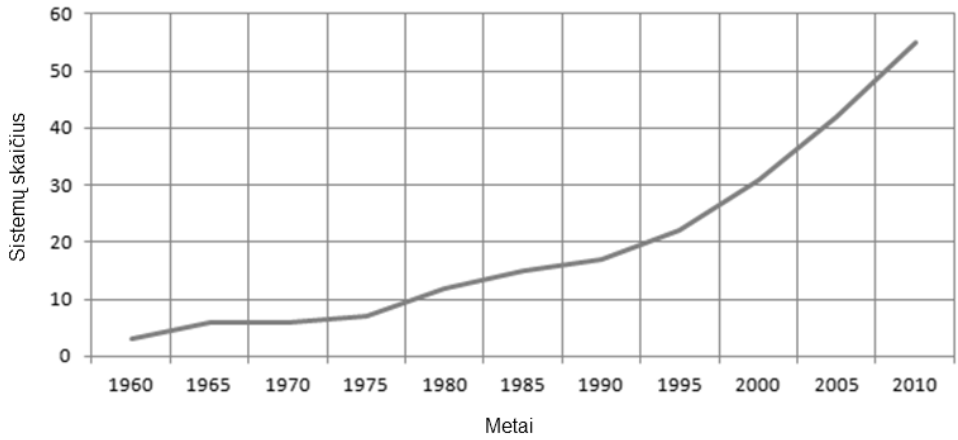
Varviko universitete (Jungtinė Karalystė) beveik tuo pačiu metu sukurta panaši į ASSYST sistema BOSS [41, 42, 43]. Ji tebetobulinama iki šiol ir tyrime buvo analizuojama giliau.

Trečioji automatinio programavimo užduočių vertinimo sistemų karta sietina su saityno atsiradimu. Šios kartos sistemų sąsaja dažniausiai realizuojama panaudojant saityno technologiją. Atsirado eksperimentinių

sudėtingų vertinimo būdų, tokių kaip vertinimas panaudojant diagramas *CourseMarker* sistemoje [26, 31, 32, 33, 50]. Sistemose įdiegiama galimybė sprendimui naudoti įvairias programavimo kalbas, automatinis programos struktūros vertinimas, platesni pranešimai užduotį sprendžiančiajam, plagiato prevencija ir pan. [20].

Automatinio programavimo užduočių vertinimo sistemų, taikomų varžybose, vystymasis gana panašus. Jį apžvelgia disertacijoje Skūpienė [75]. Skūpienė [73] pažymi, kad Lietuvos mokinių informatikos olimpiadose įvyko perėjimas nuo sistemų, besiremiančių įrankiais į saityno tipo sistemas. Taip įvyko dėl poreikio lengvai surinkti mokinių darbus, pateikti greitą atsakymą apie programos funkcionalumą su sąlygos testais.

Kasmet išleidžiama vis naujų automatinio ir pusiau automatinio programavimo užduočių vertinimo sistemų. Jų augimo tempus iliustruoja 2.2 paveikslas.



2.2 pav. **Automatinio ir pusiau automatinio programavimo užduočių vertinimo sistemų skaičiaus augimas pagal metus [64]**

2.2 Pagrindinės tyrimų kryptys mokslinėje literatūroje

Paskutiniuoju metu skiriamas didesnis dėmesys automatinių sistemų taikymui. To pavyzdys – 2005 m. išleistas žurnalo „ACM Journal of Edu-

cational Resources in Computing“ specialus numeris, skirtas automatinio vertinimo problemų analizei [1].

Tačiau žymi dalis dėmesio yra nukreipta į aukštųjų mokyklų kursų dėstymą [20, 29, 65, 87]. Nemažai straipsnių gilinasi ir į automatinius įrankius [4, 26, 28, 78, 85] Straipsnių apie testavimo sistemų taikymą konkursuose [6, 46, 86] Nemažai šia tema straipsnių spausdina Lietuvoje leidžiami žurnalai „Olympiads in Informatics“ [6, 10, 19, 45, 46, 55, 61, 62, 63, 70, 80, 81, 84, 86] bei „Informatic in Education“ [15, 16, 22, 23, 36, 58, 60, 83]

Tyrimų kryptys kito kartu su automatinių vertinimo sistemų kartomis. Naujesnių kartų sistemų tyrimai dažnai tęsė jau anksčiau pradėtus tyrimus, tačiau su kiekviena karta atsirasdavo ir naujų krypčių.

Pirmos kartos sistemoms buvo būdingi tyrimai:

- koncepcijos pristatymas [35];
- aukšto lygio programavimo kalbų palaikymas [24, 57];
- sistemų plėtimo galimybės [30].

Su antros kartos sistemomis išsivystė tyrimai šiomis kryptimis:

- kelių kalbų palaikymas automatiniame testavime [85];
- pusiau automatinio testavimo idėjų pristatymas [39, 40];
- užduotį sprendusiojo gaunamų testavimo rezultatų tobulinimas [68];
- turinio (užduočių, testų, studentų) valdymas [41, 42, 43, 52];
- statinė programavimo stiliaus analizė [25, 40];
- programavimo varžybų sistemos [17, 74];
- studentų savarankiškas naudojimasis sistema[40].

Atsiradus trečios kartos sistemoms atsirado naujos tyrimų kryptys:

- saityno sąsajos taikymas sistemose [26, 31, 59];
- debesų kompiuterijos bei nutolusių tarnybinių stočių naudojimas testavimui [82];

- galimybė naudoti kelias programavimo kalbas vienai užduočiai spręsti [49];
- užduotį sprendusiojo gaunamos informacijos įvairovės didinimas [34, 53, 54, 79];
- plagiato identifikavimas [9, 13, 21];
- įvairios vertinimo tobulinimo idėjos [11];
- dėstomų kursų valdymas bei integracija į mokymo aplinkas [48];
- automatinis testų generavimas [71];
- užduočių perkėlimo tarp sistemų tyrimai bei problemos [64].

Dauguma šių tyrimų krypčių tebėra aktyviai aptariamos ir šiuolaikiniuose straipsniuose.

Paskutiniais metais daugiausia buvo gilinamasi į sistemų integraciją į virtualias mokymo aplinkas, automatinį testų generavimą ir užduočių perkėlimą tarp sistemų.

Automatinis testų generavimas labai aktualus viešoms universitetų uždavinių bazėms, kurių užduočių testų rinkinius išviešinę vartotojai trukdo patikrinti kitų vartotojų pateiktus sprendimus. Nelabai sąžiningi vartotojai pritaiko sprendimus konkretiems testams.

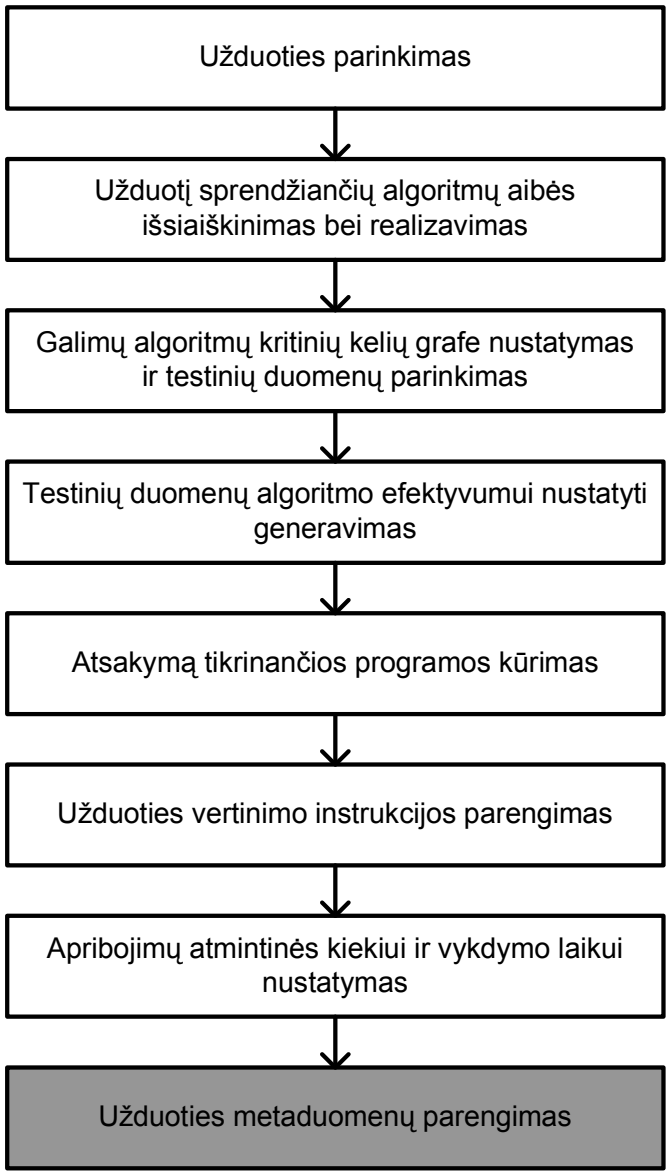
Užduočių perkėlimo tarp sistemų problema taip pat yra labai aktuali ir dabar, nes užduočių parengimo automatinei vertinimo sistemai uždavinys yra reikalaujantis daug pastangų, nors tik nežymi dalis jų yra susieta su automatinių testavimo sistemų architektūra (2.3 pav.).

2.2.1 Vertinimo sistemų veikimo būdai

Ala-Mutka [5] pateikia automatinių programų vertinimo sistemų klasifikacijos modelį ir bando palyginti dalį sistemų. Atskiriami du programų analizės metodai – statinis ir dinaminis.

Dinaminė analizė sudaro prielaidas vykdymo sekai stebėti, atrasti pavojingą veikseną. Vertinimui naudingiausi šie analizės rezultatai:

- funkcionalumo įvertis;



2.3 pav. **Užduoties parengimas automatinei vertinimo sistemai. Paryškinta dalis esmingai skiriasi skirtingoms automatinio vertinimo sistemoms**

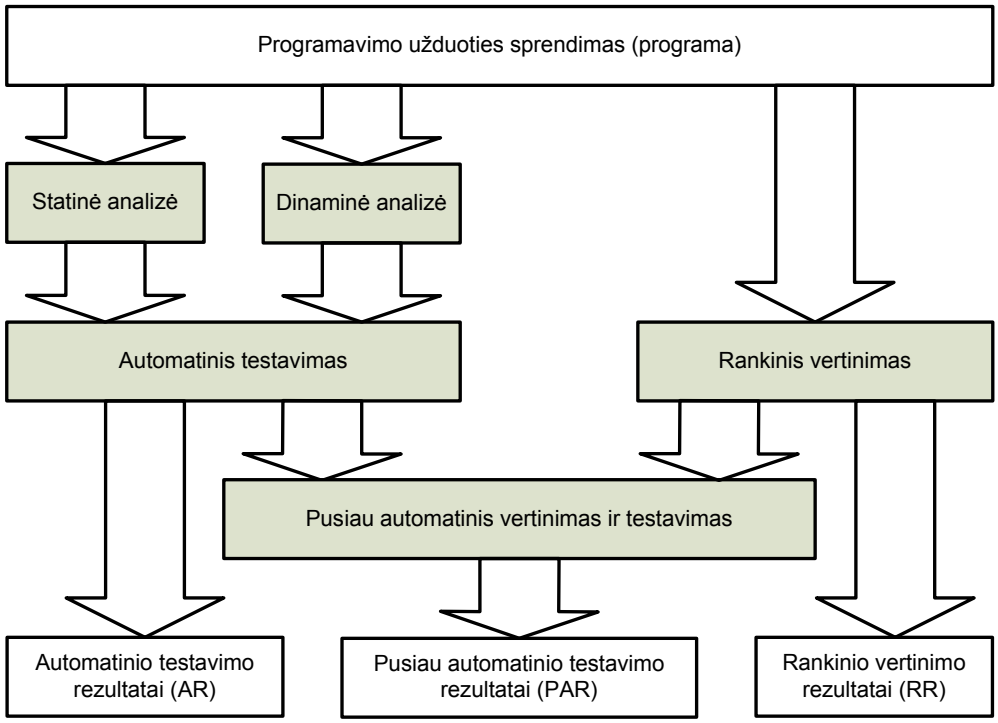
- efektyvumo įvertis;
- studento testavimo gebėjimų įvertis;
- kitos savybės (programavimo kalbos konstrukcijų naudojimas, duomenų struktūrų stebėjimas, ...).

Panaudojant statinę analizę dažnai realizuojamos šios funkcijos:

- programavimo stiliaus vertinimas;
- programų sintaksinių ir semantinių klaidų paieška;
- programų matų skaičiavimas (vykdomų eilučių skaičiuss ir pan.);
- programos dizaino ir sąsajos vertinimas;
- programų struktūros ir veiksmų palyginimas;
- programų raktažodžių paieška;
- programų struktūrogramų analizė;
- programų plagiato atpažinimas.

Sistemos skirstomos į visiškai automatines ir pusiau automatines. Pusiau automatinių sistemų (2.4 pav.) esminis skirtumas nuo automatinių yra vertinimo algoritme – pusiau automatinėse sistemose panaudojamas automatinio testavimo modulis, jo darbo rezultatai rodomi vertintojui ir vertintojas priima galutinius sprendimus dėl atskirų vertinimo komponentų ir galutinio įvertinimo. Dažnai vertintojas be parašomo įvertinimo parašo mokiniams vieną kitą sakinį, kurie nurodo esmines klaidas, pastabas apie darbo kokybę ar programos trūkumus. Yra pusiau automatinių vertinimo sistemų, kurios šiems tekstiniams pranešimams kurti siūlo fiksuotus arba išplečiamus frazynus.

Taip pat yra sprendimų, kur programų vertinimui pritraukiami kiti studentai. Tokiu būdu pritrauktas didesnis vertintojų skaičius greičiau susidoroja su dideliais kiekiais studentų darbų universitetuose bei studentai tuo pačiu pagilina savo žinias.



2.4 pav. **Vertinimo sistemų sąveikos būdai**

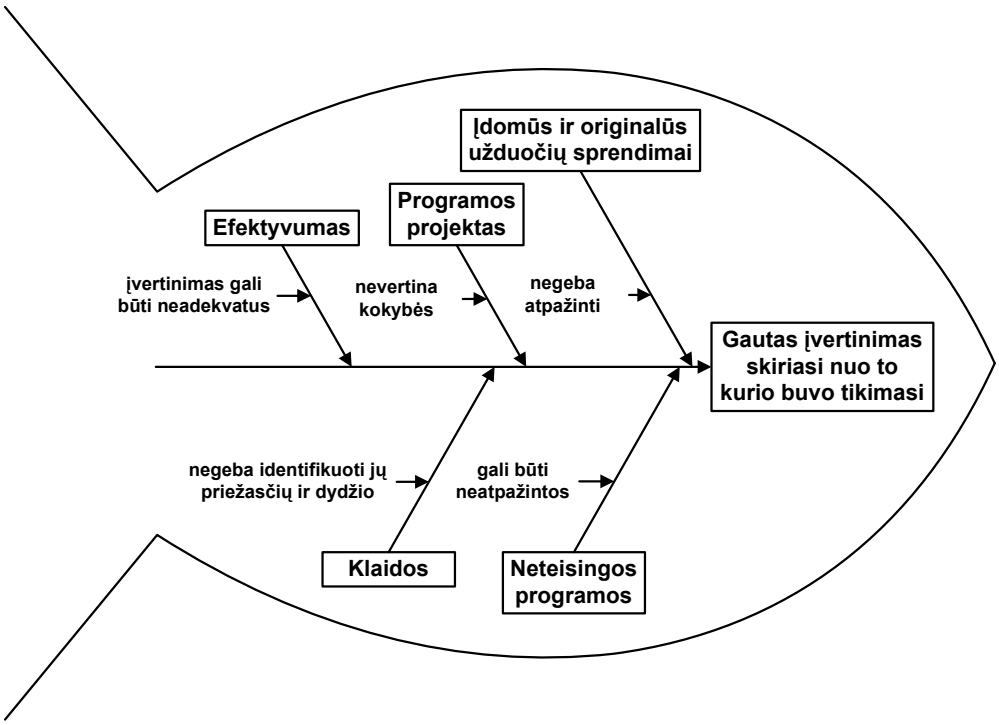
2.2.2 Juodosios dėžės principu veikiantis programų testavimo metodas

Juodosios dėžės principas gana dažnai naudojamas įvairiose inžinerijos kryptyse. Šia sąvoka norima pabrėžti, kad vartotojas negali įsiterpti į įrenginio ar programos veikią ir gali ja naudotis tik pagal dokumentaciją. Visa informacija gaunama tik analizuojant veikimo rezultatus.

Programos testavimas juodosios dėžės metodu vyksta su iš anksto parinktais duomenimis, tikintis gauti rezultatus, kuriuos galima patikrinti. Išvados apie programos kokybę daromos tik iš išoriškai pamatuojamų parametrų: vykdymo laiko, panaudotos OS atmintinės bei rezultatų teisingumo.

Tačiau, nors šis metodas yra vienas populiariausių, dar E. Dijkstra [18] jį kritikavo tvirtindamas, kad „testavimas sėkmingai gali parodyti esančias klaidas, tačiau nėra pajėgus įrodyti, kad jų nėra“.

Pagrindinė kritika gali būti vizualiai pavaizduota *Ishikawa* tipo problemų ir jų priežasčių diagrama, esančia 2.5 paveiksle.



2.5 pav. ***Ishikawa* tipo problemų ir jų priežasčių diagrama, vaizduojanti juodosios dėžės testavimo problemas**

2.2.3 Balų agregavimo funkcijos

Automatinėse ir pusiau automatinėse sistemose gali būti skirtingi vertinimo formavimo mechanizmai – sistema gali sumuoti balus už skirtingas savybes, kitu atveju gali reikalauti tol tobulinti, kol bus gautas visus reikalavimus patenkinantis rezultatas. Galimi ir kiti variantai. Matematiškai apibrėžtas balų apjungimo būdas vadinamas *balų agregavimo funkcija.*

Šis balų agregavimo funkcijos yra populiariausios automatinėse testavimo ir vertinimo sistemose:

- Dalinio agregavimo funkcija – testavimo rezultatų agregavimo būdas, kai kiekvienam iš g testų priskirtas atskiras įvertis w_i

($i = 1 \cdots g$). Įverčių suma sutampa su visa užduoties verte P:

$$P = \sum_{i=1}^{g} w_i \tag{2.1}$$

Užduoties įvertinimas S gaunamas taip:

$$S = \sum_{i=1}^{g} w_i t_i, \tag{2.2}$$

kur $t_i \in \{0, 1\}$ – testo i įveikimo indikatorius. Tai labai populiarus vertininimo būdas. Paplitęs ir Tarptautinėse informatikos olimpiadose (IOI), Lietuvos mokinių informatikos olimpiadose ir įvairiuose kituose programavimo konkursuose. Ši funkcija kritikuojama dėl situacijų, kai mokinys atspėjęs dažniau pasikartojantį atsakymą ir tiesiog jį spausdindamas gali gauti nemažai neužtarnautų balų.

- „Viskas arba nieko“ agregavimo funkcija – balai rašomi tik tuo atveju, jei programa įveikia visus testus. Gana populiari agregavimo funkcija programavimo mokymo sistemose. Užduoties balai P arba suteikiami visi, arba – nė vieno:

$$S = P \prod_{i=1}^{g} t_i \tag{2.3}$$

Iš dalies naudojama ir informacinių technologijų valstybinio brandos egzamino praktinių užduočių vertinimo sistemoje. Dažnai mėgstama šį metodą rinktis pabrėžiant, kad jei bent vienas testas neveikia – tai reiškia, kad programa ne iki galo gerai veikia. Su tuo galima sutikti, jei užduotis nėra labai sudėtinga, neturi sunkiai nuspėjamų sudėtingų atvejų.

- „Viskas arba nieko grupėms“ agregavimo funkcija. Testavimo rezultatų agregavimo būdas, kai tik įveikus visus testus iš vienos grupės

gaunamas grupei priskirtas ne nulinis įvertis. Grupių įverčiai juos agreguojant yra sumuojami.

$$S = \sum_{i=1}^{g_c} w_i \prod_{j=1}^{g_{c\,i}} t_j \tag{2.4}$$

kur g_c testų skaičius, w_i $(i = 1 \cdots g_c)$ testų grupių vertės ir g_{c_i} testų skaičius grupėje c_i $(i = 1, \cdots g_c)$. Svoriai tenkina lygybę $\sum_{i=1}^{g_c} w_i = P$, kur P yra užduoties vertė. Ši agregavimo funkcija populiarėja tarptautinėje informatikos olimpiadoje. Savo esme ji yra tarpinis variantas tarp aukščiau minėtų agregavimo funkcijų. Jos taikymas leidžia suteikti taškų ir už ne iki galo baigtą sprendimą, jei sėkmingai skaičiuojami rezultatai keletoje kitų situacijų.

2.3 Sistemų tyrimo metodai

Darbui taikytas teorinis tyrimo metodas, vykdyta lyginamoji analizė.

Pagrindiniu sistemų parinkimo tyrimui argumentu buvo jų priklausomybė tam tikrai taikymo sričiai, informacijos apie jas gausa bei šių sistemų minėjimas kituose šaltiniuose. Taip pat argumentu buvo sistemų populiarumas tarp mokinių bei studentų.

Trumpai pristatykime tirtas sistemas:

- *Valadolido universiteto programavimo mokymo sistema (UVa Online Judge).* Vienas didžiausių pasaulyje programavimo užduočių rinkinių. Laisva registracija. Sistema galima naudotis laisvai, bet veikimo algoritmai nėra viešinami. Informacija gauta iš autorių straipsnių bei asmeninio bendravimo. Pateikti uždaviniai neturi aiškaus klasifikavimo, todėl uždavinių pasirinkimui verta naudoti kitose svetainėse pateiktus sąrašus.
- *EduJudge sistema. Iš UVa Online Judge išsivysčiusi daugiakomponentinė sistema,* kurta stengiantis užduočių banką, programavimo užduočių testavimą paversti paslauga, teikiama įvairiems mokymo

aplinkos Moodle naudotojams. Išsiskiria numatytų funkcijų gausa, daugiakomponentiškumu, galimybe išskirstyti į daug nutolusių darbo stočių. Tebėra iki galo neužbaigta, tačiau vystymas sustojęs.

- *JAV informatikos olimpiados treniruočių portalas (USACO training program).* Viena populiariausių algoritmavimo mokymo sistemų-mokyklų. Laisva registracija, aiški užduočių seka ir patarimai, spartus automatinis testavimas. Užduočių rinkinys nėra labai gausus, nėra pildomas. Veikimo principus autorius ne kartą pristatė straipsniuose.
- *Varviko universiteto studentų programų vertinimo sistema (BOSS Online Submission System).* Viena pirmųjų sistemų, kuri buvo plačiai prieinama universitetams. Tyrimo metu buvo uždaro kodo, tačiau šiuo metu jau yra atviro kodo supaprastintas analogas BOSS2. Kaip ir daugumoje universitetinių sistemų skiriama daug dėmesio plagiavimo prevencijai. Automatinis programavimo užduočių testavimas tėra tik viena šios sistemos naudojimo kryptis.
- *Notingemo universiteto studentų programų vertinimo sistemos (Ceilidh, CourseMarker).* Daugiausiai straipsniuose minimos automatinio programavimo užduočių testavimo sistemos universitetuose. Sistemos specializuotos automatinio programavimo užduočių testavimui. CourseMarker kurta tobulinant Ceilidh sistemą.
- *2002 m. Tarptautinės informatikos olimpiados programavimo varžybų aptarnavimo sistema (IOI 2002 Contest and Grading System, IOI'2002 PVAS).* Pirma tarptautinėje olimpiadoje naudota sistema, kurios programos tekstas yra viešai pasiekiamas.
- *Lietuvos informatikos olimpiados programavimo varžybų aptarnavimo sistema (Lietuvos informatikos olimpiados PVAS).* Tyrimo metu olimpiadoje naudota iš IOI 2002 Contest and Grading System kilusi sistema. Neseniai sitemos branduolys buvo pakeistas į Vokietijos

olimpiadų organizatorių kurtą sistemą, o Lietuvoje kurti moduliai adaptuoti naujai sistemai.

- *Informacinių technologijų valstybinio brandos egzamino praktinių užduočių vertinimo sistema (IT VBE PUVS).* Tyrimo metu buvo analizuojamas sistemos prototipas. Šiuo metu egzamine naudojama sistema yra žymiai patobulinta.

Apie sistemas informacija rinkta literatūroje, sistemų svetainėse viešai publikuojamoje dokumentacijoje (pvz., CourseMaker). Dalį sistemų pavyko išbandyti (UVa Online Judge. USACO, IOI‘2002 PVAS, LIO PVAS, IT VBE PUVS). Kitos buvo sunkiau prieinamos, pasiekiamos tik komerciniams vartotojams, laisvai neplatinamos (BOSS, CourseMarker, Ceilidh).

Sistemas analizuojant joms buvo suteikiami raktiniai žodžiai, kurie parodė jų vietą dabartinėje sistemų klasifikacijoje, o taip pat išryškino jas skiriančius bruožus.

2.4 Sąsajos tarp tiriamų sistemų

Šiuo metu Lietuvos mokyklose labiausiai žinoma Lietuvos olimpiadų PVAS. Ji yra viena paprasčiausių. Sistemai būdingas automatizuotas juodosios dėžės testavimas – taigi dinaminė analizė. Esminiai sistemos patobulinimai IOI‘2002 PVAS atžvilgiu – dviejų mokinių amžiaus grupių atskyrimas, automatinė registracija varžyboms, bei pusiau automatinio vertinimo modulis skirtas sprendimų stiliui bei idėjai vertinti. Sistemos vienas svarbiausių privalumų, perimtų iš IOI‘2002, – sistema dirba tinkle, turi kelių testavimo mašinų valdymo mechanizmą. 2010 metais buvo pereita prie kitos sistemos, kuri paremta Vokietijoje naudojama olimpiadų sistema. Esminis bruožas – lengvesnis administravimas, mokiniams pasiekiami rezultatai ir sistemos pranešimai pasibaigus varžyboms, galimybė eksperimentuoti su programomis varžyboms pasibaigus, visi duomenys saugomi naudojant duomenų bazių valdymo sistemą MySQL. Tačiau Lietuvoje taikomo pusiau automatinio vertinimo ji realizuoto neturi, todėl

Sistemos		Skirtos mokymui					Trumpo naudojimo laikotarpio		
		Pasirengimo varžyboms			Universitetinės		Varžybų		Egzaminų
		UVA Online Judge	EduJudge	USACO training program	BOSS Online Submission System	Ceilidh ir CourseMarker	IOI 2002 Contest and Grading System	Lietuvos informatikos olimpiados PVAS	IT VBE prakt. užd. vertinimo sistema
Dinaminė analizė	Funkcionalumo ir efektyvumo įverčiai	+	+	+	+	+	+	+	+
	Studento testavimo gebėjimų įvertis (galima testuotis su savo testais)	-	-	-	-	+	+/-	+/-	-
	Kitos savybės (pr. kalbos konstrukcijų naudojimas, duomenų struktūrų stebėjimas, ...)	-	+[2]	-	-	+[2]	-	-	+[2]
Statinė analizė	Programavimo stiliaus vertinimas	-	+[1]	-	+[2]	+	-	+[3]	+
	Sintaksinių ir semantinių klaidų paieška[4]	-	-	-	-	+[2]	-	-	+[2]
	Programų matų skaičiavimas (vykdomų eilučių skaičius, ...)	-	+[1]	-	+[2]	+	-	-	+
	Programos dizaino ir sąsajos vertinimas	-	+[3]	-	+[2]	+[3]	-	-	+[3]
	Programų struktūros ir veiksmų palyginimas	-	-	-	+[2]	+	-	-	+
	Programų raktažodžių paieška	-	+[1]	-	+[2]	+	-	-	+
	Programų struktūrogramų analizė	-	+[1]	-	+[2]	+[2]	-	-	-
	Programų plagiato atpažinimas	-	+[1]	-	+	+	-	-	-
Vertin. automat.	Vertinimas visiškai automatizuotas	+	+	+	+	+	+	+	+
	Kai kurie komponentai gali būti vertinami rankiniu būdu	-	+	-	+	+	-	+	+
Vertinimo formavimo mechanizmas	Reikalauja veikiančių visų teisingų testų	+	+	+	+	+	-	-	-
	Įvertinimas formuojamas pagal skirtingą teisingų testų vertę	-	+	-	-	+	+	+	+
	Vertinimas grupuojant testus	-	+	-	-	-	-	-	-
	Įvertinimas gali priklausyti nuo laiko momento ir konkurentų pasiekimų	-	+	-	-	-	-	-	-
	Vertinama tik praėjus varžybų momentui	+	+	-	+	+	+	+	+
Užd. bankas	Yra parengtas užduočių bankas	+	+	+	-	-	-	+/-	+/-
	Gali pildyti kiti vartotojai (pvz., mokytojai)	-	+	-	+	+	-	-	-
Testo klaus.	Atviri	-	+	-	-	+	-	-	-
	Uždari	-	+	-	-	+	-	-	-
Verti mas	Yra galimybė išversti	-	+	-	-	+	+	+	-
	Išversta	-	+/-	-	-	-	-	+	+
Kaina	Naudotis galima nemokamai	+	+	+	-	-	+	+	-
	Atvirasis kodas	-	+	-	-	-	+/-	+/-	+/-

[1] Kreipimosi į kitas sistemas būdu.
[2] Galima naudojant papildomus įrankius.
[3] Atliekant pusiau automatinį vertinimą.
[4] Sintaksinis tikrinimas kompiliavimo metu realizuojamas visose sistemose. Čia akcentuojama papildomos analizės galimybė.

2.1 lentelė. **Automatizuoto vertinimo sistemų palyginimas**

teko adaptuoti modulį taikytą senesniajai sistemai. Vienintelis šio modulio trūkumas, kad jis nėra pilnai integruotas į sistemą ir mokiniai galutinius balus gauna atspausdintus varžyboms pasibaigus.

IT valstybinio brandos egzamino PUVS skiriasi platforma – ji veikia Windows operacinėje sistemoje, paprasta struktūrine schema su viena testavimo mašina. Esminis privalumas – numatyta ne tik juodosios dėžės principo dinaminė analizė, bet ir statinė analizė. Ji atlieka programų matų skaičiavimus, vertina stilių, atlieka raktažodžių paiešką. Taip pat joje tobulinamas pusiau automatinis interaktyvus vertinimas, leidžiantis eksperimentuoti su sprendimais.

Aukščiau paminėtos sistemos neturi mokymo funkcijos – jos numatytos darbui apribotu laiku. Pateikiamos programos sprendžiamos varžybų ar egzamino metu, užduočių dalyviai rinktis negali, sistemos grąžinama informacija užduotį sprendžiančiajam ribota, vertinimo rezultatai skelbiami po tam tikro laiko. Testiniai duomenų rinkiniai slepiami iki rezultatų skelbimo.

Toliau minimos sistemos skirtos daugiau mokymui nei varžyboms, nors tai netrukdo jas panaudoti ir varžyboms.

Sistemos BOSS, Ceilidh, CourseMarker yra skirtos universitetų bendruomenėms, naudojamos programavimo mokymo kursuose. BOSS sistema daug dėmesio skiria atvirų klausimų testavimui. Automatinis programų vertinimas – tai tik vienas iš kelių jos modulių (Heng, 2005). Joje realizuota tik juodos dėžės dinaminės analizės vertinimo metodika. Pagal galimybes naudoti įvairius vertinimo modulius, dinaminę ir statinę analizę toliausiai pažengusi sistema yra CourseMarker. Ši sistema išsivystė iš prieš tai naudotos sistemos Ceilidh, kuri buvo viena labiausiai paplitusių (ją naudojo apie 200 įstaigų). Kuriant CourseMarker sistema buvo perrašyta iš naujo stengiantis išlaikyti didžiąją dalį Ceilidh galimybių. Šiai sistemų grupei būdinga naudoti gausybę įvairių įrankių. Tai suteikia sistemai universalumo, tačiau nukenčia sistemos mobilumas. Ji tampa

sunkiai perkeliama į kitus kompiuterius, sudėtingai integruojama. Šios sistemų grupės atstovams būdingos pastangos tikrinti darbus plagiato indikatoriais.

UVA Online Judge [66, 69] ir USACO training program [46] (toliau USACO) yra prieinamos nemokamai. UVA Online Judge ir USACO buvo kuriamos norint paruošti studentus ir mokinius programavimo varžyboms (ACM ICPC ir IOI atitinkamai). Todėl jų mokomoji funkcija truputį skiriasi nuo kitų universitetinių sistemų. Šiose sistemose mažiau kreipiamas dėmesys į programavimo stilių, sprendimo elegantiškumą. Daugiau dėmesio skiriama algoritmų funkcionalumui, efektyvumui, jos pilnai automatinės, pritaikytos priimti didelį lankytojų kiekį. Šiuo metu USACO užduočių bankas yra nedidelis (97 užduotys), geriau struktūrizuotas, užduotys mokiniams pateikiamos po kelias, jos suskirstytos į temas. USACO orientuojasi į savarankišką mokymąsi, todėl naudoti pamokose nėra patogu. UVA Online Judge suteikia žymiai daugiau laisvės renkantis užduotį (užduočių bankas yra didesnis – apie 2000 užduočių), pateikiama šios užduoties sprendimo statistika.

Internete yra nemažai sistemos naudotojų svetainių, analizuojančių, klasifikuojančių užduotis – taip stengiamasi sumažinti sistemos mokomojo vaidmens trūkumą. Vystytas EduJudge projektas turėjo praplėsti UVA Online Judge galimybes, išspręsti lokalizavimo, sistemos moduliškumo, integravimo į mokymo sistemas problemas. Tai turėjo būti sprendžiama integruojant automatizuoto programų vertinimo galimybes į virtualiąją mokymosi aplinką Moodle. Užduočių bankas buvo perimtas iš UVA Online Judge. Jis bus struktūrizuotas, patogesnis mokymui. Vienintelė bėda, kad ES projektui pasibaigus EduJudge sistema autorių noru nebuvo pilnai išviešinta, kai kurie jos komponentai tebeturi tik eksperimentams tinkamą kokybę.

2.5 Analizės rezultatai

Sistemų palyginimas parodė, kad daugumai sistemų būdingas specifinis užduočių rengimo formatas. Tai sukelia papildomas problemas norint pakeisti dabar naudojamą sistemą kita ar papildyti dabar naudojamos sistemos užduočių sąrašą. Nors IMS pasaulinis mokymosi konsorciumas (Global Learning Consortium) yra pasiūlęs klausimų ir testų saugojimui XML schemą IMS QTI (IMS QTI, 2005), tačiau ji be papildymų programavimo užduotims nėra tinkama. Mokymo objekto modelis IEEE LOM (IEEE LOM, 2002) kaip pagrindas programavimo užduotims turėtų tikti, tačiau dėl programavimo užduočių specifikos (testų duomenys, rezultatai, tikrinimo programa, laiko ribojimai ir t.t.) jį reikia plėsti. EduJudge projekto rėmuose kuriama programavimo užduočių specifikacijos schema (LEAL, QUEIRÓS, 2008), kuri išplečia IEEE LOM. Tuo būdu artimiausiu metu turėtų nusistovėti galimai standartinė užduočių mokomojo objekto metaduomenų schema. Turėtų pagerėti galimybės užduotis kilnoti tarp įvairių programų automatinių vertinimo sistemų. Kuriant kitas sistemas reiktų atsižvelgti į šią iniciatyvą ir stengtis naudoti vienodus užduočių apibrėžimo standartus.

Kita bendra sistemų savybė – didžioji dalis sistemų naudoja juodosios dėžės dinaminės analizės metodiką. Jose programos veikimas vertinamas pagal tai, ar pateiktiems duomenų rinkiniams programa pateikia teisingus atsakymus. Todėl rengiant užduotis žymią laiko dalį reiktų skirti vertinimo schemos parinkimui, gerų testinių duomenų parinkimui [74]. Jie turėtų tikrinti ne vien programos funkcionalumą su dideliais ar mažais duomenų rinkiniais. Duomenų sudėtingumas didėjant testo numeriui turėtų kisti eksponentiškai – taip parinkti duomenys demonstruoja sprendimo efektyvumą. Naudojant pastovius testų rinkinius daug metų iš eilės iškyla problemų: testų efektyvumas krinta, nes mokinių kartos keičiasi tarpusavyje duomenimis. Todėl svarstytina bent nesudėtingų testų

dinaminio generavimo idėja.

Norint naudoti automatinio programų vertinimo sistemas Lietuvos mokyklose reiktų daugiau dėmesio skirti pasirenkamos sistemos mokomosios funkcijos egzistavimui. Gerai susistemintas užduočių bankas, galimybė susitvarkyti savo užduočių rinkinį tikrai svarbi mokymui.

Didelė dalis aptariamų sistemų neturi numatytos lokalizavimo galimybės. Šiuo aspektu daug vilčių kelia EduJudge projektas, kuriame yra numatyta ne tik aplinkos vertimas, bet ir sąlygų keliomis kalbomis galimybė. Aplinkos pranešimai išversti, tačiau kai kur pamatomi „Moodle“ vertimo trūkumai.

Sistemos vertę tikrai pakelia pusiau automatinio vertinimo galimybė. Nei dinaminė, nei statinė analizė šiuo metu dar neturi tokio pajėgumo, kad galėtų įvertinti programas visais aspektais. Todėl svarbi galimybė mokytojui ne tik stebėti, bet ir keisti mokinio darbo vertinimą.

Apžvelgus sistemų palyginimo 2.1 lentelę matome, kad mokykloms galima rekomenduoti pasidomėti automatinio vertinimo sistemomis. Nemokamų sistemų išbandymas nieko neįpareigoja, o tarp jų tikrai yra padedančių individualizuoti mokymą. EduJudge projekto sistema neturėtų savo galimybėmis nusileisti populiariausioms komercinėms sistemoms, todėl į ją vertės atkreipti dėmesį, jei jos programinis kodas bus paviešintas.

Automatinės programų testavimo sistemos efektyvina individualizuotą mokymą, kuris būtinas dėstant programavimą. Galimybės pasitikrinti sprendimą be mokytojo įsikišimo, gauti tolimesnę užduotį, gauti pagalbinį patarimą sprendimui, gauti įvertinimą pagal sprendimui sugaištą laiką ir kitos rodo, kad šiuolaikinės sistemos dar turi erdvės tobulėjimui.

Viena svarbiausių išvadų gautų tyrime yra ta, kad nors nemažai sistemų turi galimybę išsaugoti rankinio vertinimo būdu parašytus įverčius, tačiau tyrimo atlikimo metu tik disertacijos autoriaus sukurta IT VBE PUVS turėjo interaktyvaus pusiau automatinio vertinimo galimybę.

2.6 Automatiniam vertinimui tinkamos užduotys

Apžvelgus automatines sistemas buvo taip pat ištirtos galimybės naudoti įvairius algoritmavimo uždavinius šioms sistemoms. Uždaviniai klasifikuoti remiantis keliais aspektais: pagal duomenų įvedimo bei išvedimo mechanizmus, tinkamus automatinėms sistemoms, pagal algoritmus, reikalingus jiems išspręsti, taip pat pagal užduočių kilmę (varžybas, kuriose pirmą kartą panaudotas, autorių ir pan.)

2.6.1 Užduočių klasifikavimas pagal įvedimo ir išvedimo būdą

Analizė parodė, kad populiariausi yra uždaviniai, skaitantys iš standartinio įvedimo ir rašantys į standartinį išvedimą; užduoties sąlygoje pradinių duomenų ir rezultato formatas nurodytas labai griežtai.

Kita gana populiari klasė, kai skaitoma iš nurodyto pavadinimo failo ir rezultatai išvedami į nurodyto pavadinimo failą. Kitais aspektais užduotys panašios į pirmojo tipo. Gal kiek keičia sprendimą galimybė failą skaityti kelis kartus.

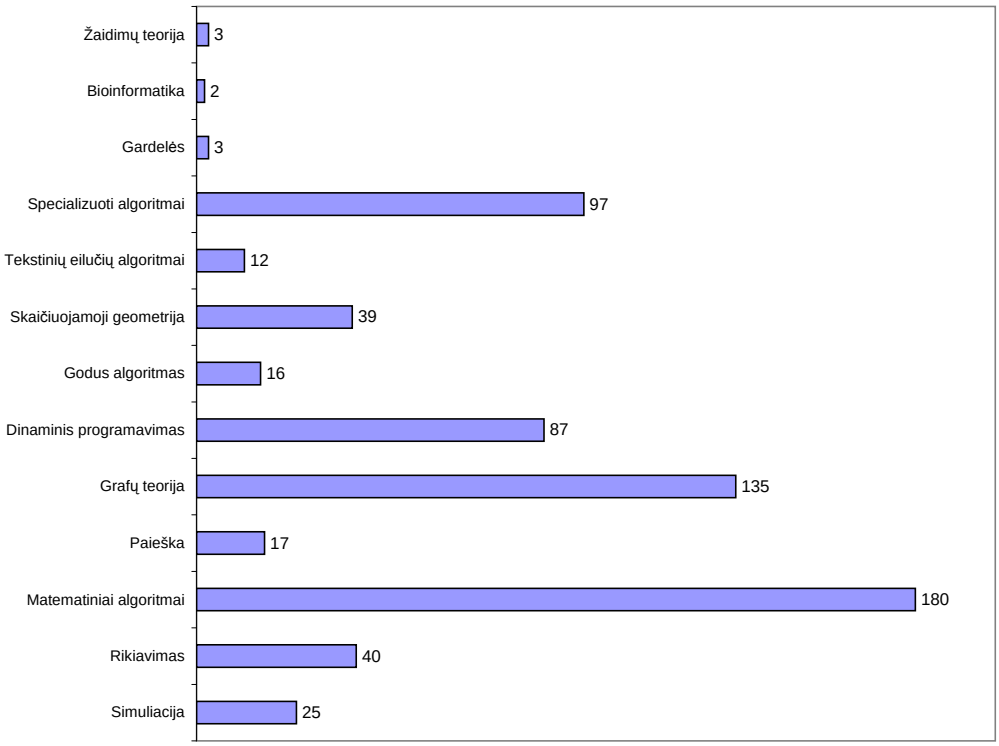
Trečias tipas užduočių – parašyta programa duomenis gauna bei rezultatus pateikia per nurodytą biblioteką, tiesioginės prieigos prie duomenų failo nėra, bibliotekos nekompiliuotas kodas nepateikiamas. Tokiu būdu programa turi dirbti su biblioteka interaktyviai. Biblioteka gali interaktyviai reaguoti į programos pateiktas užklausas, tokiu būdu realizuoti stalo žaidimus ir panašias kitas, realiam gyvenimui artimas, situacijas.

Ketvirtas užduočių tipas – mokiniui pateikiami visi pradiniai duomenys ir prašoma pateikti juos atitinkančius perskaičiuotus duomenis. Užduotis sprendžiama programa, bet pati programa, kaip užduoties sprendimas, nepateikiama.

2.6.2 Užduočių klasifikavimas pagal algoritmus

Analizuoti duomenys apie užduotis, saugomas UVA OnlineJudge. Šios automatinės užduočių duomenų bazės bei sprendimų testavimo sistemos

entuziastai yra sukūrę vikio principu veikiančią svetainę `http://www.algorithmist.com/` . Patyrinėjus joje esančius duomenis atlikta algoritmų populiarumo šiose 656 užduotyse analizė (2.6 pav.)



2.6 pav. **Užduočių klasifikavimas pagal algoritmus [77]**

2.7 Išvados

Šis analitinis automatinių ir pusiau automatinių programavimo užduočių testavimo sistemų tyrimas parodė, kad nors sistemos kuriamos jau gana seniai, didelė dalis sistemų tebeturi panašias problemas:

- sistemos tebėra netolerantiškos nesikompiliuojančioms programoms;
- sistemų pranešimai tebėra ne iki galo aiškūs užduotis sprendžiantiems;
- sistemų komponentinė struktūra yra artima, tačiau taikymo būdai skirtingi ir todėl sistemų komponentų sudėtingumas skiriasi. Nemažai sistemų turi problemų su programų testavimo modulio saugumu;

- užduočių perkėlimas tarp sistemų yra sudėtingas, nes jų saugojimas nėra standartizuotas.

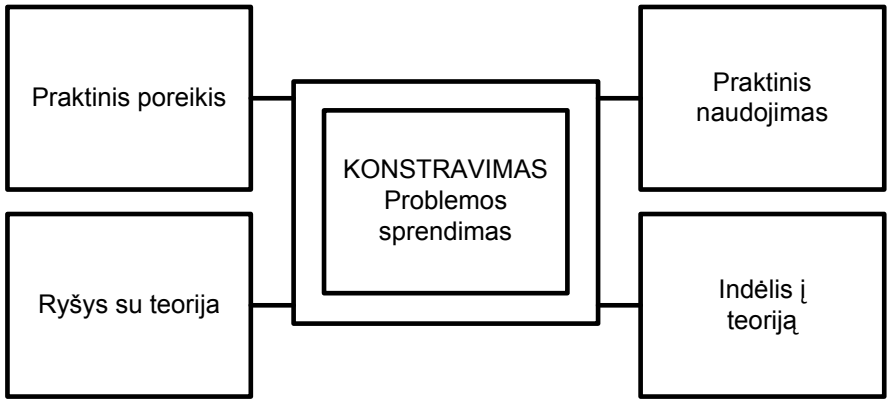
Atlikus tyrimą įsitikinta, kad automatiniam testavimui tinkamos sistemų užduotys gana panašios, daugiau skiriasi reikalingų duomenų struktūrų sudėtingumas ir reikalingų algoritmų sudėtingumas.

Viena svarbiausių šio teorinio tyrimo išvadų yra ta, kad automatinio testavimo ir rankinio vertinimo sąveika nėra ištirta.

3 Pusiau automatinio vertinimo ir testavimo metodas

3.1 Tyrimo planas

Pusiau automatinio mokinių parašytų programų testavimo tyrimui buvo pasirinktas tyrimo konstravimu metodas. Šiuo metodu siekiama sukurti novatoriškus sprendimus praktinėms ir teorinėms problemoms spręsti. Šiuo metu tai populiarus tyrimo metodas informatikoje ir informatikos inžinerijoje [72].



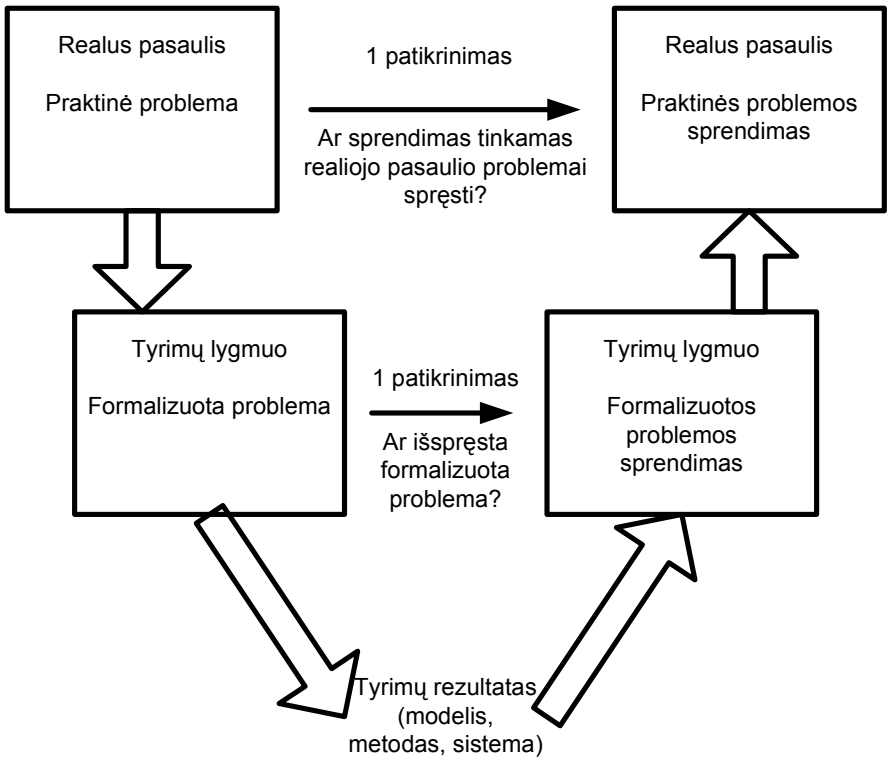
3.1 pav. **Tyrimo konstravimu komponentai [44]**

Kasanen ir kt. [44] aprašytas tyrimo konstravimu procesas schematiškai pristatomas 3.1 diagrama. Idealiu atveju tyrimo procesas turėtų vykti tokia seka:

- Atrandama problema, reikalaujanti praktinio sprendimo.
- Įsigilinama į problemos mokslinę kryptį ir kontekstą, abstrahuojami stochastiški duomenys.
- Sukuriama aibė idėjų tobulinimui bei atnaujinimui – remiamasi euristine sprendimų paieška per daug nesigilinant į teorinį jų pagrindą.

- Eksperimentiškai įsitikinama, kad sukurtas sprendimas veikia.
- Ištiriamos teorinės patobulinimo prielaidos.
- Išsiaiškinama galimybė taikyti naujus patobulinimus kitose srityse.

Praktiškai šis tyrimo metodas retai įgyvendinamas nuoseklia tvarka. Dažniau jis būna iteratyvus ar net rekursyvus. Tyrimo konstravimu schema artima pateiktai pristatytai 3.2 paveiksle.



3.2 pav. **Lassenius ir kt. [47] pristatyta tyrimo konstravimu eiga**

Disertacijoje medžiaga pateikta daugiau įprasta tvarka. Idėjų generavimas, problemos formalizavimas, metodo kūrimas ir jo eksperimentinės realizacijos tyrimas ir tobulinimas pateikti trečiajame skyriuje. Eksperimentai su metodą pilnai realizuojančia sistema bei šio metodo galimi taikymai iškelti į tolimesnius skyrius. Nors informacija pateikta nuosekliai, dalies rezultatų ir eksperimentų pateikimo tvarka neatitinka chronologinės tvarkos.

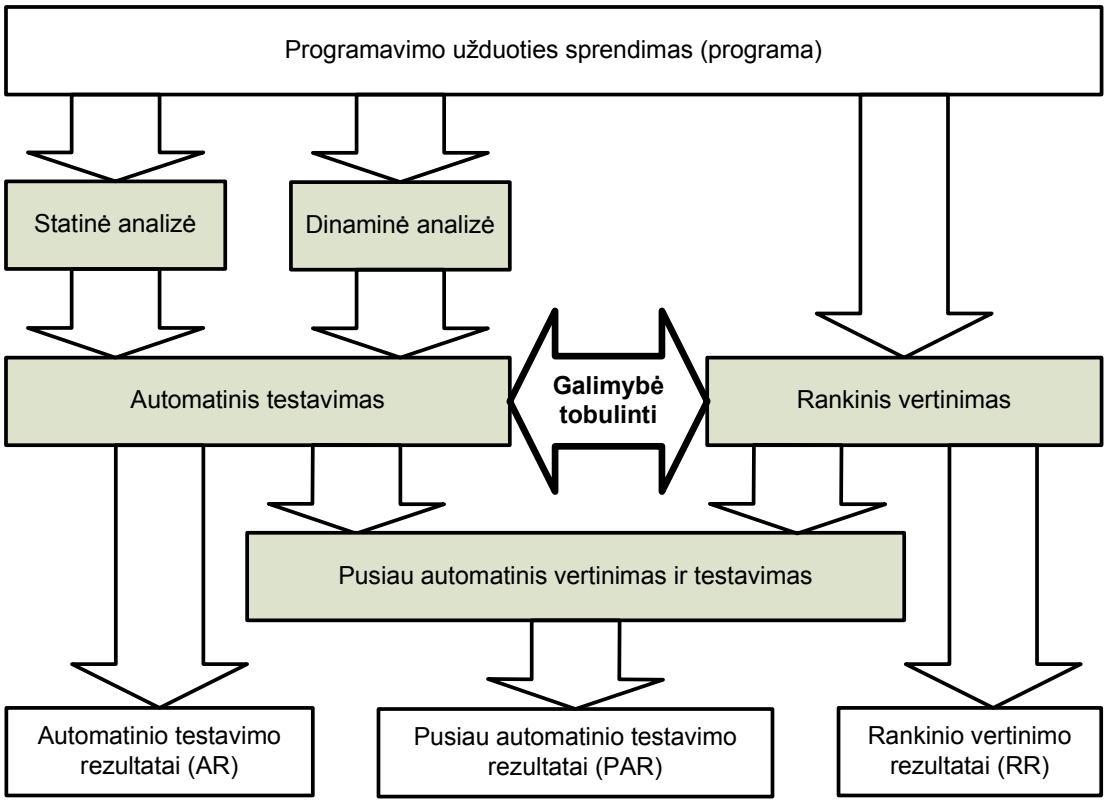
3.2 Pusiau automatinio vertinimo tobulinimo priežastys

Pusiau automatinio vertinimo ir testavimo jau pats pavadinimas suponuoja idėją, kad šiame vertinimo ir testavimo metode persipina automatinio testavimo pateikti rezultatai ir vertintojo parašyti subjektyvūs įverčiai. Tačiau šiuo metu dar nėra giliai ištirti tiek metodai, taikytini įverčių apibendrinimui, tiek pats automatinio vertinimo proceso automatinės ir rankinės dalių sąveika.

Esant poreikiui vertinti Lietuvos informacinių technologijų valstybinio brandos egzaminą buvo nuspręsta, kad abiturientų parašytos net ir nekompiliuojamos programos turi būti vertinamos. Taip pat buvo laikomasi nuomonės, kad tai turi būti pozityvus vertinimas – balai turi būti rašomi už tai, ką abiturientas pademonstravo išmokęs. Buvo aišku, kad vertinimas turi būti arba rankinis, arba pusiau automatinis. Rankinis vertinimas buvo atmestas dėl riboto vertintojų skaičiaus ir vertinimui skiriamo laiko.

Buvo iškelta idėja, kad gilesnė rankinio mokinių parašytų programų vertinimo proceso analizė gali leisti sukurti naują pusiau automatinio vertinimo ir testavimo metodą, kuris leistų pasiekti aukštesnį vertinimo efektyvumą neprarandant rankinio vertinimo kokybės.

Tokiai hipotezei formuluoti turėjo įtakos ilgas stebėjimas IT mokytojų, vertinančių programas rankiniu būdu, taip pat mokinių, kurie bando ištaisyti savo klaidas. Net ir programų, kurių ilgis neviršija 100 eilučių, kūrime yra pastebimi panašūs procesai į būdingus didelių programų sistemų kūrimui – programos modifikavimas kaitaliojasi su programos išbandymu su tam tikrais duomenimis. Stebint programos darbo rezultatus generuojamos naujos programos tobulinimo idėjos, diagnozuojama klaidų vieta, generuojami nauji testiniai duomenys. Po to vėl ateina modifikavimo fazė ir po kurio laiko vėl grįžtama prie testavimo su patobulinta

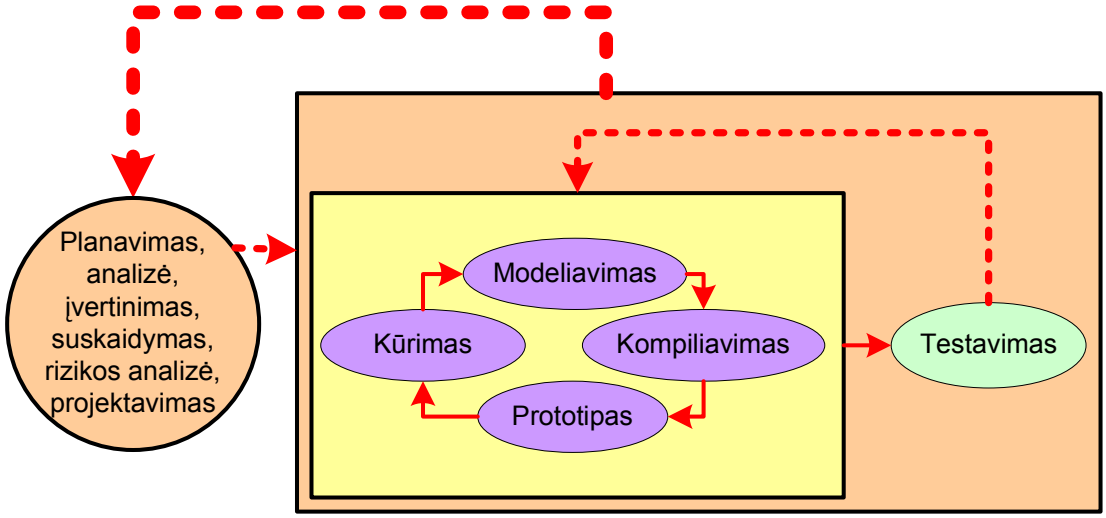


3.3 pav. **Idėja vertinimo sistemai tobulinti**

programa ar modifikuotais pradiniais duomenimis. Jei palygintume spartaus programų sistemų kūrimo ciklą su šiuo darbo metodu, atrastume daug bendro (3.4 pav.). Matyt, tai yra būdinga daugeliui nedidelių projektų.

3.3 IT VBE praktinių užduočių vertinimo problemos analizė

Apie 2005 metus kilo poreikis kurti Lietuvoje informacinių technologijų valstybinį brandos egzaminą. Egzamino kūrėjai suprato, kad reikalinga ne tik testinė dalis, bet ir praktinės programavimo užduotys [8]. Buvo įvertinta, kad iškils tam tikrų problemų įvertinti didelį kiekį abiturientų parašytų programų. Reikėjo aukštos vertinimo kokybės, galimybės įvertinti darbus greitai (egzaminų vertinimams dažniausiai skiriama maždaug savaitė). Net ir pasitelkus nemažai vertintojų darbas atrodė gana



3.4 pav. **Spartaus programų kūrimo modelio komponentas [27]**

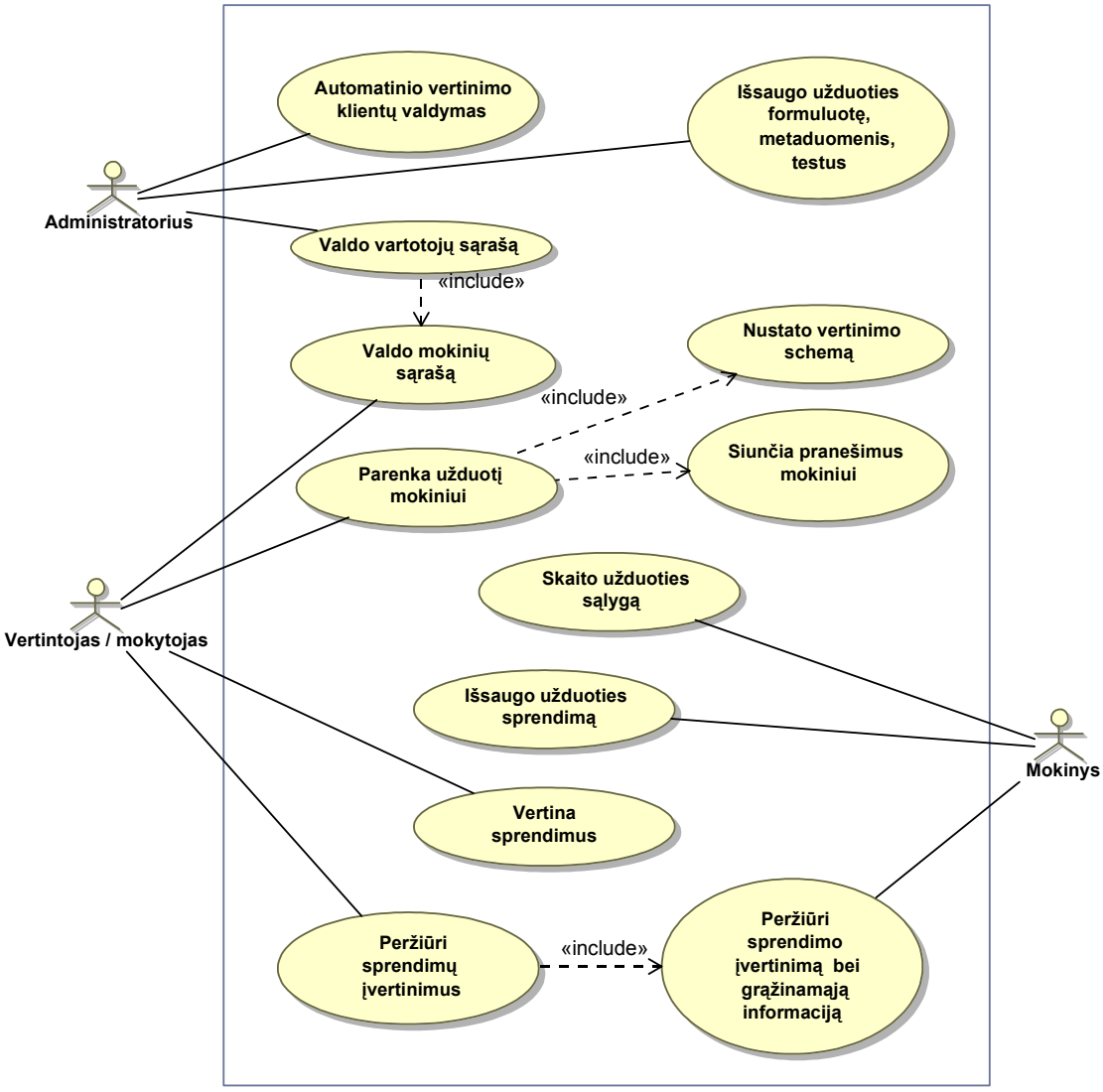
sudėtingas.

Buvo atsižvelgta į tai, kad Lietuvoje jau daugiau kaip dešimt metų vykdavo informatikos olimpiados. Nuo 2002 metų informatikos olimpiadose buvo taikomos automatinės programų testavimo sistemos. Todėl nuspręsta pasitelkti į pagalbą Lietuvos informatikos olimpiadų vertinimo komisijos atstovus, tarp kurių buvo ir šios disertacijos autorius.

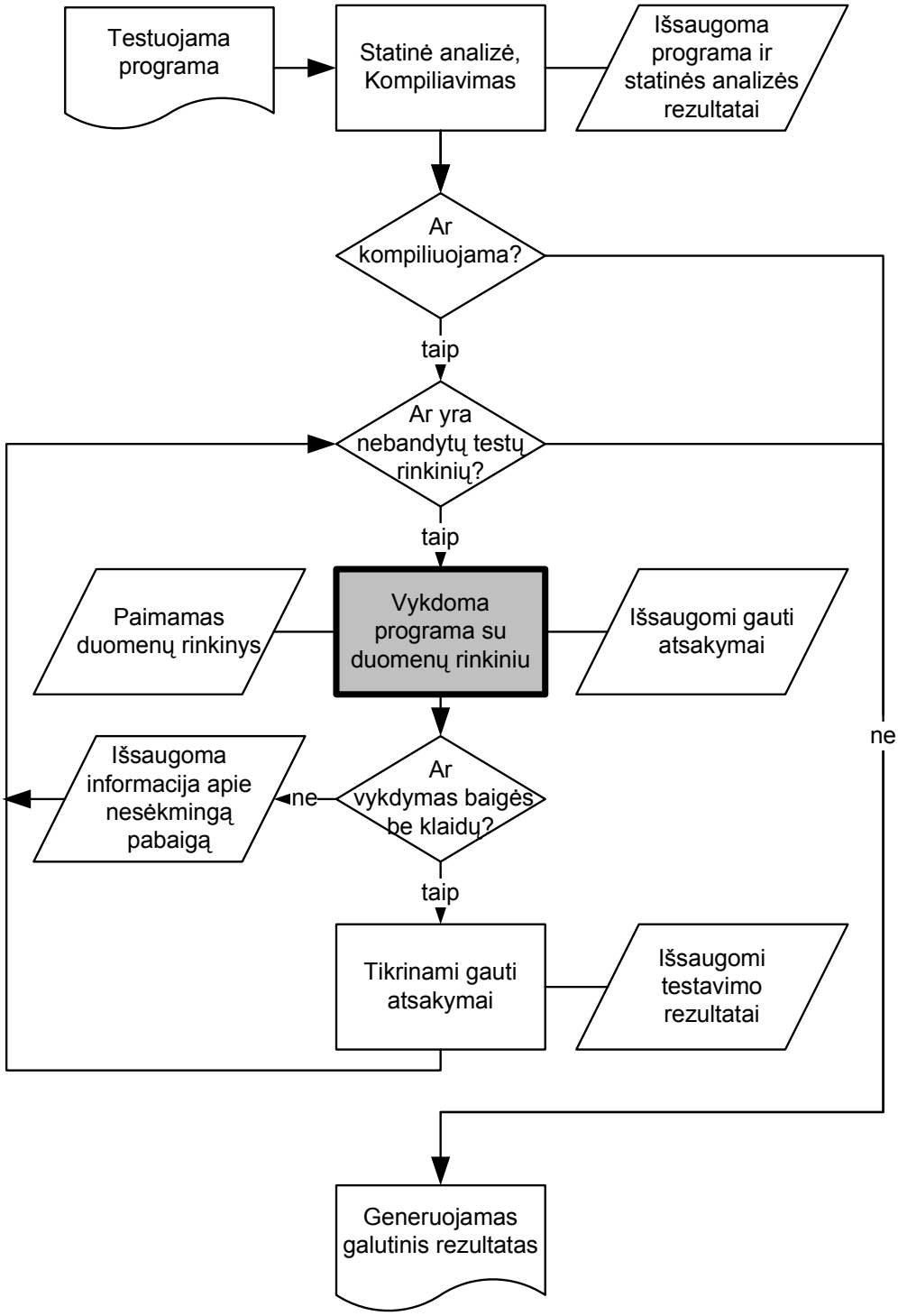
Buvo išanalizuota tipinės automatinės programų testavimo sistemos panaudos būdų 3.5 diagrama bei tipinio juodąja dėže paremto testavimo 3.6 diagrama. Įvertinus techninę užduotį, tapo aišku, kad, nors kai kurios automatinio testavimo olimpiadose savybės yra artimos, tačiau egzamino poreikiai yra specifiniai.

Išskirti šie esminiai skirtumai:

1. Reikalinga, kad testavimo sistema veiktų kiek galima panašesnėje aplinkoje kaip ir mokinių naudojamoje egzamino metu (Operacinė sistema Windows, lokalizuotas FreePascal kompiliatorius). Olimpiadose testavimas vykdomas operacinės sistemos Linux aplinkoje dėl saugumo ir naudojamų operacinių sistemų įvairovės.



3.5 pav. **Panaudos būdų diagrama tipinei automatinio programų testavimo sistemai**



3.6 pav. **Tipinė juodaja dėže paremto testavimo schema**

2. Negalima laikyti, kad uždavinys iš dalies išspręstas, jei sėkmingai įveikiami keli iš pateiktų testų. Užduotys nėra sudėtingos, todėl kiekvienos klaidos priežastys turi būti aiškios. Olimpiadose tuo tarpu yra teikiami daliniai įvertinimai, kurie remiasi skirtinga testų ar testų grupių verte.

3. Negalima vertinti labai žemais balais tų, kurių programose pasitaiko smulkių klaidų (praleistas skyrybos ženklas, neaprašytas kintamojo vardas ir t. t.). Olimpiadose laikomasi politikos, kad negalima keisti nė vieno dalyvio parašyto ženklo.

4. Egzamino vertinimo sistemoje mokinio rolė nėra reikalinga, nes mokinys neturi ir neturės galimybės stebėti vertinimo rezultatų tiesiogiai. Formalizuojant problemą šios rolės galima neakcentuoti, tačiau dauguma šiuolaikinių sistemų ją turi.

Lietuvos mokinių informatikos olimpiadose yra paplitęs automatinis paketinis vertinimas. Tuo metu vertinime buvo naudojama taškų dalinio agregavimo funkcija (kaip ir Tarptautinėse informatikos olimpiadose) – buvo teikiami daliniai taškai už įveiktus testus. Jau buvo žinomos ir pagrindinės tokio vertinimo problemos – būdavo darbų, kurie gaudavo aukštus įvertinimus, nors žymi sprendimo dalis remdavosi euristika ar atspėtu dažniau pasitaikančiu atsakymu. Taigi, nors pilnai automatinio testavimo metodas, naudojamas olimpiadose buvo patrauklus dėl jo teikiamo greito programų vertinimo, tačiau vertinimo kokybė netenkino egzamino kūrėjų.

3.4 Formalizuota pusiau automatinio vertinimo problema

Formalizuotai pusiau automatinio vertinimo sistemai reikia suformuluoti esminius funkcinius ir nefunkcinius reikalavimus.

3.4.1 Pusiau automatinio vertinimo funkciniai ir nefunkciniai reikalavimai

Apibrėžkime pagrindinius *funkcinius reikalavimus* ITVBE vertinimo sistemai.

Sistema prieš leisdama vertinti konkrečią programą turi:

- leisti administratoriui įdiegti į sistemą užduoties specifikaciją su testais;
- leisti administratoriui keisti užduoties vertinimo kriterijus;
- leisti administratoriui suformuoti ir pateikti vertintojui reikalingų vertinti programų paketą;
- leisti administratoriui pateikti programų paketą kitam vertintojui;
- leisti administratoriui stebėti darbo proceso rezultatus.

Sistema vertindama konkrečią programą turi:

- gebėti vykdyti statinės analizės įskiepius ir pateikti vertintojui rezultatus;
- gebėti automatiškai testuoti pateiktą programą panaudodama „juodosios dėžės“ testavimą. Tam ji turi gebėti sukompiliuoti pateiktą programą, parengti vykdymui saugią aplinką (kurioje programa negalėtų pakenkti sistemos stabilumui bei negalėtų neteisėtu būdu gauti teisingus atsakymus), paruošti pradinių duomenų failus, vykdyti kandidato programą tuo pačiu stebėdama vykdymo laiką, nutraukti programos vykdymą, jei ji viršija jai skirtus atminties ir vykdymo laiko ribojimus, vykdyti programos pateiktų atsakymų vertinimo programas, kaupti vertinimo rezultatus;
- pateikti automatinio testavimo rezultatus bei vertinamą programą vertintojams;
- leisti modifikuoti vertinamą programą vertintojams bei eksperimentiškai pertestuoti nekeičiant automatinio testavimo rezultatų, bet

parodant įveiktus testus; vizualizuoti padarytus keitimus, leisti atstatyti į pradinį variantą;

- leisti pildyti/keisti pusiau automatinio testavimo įvertinimus, jei automatinis testavimas nesuteikė balų;
- visada leisti pildyti/keisti rankinio testavimo įvertinimus.

Įsigilinus į pradinius funkcinius reikalavimus, galima pastebėti, kad administratoriaus rolės veiksmai turėtų būti automatizuoti, tačiau pagrindinės problemos – vertinimo kokybės ir vertinimo spartos ypatingai nekeičia. Todėl formalizuojant problemą šios rolės galima atsisakyti paliekant tik prielaidą, kad administratorius parengia visus pradinius duomenis – vertinimo kriterijus, užduočių testų ir jų atsakymų rinkinius, vertinimo programas parengia iš anksto ir vertinimo sistema tiesiog jais naudojasi.

Formalizuotame uždavinyje *nefunkciniai reikalavimai* nebus akcentuojami, nes jų realizavimas nėra tiek abstraktus. Pakartosime tik pagrindines siekiamybes.

Pusiau automatinė sistema turi:

- vertinti tolerantiškai esant smulkiems netikslumams;
- padėti diagnozuoti tikslią programos klaidos vietą;
- pateikti aiškius pranešimus žmogui–vertintojui;
- veikti sparčiai;
- sudaryti sąlygas vertintojui patogiai eksperimentuoti su vertinama programa.

3.4.2 Sistemos keliami reikalavimai egzamino specifikacijai

Kiekviena automatinio ir pusiau automatinio vertinimo sistema kelia tam tikrus reikalavimus renginiui, programavimo užduočių sąlygų autoriams, renginio vykdytojams. Tai nėra sistemų ribotumo pasekmė, bet greičiau sistemose naudotų priemonių pasekmė. Prieš taikant automatinį

ar pusiau automatinį testavimą turi būti įvykdyti įvairūs parengiamieji darbai, suformuoti duomenų masyvai.

Kompiuteriai, OS, kompiliatoriai, aplinkos

Norint kokybiškai palyginti mokinių žinias, reikia siekti, kad mokinių darbo priemonės (tiek techninė įranga, tiek naudojama programinė įranga) esmingai nesiskirtų.

Dažnai konkursuose mokiniai skundžiasi ne tokiais sparčiais kompiuteriais. Ypač tai būdinga, jei yra skirtingų kompiuterių klasėje ar mokymo įstaigoje. Nors programos teksto rinkimui kompiuterio sparta dažnai neturi itin didelės reikšmės, bet jos svarba ryškesnė kompiliuojant ir išbandant programas, ieškant pagalbos failuose ar naršant internete. Taip pat kartais tenka stebėti, kad modernesnių programavimo aplinkų automatinio teksto pratęsimo funkcija ne tokiuose sparčiuose kompiuteriuose suveikia lėtai ir net trukdo rinkti tekstą.

Ypatingai svarbu, kad mokinių naudojama darbo aplinka būtų kiek galima artimesnė testavimo aplinkai. Pavyzdžiui, esama patirties Lietuvos informatikos olimpiadose, kai dalis mokinių programuoja naudodami Windows OS, o testavimas vykdomas Linux OS. Nors programavimo kalbų kompiliatoriai sutampa, stebimos situacijos, kai dėl OS realizacinių skirtumų programos, turinčios smulkių klaidų, veikia labai skirtingai. Skirtumai pasireiškia tiek pateiktais skirtingais rezultatais, tiek vykdymo sparta ar net gebėjimu sėkmingai užbaigti vykdymą. Svarbios ir skirtingos kompiliatorių versijos. Dažnai mokymui naudojami atvirojo kodo kompiliatoriai, kurių skirtingose versijose pasitaikančios klaidos lemia skirtingą veikseną. Taip pat svarbus ir optimizavimo aspektas - modernesni kompiliatoriai turi patobulintas programos kodo optimizavimo funkcijas, kurių vykdymo pasėkoje mokinio programos veikimo sparta gali esmingai skirtis.

Taip pat labai svarbu ir aiškiai apibrėžti, kokie kompiliatoriai, kokios kompiliatorių nuostatos bus naudojamos testuojant programas. Gera šių nuostatų svarbos iliustracija galėtų būti FreePascal kompiliatorius, kurio nuostatos gali perjungti Pascal programavimo kalbos poaibį. Šiuose skirtinguose kalbos poaibiuose skirtingai apibrėžtas tekstinės eilutės bei sveikojo skaičiaus tipai, yra skirtumų rekursinių funkcijų apibrėžimuose. Dėl šių priežasčių nenurodytos kompiliavimo nuostatos gali lemti, kad mokinio programa, sėkmingai veikianti jo darbo vietoje, testavimo aplinkoje gali būti net nekompiliuojama.

Užduočių sąlygos

Prieš mokiniams pateikiant užduočių sąlygas, reikia būti įsitikinus, kad kiekviena užduotis tenkina šiuos reikalavimus:

- užduotyje yra nusakytos galimos pradinių duomenų ir rezultatų reikšmių aibės;
- užduotyje yra aiškiai apibrėžti tiek duomenų įvedimo/išvedimo srautai, tiek įvedimo/išvedimo duomenų formatai;
- yra žinomi galimi užduoties sprendimo būdai ir kompiuterio skaičiavimo laikas visiems galimiems pradiniams duomenims yra priimtinas;
- yra parengti duomenų rinkiniai (testai), kurie gali pademonstruoti sprendimo teisingumą įvairiais atvejais;
- yra žinomas būdas užduoties sprendimo rezultatams patikrinti. Paprasčiausia juos patikrinti, jei kiekvienam pradinių duomenų rinkiniui yra galimas tik vienas teisingas žinomas rezultatas. Kitais atvejais reikalinga programa, gebanti iš pradinių duomenų, pateiktų rezultatų ir papildomų turimų duomenų priimti sprendimą apie rezultato teisingumą;

- yra parengta vertinimo instrukcija, kurioje yra apibrėžta, kas bus vertinama rankiniu būdu ir bent preliminariai nurodyta, kokiu būdu testavimo rezultatai bus apjungti su rankinio vertinimo rezultatais.

Programų tekstai, programų failų vardai

Testavimui pateikiamos programos turi tenkinti reikalavimus, kurie turi būti aiškiai išdėstyti prieš prasidedant užduočių sprendimui. Daugu-ma reikalavimų gali būti apibrėžti užduočių sąlygose, tačiau dalis jų turi būti žinoma iš anksto. Tokie reikalavimai gali būti susiję su konkrečia naudojama testavimo sistema. Galimų reikalavimų pavyzdžiai:

- programos tekste turi būti pateikiama informacija apie sprendimo autorių (mokinio identifikatorius, klasė, mokykla, amžiaus grupė ir pan);
- programos tekste turi būti pateikiama informacija apie užduotį (užduoties identifikatorius, kuri užduoties dalis išspręsta ir pan);
- programos tekste turi būti pateikiama informacija apie naudotą programavimo kalbą (kompiliatoriaus identifikatorius, gali būti nurodomi kompiliavimo raktai);
- gali būti keliami reikalavimai, kad mokiniai (ar mokytojai) turėtų paskyras specialiame tinklapyje, kuriame bus pateikiamos sąlygos, pradinių duomenų pavyzdžiai, bus pateikiami mokinių sprendimai, gaunami testavimo rezultatai;
- gali būti keliami reikalavimai programų failų pavadinimams, pradinių duomenų bei rezultatų failų pavadinimams.

Testiniai duomenys ir atsakymai

Rengiantis automatiniam vertinimui, reikia parengti testinius duomenų rinkinius bei vertinimo programą, vertinančią užduočių sprendimų pateiktus atsakymus.

Testiniai duomenų rinkiniai turi būti:

- pakankamai trumpi ir aiškūs, suprantami vertintojams;
- tiksliai identifikuoti galimus blogus užduoties sprendimus;
- demonstruoti programos funkcionavimą įvairiose situacijose.

Alternatyviojo vertinimo ir pusiau automatinio vertinimo kriterijai

Labai svarbi tinkamai paruošta ir patikslinta vertinimo schema. Tačiau pasirengti alternatyviajam vertinimui nėra paprasta, nes:

Kriterijai turi būti:

- aiškūs, vienareikšmiškai suprantami vertintojams;
- tiksliai atitinkantys galimus užduoties sprendimus;
- numatantys taškus už kandidato pademonstruotas žinias.

Taškai už kriterijus turi:

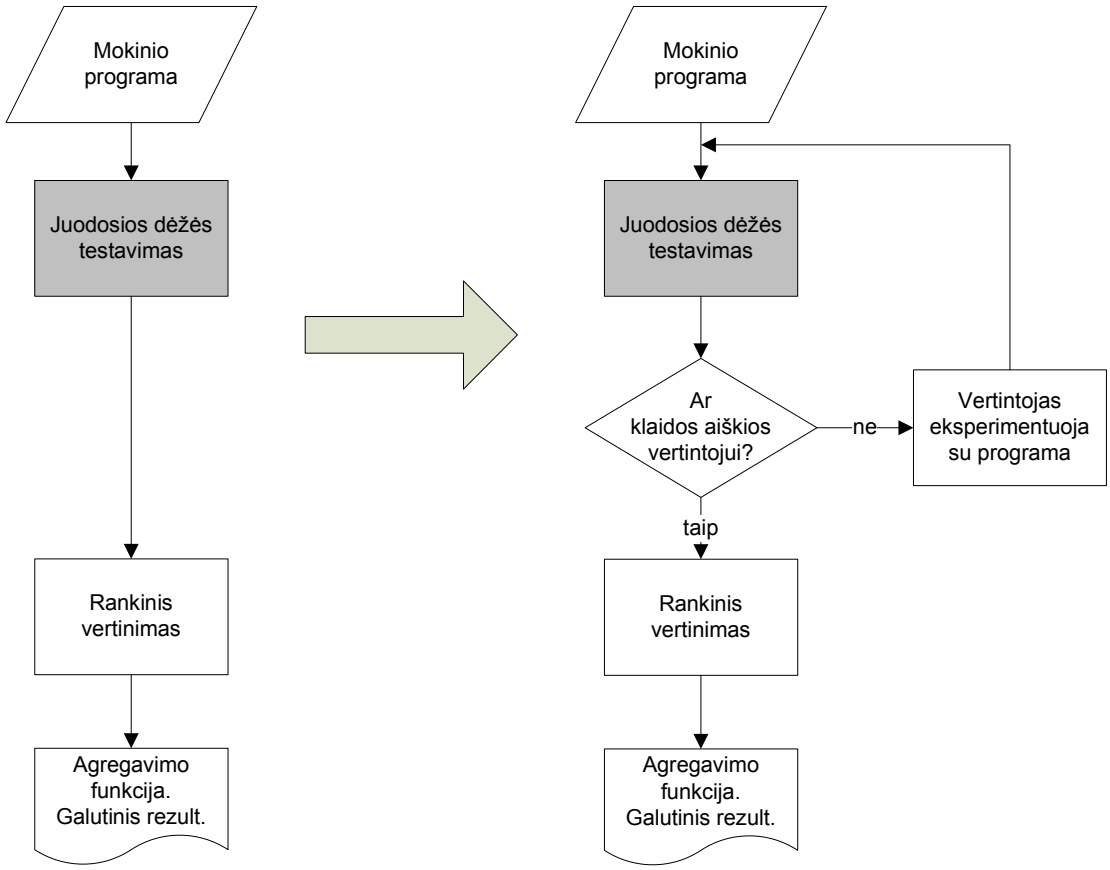
- atitikti egzamino matricą;
- būti proporcingi žinių svarbai programavimui.

Dėl šių priežasčių dažnai vykdomi žvalgomieji vertinimai, kurie leidžia patikslinti vertinimo schemas prieš pradedant vertinti darbus pilnu pajėgumu.

3.5 IT VBE praktinių užduočių vertinimo sistemos prototipas

Formalizavus problemą ir atsiribojus nuo didelės dalies įprastų vertinimo sistemos vartotojų rolių (paliktas tik vertintojas) buvo kuriamas pusiau automatinio vertinimo sistemos branduolys-prototipas, kuris realizavo tik vienos programos vertinimą. Sistemoje buvo realizuojami saugus „juodosios dėžės“ testavimas, programos peržiūra, rankinio vertinimo įverčių surinkimas.

Atlikus keletą eksperimentų su naujai kuriama sistema buvo padaryta išvada, kad vertintojams sunku iš pirmo karto surasti programoje klaidą.



3.7 pav. **Pusiau automatinio programų testavimo proceso patobulinimas**

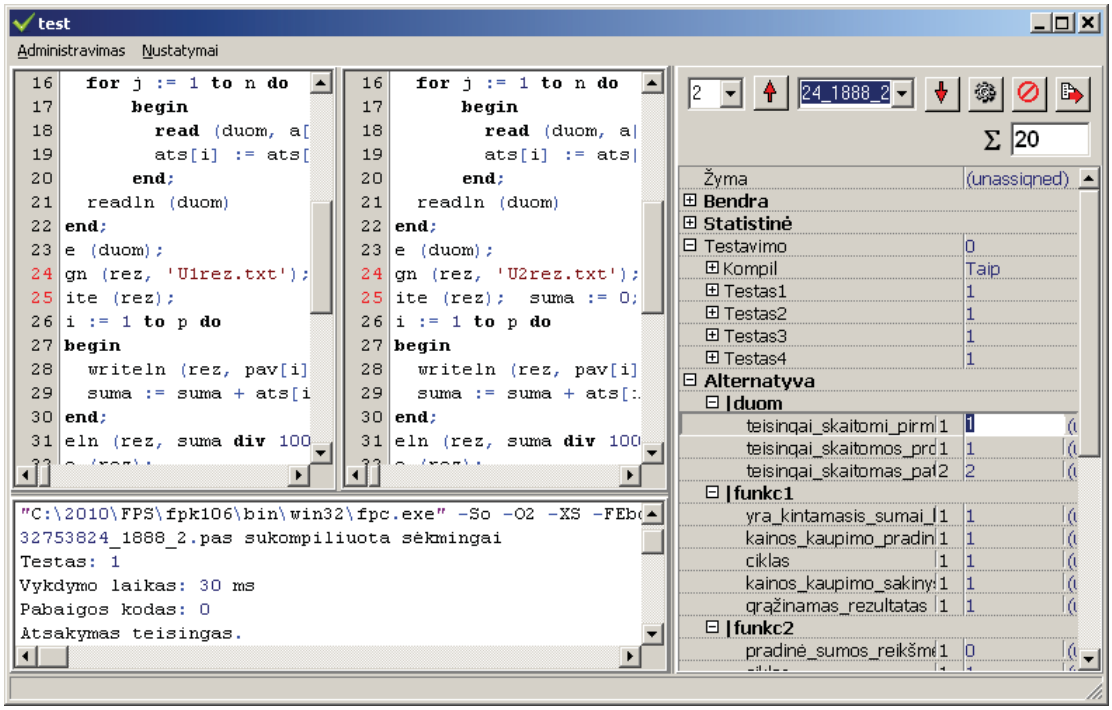
Tokiu atveju vertintojai dažnai perkeldavo programą į įprastinę programavimo aplinką ir pradėdavo eksperimentuoti joje. Tačiau patikrinti programos veikimą su testų grupe įprastinėje programavimo aplinkoje nėra patogu. Taip kilo mintis dalį programavimo aplinkos funkcijų perkelti į testavimo sistemą. Pirmasis ir esminis patobulinimas buvo labai paprastas – teksto peržiūros komponentas buvo pakeistas teksto redagavimo komponentu ir pridėtas mygtukas leidžiantis pakartotinai pertestuoti modifikuotą programą. Vertinimo proceso patobulinimo esmę gerai atspindi 3.7 paveikslas.

Šis sistemos pataisymas vertintojams patiko, tačiau beveik iš karto pasijuto keletas naujų problemų – vertintojai nebeprisimindavo kas buvo pataisyta. Testavimo rezultatai taip pat būdavo jau pasikeitę. Buvo padaryta išvada, kad visada reikia turėti originalų programos tekstą ir sekti vertintojo padarytus pakeitimus. Taip pat buvo nuspręsta, kad sistemoje reikalingi du lygiagretūs langai su pradiniu ir pataisytu programos tekstu. (3.8 pav.).

Nuspręsta pradinio programos automatinio testavimo rezultatus saugoti atskirai. Tokiu būdu eksperimentinio testavimo rezultatai atskirti nuo vertinimo proceso. Jie teikia informaciją apie atliktų programos pakeitimų įtaką jos funkcionalumui, tačiau pakeitimų svoris ir įtaka pusiau automatiniam vertinimui galima tik įtakojant vertintojo apsisprendimą.

3.9 paveiksle pateikta sistemos koncepcinė klasių diagrama, kurioje aiškiai matomi atskirti programos automatinio testavimo ir modifikuotos programos testavimo rezultatai.

Iteratyviai tobulinant prototipą paaiškėjo, kad vertintojams reikalingi aiškesni pranešimai apie priežastis, kodėl programos pateiktas atsakymas nėra teisingas. Susidurta su įprastinių atsakymo vertinimo programų netobulumu. Įprastinės programos dažnai identifikuodavo tik eilutę, kurioje atsakymas nesutampa su teisingu vienareikšmiu atsakymu. Padaryta

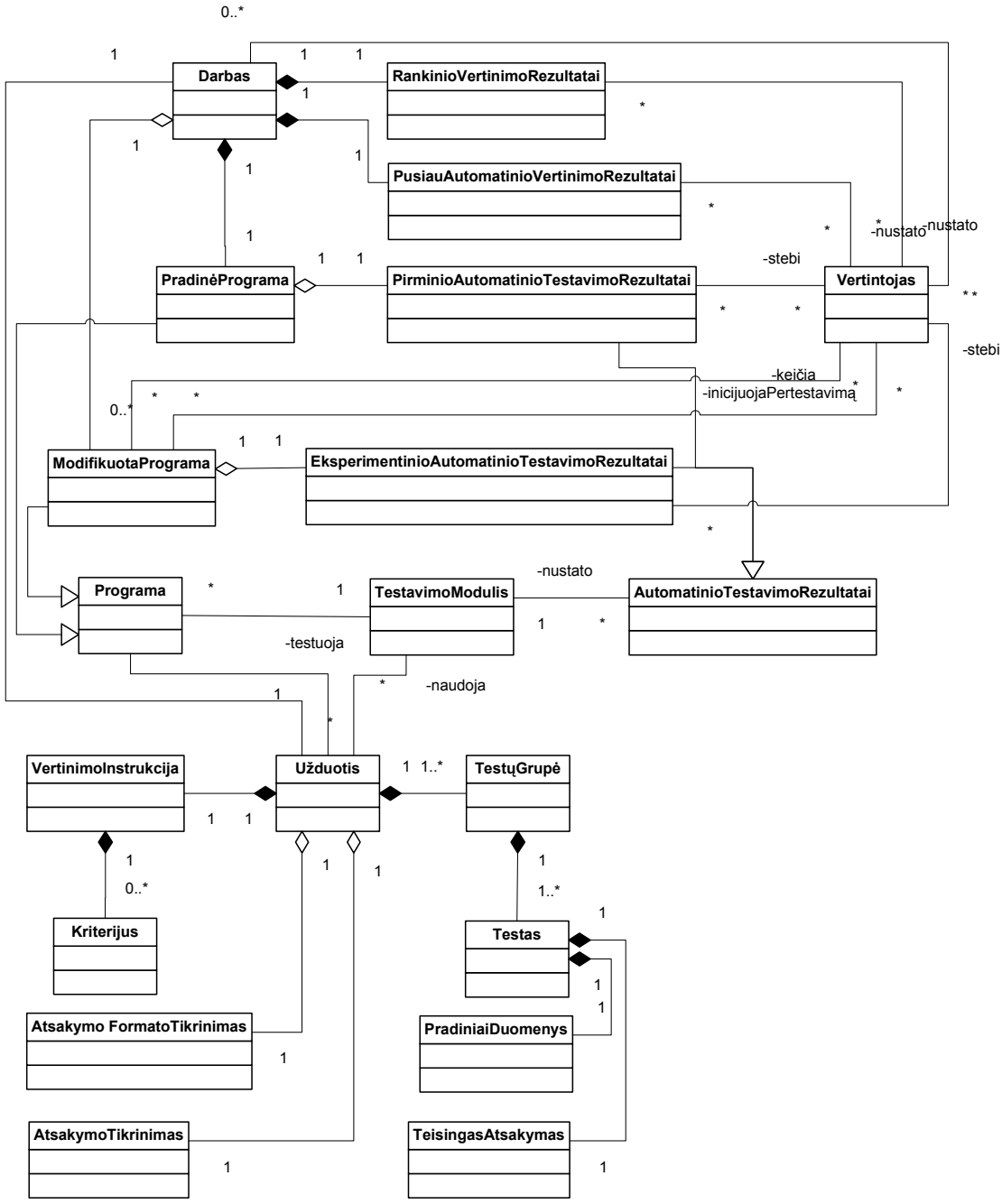


3.8 pav. **IT VBE praktinių užduočių vertinimo sistemos ekrano vaizdas. Matomi originalus ir pataisytas programos tekstai**

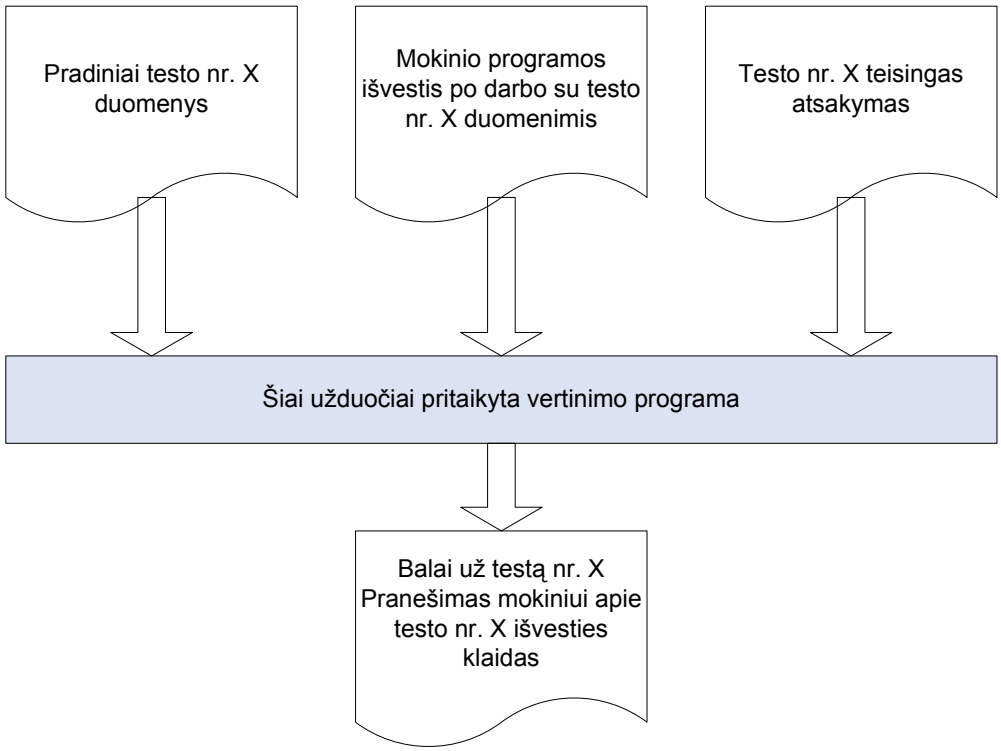
išvada, kad norint tiksliau diagnozuoti klaidos vietą, reikia tikrinti programos pateikto atsakymo formato atitikimą formatui, nurodytam užduoties sąlygoje. Tam geriausiai tinka leksinis analizatorius, sukurtas panašiais principais kaip ir kompiliatoriuose. Tačiau kiekvienai užduočiai kurti leksinį analizatorių pasirodė per sudėtinga. Buvo nuspręsta, kad atsakymo vertinimo programą galima išskaidyti smulkesniais komponentais, deleguojant vertinimą kitoms dviems programoms: atsakymo formato tikrinimo programai bei atsakymo teisingumą tikrinančiai programai (supaprastintai) (3.10 ir 3.11 pav.)

Pakeitus reikalavimus leksinis analizatorius galėjo būti vienas su jam suformuojamais konfigūraciniais duomenimis. Supaprastėjo ir atsakymų tikrinimo programa, nes, jei būdavo galimas teisingas atsakymas vienareikšmis, tai dažniausiai užtekdavo palyginti failus, ignoruojant pasikartojančius tarpus.

Sistema buvo pildoma statinės analizės įskiepiais, kurių dalis buvo



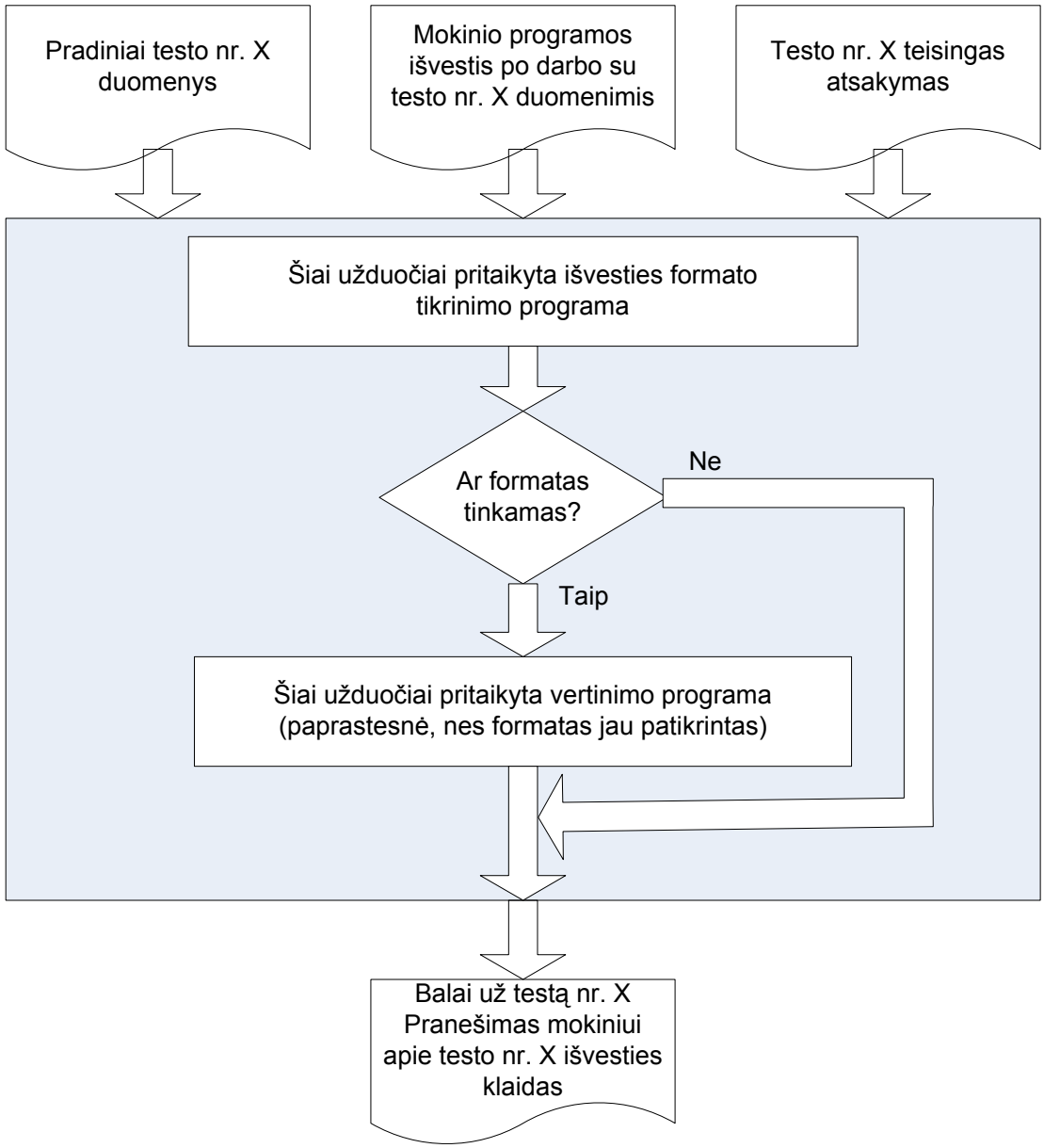
3.9 pav. **IT VBE praktinių užduočių vertinimo sistemos prototipo klasių diagrama**



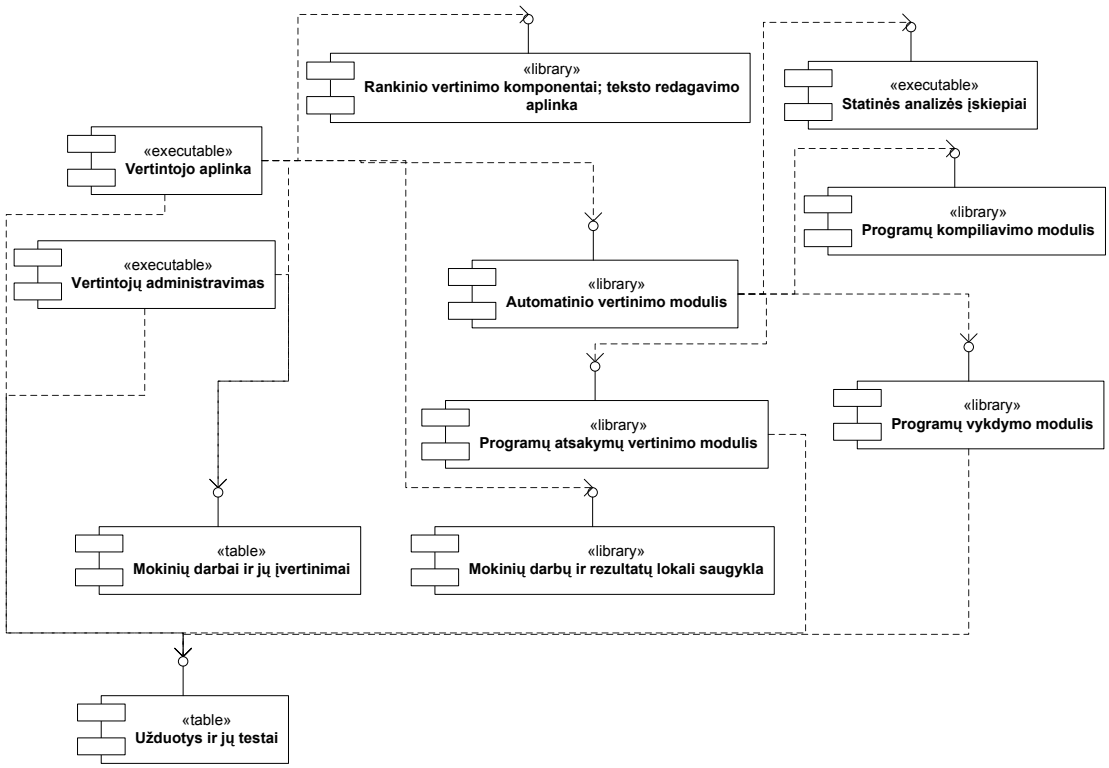
3.10 pav. **Įprastas juodosios dėžės testavimo metodo komponentas: atsakymo vertinimo programa**

skirta programavimo stiliaus automatiniam vertinimui. Po kelių programos tobulinimo iteracijų jos komponentinė schema yra nusistovėjo tokia, kaip pateikta 3.12 paveiksle.

Tolimesnis sistemos tobulinimas buvo siejamas su daugiapakopiu vertinimu ir rezultatų apibendrinimu. Sistemos duomenų srautai buvo numatyti panašūs kaip daugumos Lietuvos valstybinių egzaminų. Nustatytas vertinimo principas, kuriame nurodyta, kad kiekvieną sprendimą turi įvertinti ne mažiau dviejų vertintojų. Jei šių vertintojų balai esmingai skiriasi, skiriamas trečiasis vertintojas. Vertinimo metu vertintojai neturi žinoti kito vertintojo balų. Sistemos duomenų srautų diagrama pateikta 3.13 paveiksle.



3.11 pav. **Patobulintas juodosios dėžės testavimo metodo komponentas: atsakymo vertinimo programa, deleguojanti vertinimą kitoms dviems programoms**

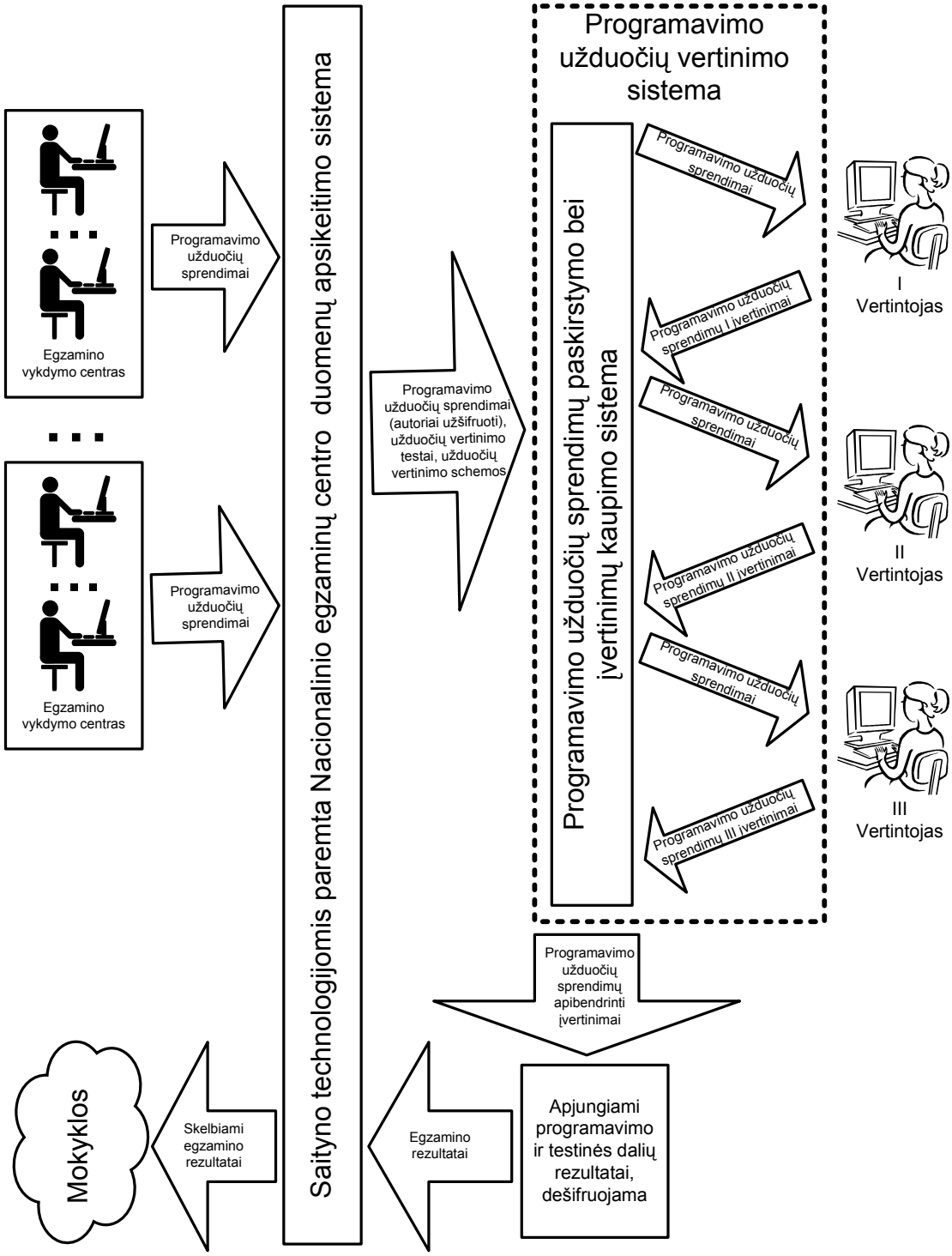


3.12 pav. **Komponentinė vertinimo sistemos prototipo schema**

3.6 IT VBE praktinių užduočių vertinimo sistemos kaita

Iki 2011 metų IT VBE programa reikalavo praktines užduotis atlikti Paskalio programavimo kalba [56]. Tai ribojo mokinių galimybes rinktis kitą programavimo kalbą informacinių technologijų programavimo modulyje, nors jau 2002 metais parengti Lietuvos bendrojo lavinimo mokyklos bendrosios programos ir bendrojo išsilavinimo standartai XI-XII klasei tam neprieštaravo [2].

2010 metais Švietimo ir mokslo ministro įsakymu patvirtintas IT VBE programos pakeitimas: „Praktinei egzamino užduoties daliai atlikti naudotinos programavimo aplinkos Free Pascal, CodeBlocks ir Dev C++.“ [3]. Šis pakeitimas leidžia mokiniams praktines užduotis atlikti ne vien Paskalio, bet ir C bei C++ programavimo kalbomis.



3.13 pav. **IT VBE programavimo užduočių vertinimo sistemos duomenų srautų diagrama**

3.6.1 Pasikeitimų įtakos tyrimas

Pasikeitusi IT VBE programa turėjo įtakoti pakitusias vykdymo ir (numanomai) vertinimo instrukcijas. Tačiau šie pasikeitimai neapsiriboja vien nurodymų ar instrukcijų pakeitimu. Programavimo kalbų pasirinkimo galimybė įtakojo vertinimo sistemos adaptavimo pakitusiai situacijai darbus. Taip pat tai kelia žmogiškojo resurso – vertintojų parengimo vertinimui bei vertinimo kokybės klausimus. Prieš egzaminą buvo stengiamasi išanalizuoti galimas problemas ir pateikti galimus jų sprendimus. Egzaminui įvykus buvo patikrinta, kiek prognozės atitiko egzamino realijas.

Automatinių sistemų mokinių parašytoms programoms testuoti yra gana daug. Jos dažnai yra daugiakalbės – jas galima naudoti įvairiomis programavimo kalbomis parašytoms programoms testuoti. Tačiau tai daugiausia būdinga būtent dinaminę analizę ir konkrečiai juodosios dėžės metodą naudojančioms sistemoms. Kitokių sistemų adaptavimas kelioms programavimo kalboms reikalauja gilios kiekvienos pridedamos programavimo kalbos analizės ir gana sudėtingų papildomų programavimo darbų, todėl toks adaptavimas vykdomas retai. Pusiau automatinių vertinimo sistemų mokinių parašytoms programoms pritaikomumas kelioms programavimo kalboms nėra ištirtas. „Daugiakalbių“ programavimo varžybų pasitaiko gana nemažai, tačiau „daugiakalbiškumas“ programavimo egzaminuose yra retas. Todėl buvo tiriamos jos tobulinimo galimybės pasikeitus IT VBE vykdymo reikalavimams.

Buvo atliktas empirinis tyrimas, kuriame nagrinėtos teorinės galimybės pritaikyti dabartinę IT VBE pusiau automatinę mokinių programų testavimo sistemą pridedant papildomas kalbas. Atlikta IT VBE sistemos analizė ir modifikavimas. Ištirti Lietuvos mokinių olimpiados nugalėtojų programavimo kalbų pasirinkimai bei atlikta kelių pasirinktų mokinių apklausa. Įvertintos teorinės „dvikalbystės“ įvedimo egzamine sukeltos

grėsmės vertinimo kokybei. Parengta metodika šių grėsmių diagnozavimui bei galimam mažinimui.

3.6.2 Pasikeitimų įtakos tyrimo rezultatai

Kelių paskutinių metų informatikos olimpiadų patirtis rodo, kad didžioji gerai pasirodančių mokinių dalis migruoja iš Pascal kalbos link C++. Atlikta mokinių apklausa parodė, kad mokiniai palankiai žiūri į C++ dėl įvairių priežasčių: populiari profesionali Microsoft programavimo aplinka Visual C++; greitai tobulėjančios nemokamos programavimo aplinkos bei kompiliatoriai; galingos bibliotekos; didelis panašios sintaksės kalbų populiarumas (Java, C#, Javascript, PHP); populiarėjanti su C kalba sietina Linux operacinė sistema. Kai kurie mokiniai nurodė, kad C kalbą išmoko net anksčiau nei Pascal ir pastaroji jiems nepatraukli. Apklausoje pasitaikė mokinių, kurie tvirtino, kad pasistengs per artėjančius metus mokytis C++, nes „ji olimpiadoje leidžia pasinaudoti standartiniais algoritmais“. Panašu, kad IT VBE atliepia šių mokinių poreikiams.

Atlikus IT VBE mokinių programų vertinimo sistemos analizę (3.12 pav.) buvo padaryta išvada, kad sistema gana glaudžiai susijusi su Pascal programavimo kalba:

- ji neskirsto darbų pagal programavimo kalbą;
- teksto peržiūros ir modifikavimo sistemoje naudoja Pascal kalbos sintaksės paryškinimus;
- yra glaudžiai susieta su FreePascal kompiliatoriumi;
- teksto išdėstymo įvertinimas daromas specialiu statinės analizės įskiepiu, kuris netinkamas kitoms programavimo kalboms.

Po preliminarios sistemos analizės buvo atlikta visos sistemos išeitinio kodo bei jos modulių analizė. Buvo identifikuoti stipriai susieti su Pascal programavimo kalba moduliai:

- mokinių programų saugyklos modulis turi saugoti mokinio programoje panaudotą programavimo kalbą, taip pat turi būti galimybė saugoti daugiau nei du išvesties failus, nes C++ rašytos programos pranešimus apie sistemines klaidas nukreipia į stderr standartinės išvesties įrenginį;

- vartotojo sąsajoje esanti originalaus mokinio programos teksto peržiūros ir modifikavimo sistema. Ji sudaryta iš dviejų tekstinių langų, tekstas yra spalvinamas, pagal programavimo kalbos sintaksę paryškinami baziniai žodžiai. Naudojamas teksto redagavimo komponentas iš bibliotekos, kurį galima pritaikyti kelioms programavimo kalboms;

- mokinio programos kompiliavimo modulis. Svarbu naudoti programavimo kalbą atitinkantį kompiliatorių, todėl šio modulio modifikavimas sukelia poreikį modifikuoti ir nuostatų modulį, kuriame būtų numatytas informacijos saugojimas apie sistemai pasiekiamus kompiliatorius. Kompiliatorių diagnostiniai pranešimai gana skirtingi, todėl reikia keisti diagnostinių pranešimų analizės mechanizmą;

- mokinio programos vykdymo modulis. Pascal ir C++ skirtingai praneša sistemai apie įvykusią vykdymo klaidą. Taip pat C++ rašytos programos efektyviai naudoja du išvedimo srautus;

- mokinio programos statinės analizės įskiepiai turi būti vykdomi skirtingi priklausomai nuo programavimo kalbos.

3.6.3 IT VBE vertinimo sistemos modifikavimas

Modifikuojant vertinimo sistemą, stengtasi atlikti tik keitimus, kurie nepakeistų vertintojo sąsajos iš principo. Taip pat stengtasi, kad visas buvęs sistemos funkcionalumas būtų išlaikytas. Vertintojo sąsaja išliko beveik nepakeista, nes dauguma pakeitimų daryta ne joje.

Atliekant sistemos modifikavimo darbus, didžiausius sunkumus sukėlė statinės analizės įskiepiai, nes daugelio požiūriu jie labiausiai priklauso nuo programavimo kalbos ir turi mokinio programai susikurti sintaksinį medį, jį analizuoti, rasti sintaksės galimas klaidas ir t.t.

Dalies įskiepių atsisakyta dėl ribotų galimybių juos naudoti įvedus kelias kalbas. Tokio įskiepio pavyzdys – teksto išdėstymo tvarkingumo vertinimo įskiepis. Viena tokio pasirinkimo priežasčių – C++ kalboje yra daug paplitusių teksto dėstymo būdų ir sunku apibendrinti, kas joms visoms bendra išskyrus sistemingumą. Antra vertus, nemažai programavimo aplinkų jau realizuoja automatinį teksto dėstymo sutvarkymą. Todėl atsisakius automatinio įrankio teksto dėstymo kokybei vertinti, jei šis kriterijus bus vertinimo schemoje, jį teks vertinti vertintojams rankiniu būdu.

3.6.4 Įvertintos teorinės „dvikalbystės“ įvedimo IT VBE sukeltos grėsmės vertinimo kokybei

Mokinių programų vertinimo kokybė priklauso nuo vertintojų kvalifikacijos

C++ kalbą, kaip naują IT VBE gali būti įsisavinę mažiau vertintojų. Todėl šią kalbą pasirinkę mokiniai gali susidurti su skirtingomis vertinimo patikimumo priežastimis:

- vertintojas programą su klaida nuvertina, nes jam dėl patirties stokos nesiseka greitai identifikuoti klaidą;
- vertintojas programą su klaida pervertina, nes jis moka ne vieną programavimo kalbą ir yra aukštesnės kvalifikacijos, todėl greitai rasta klaida atrodo nereikšminga.

Abi priežastys daugiau veikia programų su klaidomis vertinimą.

Galimi programavimo stiliaus vertinimo netolygumai

Skirtingose programavimo kalbose yra nusistovėję ganėtinai skirtingi požiūriai į teksto dėstymą, kintamųjų vardų formavimą, net komentavimą.

Jei vertinimo instrukcijose bus numatyta nemažai taškų šiems programavimo stiliaus elementams, gali iškilti diskusijų dėl bendros nuomonės, kas laikytina tinkamu programavimo stiliumi.

Mokinių programų vertinimo objektyvumas priklauso nuo užduoties parengimo kokybės[23]

Kuriant užduotį reikia atsižvelgti į tai, kad skirtingose programavimo kalbose yra skirtingos standartinės bibliotekos. Todėl tam tikrų užduočių sudėtingumas gali žymiai skirtis. Geras to pavyzdys galėtų būti žodžių, atskirtų tarpais, skaitymas iš tekstinio failo. Pascal kalboje tam nėra numatyta standartinių priemonių, todėl reikia skaityti po simbolį arba skaidyti į žodžius nuskaitytą visą eilutę. C kalbos bibliotekos fscanf() funkcija šį darbą atlieka be papildomų pastangų. Galima surasti ir daugiau pavyzdžių: standartiniai rikiavimo algoritmai su C funkcija sort() ar rikiuotomis duomenų struktūromis iš STL bibliotekos.

Į tai neatsižvelgus gali susidaryti situacija, kuri iš dalies yra Lietuvos informatikos olimpiadose, kai dauguma olimpiados nugalėtojų yra pasirinkę C/C++ kalbas. Gal šios kalbos pasirinkimas ir nėra bėda olimpiadose, tačiau tai gali kelti grėsmes egzaminui dėl nelygiavertės egzamino užduoties skirtingų mokyklų mokiniams, nes ne visose mokyklose yra mokytojų pasiruošusių dėstyti šią kalbą.

Galimi techniniai vertinimo nesklandumai

Net esant gerai paruoštai sistemai ir kokybiškoms užduotims, dirbant tik su Pascal programomis būdavo tam tikrų techninių nesklandumų. Žymi jų dalis buvo susijusi su FreePascal leidžiamais skirtingais dialektais. Įdiegus FreePascal standartiškai nustatoma FPC dialektas, kuris šiek tiek skiriasi nuo mokyklose labiausiai paplitusio istorinio Turbo Pascal 7.0 dialekto (TP7). Be to, yra ir OBJFPC, DELPHI dialektai. Bėda ta, kad dalis mokinių naudoja įvairias konstrukcijas, būdingas tik tam dialektui ir, kompiliuojant kito dialekto režimu, dažnai nepasiseka programų sukompiliuoti sėkmingai [76]. Mokinių tarpe gana populiarios konstrukcijos,

kai funkcijos vardas viduje funkcijos naudojamas kaip kintamasis. TP7 dialekte tai neleistina, o jei funkcijos vardas pasitaiko dešinėje priskyrimo pusėje — tai interpretuojama kaip rekursinis kreipinys. Tokio tipo problemos dažnai sukelia diskusijas tarp vertintojų, ar tai tikrai mokinio klaida.

Įvedus C++ kalbą iš tiesų techninių nesklandumų galime sulaukti daugiau. To priežastis – pasirinkta ne viena C++ programavimo terpė, o dvi. Jose integruoti skirtingų kartų C++ kompiliatoriai, kurie turi skirtingas galimybes, turi ir skirtingų klaidų. Todėl gali tekti aiškintis, ar programa veikia vienodai sukompiliuota skirtingais kompiliatoriais. Taip pat negalima pamiršti dar vieno aspekto – šios aplinkos tinka sėkmingai programuoti ir grynąja C kalba. Dažniausiai tai galima atpažinti pagal kitokį bylos prievardį (.c) ar įtraukiamas bylas (.h). Tačiau patirtis mokinių nėra didelė, dauguma jų bus savamoksliai C/C++ programuotojai ir gali kilti nesklandumai dėl supainiotų prievardžių ar įtrauktų bylų.

3.6.5 Pasiūlymai sklandžiai egzamino kaitai

Vertintojų kvalifikacijos problemą galima spręsti keliais skirtingais būdais. Galimas sprendimas yra trumpi mokymai, kuriuose būtų pristatyti C/C++ kalbų pagrindai. Kadangi spėtina, jog dauguma mokinių naudosis mokyklose įprasta procedūrinio programavimo paradigma, nesudėtingų programų vertinimas gali būti ne toks sunkus, jei programos bus trumpos, paprastos, naudos nedaug konstrukcijų. Išlieka pavojus, kad dalis mokinių naudosis objektiniu programavimu, kuris nėra gerai įsisavintas daugumos mokytojų. Tačiau spėtina, kad dauguma šią programavimo paradigmą įsisavinusių mokinių turi gerus programavimo įgūdžius ir jiems nereikės alternatyviojo vertinimo.

Kitas galimas kelias – skaidyti vertintojų grupę į mokančius C/C++ ir nemokančius. Tokiu atveju pateikti C/C++ sprendimai pakliūtų tik pas

gerai mokančius šias kalbas. Tačiau šiuo atveju išlieka aukščiau minėtos grėsmės dėl skirtingos vertintojų kvalifikacijos.

Norint parengti užduotį, kuri būtų panašaus sunkumo naudojantiems skirtingas programavimo kalbas, verta atkreipti dėmesį į:

- sąlygas formuluoti labai tiksliai, nurodant kokias konstrukcijas, funkcijas galima naudoti, kurių ne;
- sukurti programavimo kalboms bendrą aprašą, paskelbiamą prieš egzaminą, kuriame būtų fiksuotas programavimo kalbos poaibis, kuriuo bus galima naudotis egzamine. Šis aprašas palengvintų ir mokinių, ir vertintojų darbą;
- kurti sąlygas labai atidžiai analizuojant visus galimus sprendimus ir naudoti tik tas užduotis, kurios galimai neturi esmingų skirtumų realizacijos prasme.

3.6.6 Situacijos įvertinimas po 2011 ir 2012 metų egzaminų

2011 m. įvyko pirmasis naujo tipo egzaminas. C++ kalba pateikta 59 praktinių užduočių sprendimų: 32 pirmosios praktinės užduoties ir 27 – antrosios. Šiuos sprendimus pateikė 33 kandidatai iš 1270 pateikusiųjų bent vieną praktinės užduoties sprendimą. Taigi, C++ kalbą pasirinko dar labai nežymi abiturientų dalis. Todėl vertinimo problemų tai beveik nekėlė – buvo pasirinkta strategija, kad šiuos darbus vertins tik C++ kalbą gerai mokantys ir savimi pasitikintys vertintojai. Tokių vertintojų buvo 5 iš 28. Pabrėžtina, kad trečiam vertinimui C++ buvo iškelta tik 6,8% (lyginant su 15,0% Paskalio darbams). Manytina, kad tai patvirtino prielaidą, jog C++ darbus vertino aukštesnės kvalifikacijos vertintojai. Tačiau gali būti, kad šiais metais C++ rinkosi stipresni abiturientai – jie už praktines užduotis surinko vidutiniškai po 21,1 taško (palyginimui

Pascal pasirinkę – 11,2 taško). Manytina, kad prielaidos apie galimą vertinimo netolygumą nei patvirtintos, nei paneigtos, nes C++ 2011 metais rinkosi per mažai abiturientų, kad turėtume galutinę išvadą.

2012 metais egzamino kaita jau buvo įvykusi ir įsitvirtinusi. Egzamine parašytų C++ kalba programų skaičius pasiekė 298 ir sudarė 12,5%. Vertintojų, kurie sutiko vertinti C++ programas taip pat žymiai padaugėjo - jų buvo 11 iš 30. Tačiau dėl C++ besirenkančių stipresnių mokinių tendencijos išliko: C++ pasirinkusių vidutinis taškų skaičius už užduotį buvo 15,8 taško, o Pascal – 9,9 taško.

3.6.7 IT VBE kaitos tyrimo ir vertinimo sistemos modifikavimo dėl egzamino kaitos išvados

Egzamino pakeitimas taip pat sukurė naujų iššūkių. Didžiausi uždaviniai iškelti užduočių kūrėjams, vertinimo sistemos kūrėjams bei vertintojams.

Pusiau automatinės sistemos adaptavimas pasikeitusiai situacijai sukelia problemas dėl neturimų galimybių naudoti adekvačius statinės analizės įskiepius. Automatinio vertinimo dalies modifikavimas privertė peržiūrėti žymią sistemos dalį. Technines problemas kėlė kompiliatorių realizacijų skirtumai bei dialektų skirtumai.

3.7 Metodo, skirto pusiau automatinėms vertinimo sistemoms tobulinti, pagrindiniai etapai

Pusiau automatinės programavimo užduočių vertinimo sistemos tobulinimui reikalinga atlikti šiuos žingsnius:

- Įdiegti / pakeisti programos teksto peržiūros komponentą teksto redagavimo komponentu.
- Pertvarkyti sistemos duomenų srautus taip, kad kiekvienai pateiktai programai būtų galima išsaugoti bent po vieną modifikuotos programos ir jos testavimo rezultatų kopiją.

- Sukurti dar vieną vertintojui galimą sistemos panaudos būdą – vertintojui turi būti sudarytos sąlygos pertestuoti pakeistą programą ir po to turi būti interaktyviai pateikiami modifikuotos programos testavimo rezultatai.
- Sistemoje turi būti atskirtas testo teisingumas nuo atsakymo formato teisingumo. Tam reikia pertvarkyti tiek atsakymo teisingumo tikrinimo komponentą,
- Sistemoje reikia numatyti gautos papildomos testavimo bei programos taisymų informacijos perdavimą vertintojui (kai kuriais atvejais ir mokiniui).
- Kiekvienai programavimo kalbai, kuria parašytas programos bus vertinamos, reikia sukurti atskirą rinkinį statinės analizės įskiepių.
- Reikia numatyti galimybę naudoti skirtingus vertinimo kriterijus programoms parašytoms skirtingomis kalbomis.

3.8 Išvados

- Egzamino programavimo užduočių vertinimo problema formalizuota, apibrėžti funkciniai ir nefunkciniai reikalavimai.
- Patobulintas klasikinis pusiau automatinio vertinimo ir testavimo metodas.
- Eksperimentine realizacija pagrįsta, kad šis metodas galėtų spręsti formalizuotą egzamino programavimo užduočių vertinimo problemą.
- Išanalizuota, kas pusiau automatinio vertinimo ir testavimo metode keičiasi, kintant egzamino reikalavimams.

4 Eksperimentinė dalis

Eksperimentiniam metodo tyrimui naudota IT VBE praktinių užduočių vertinimo sistema (3.8 pav.). Ji egzamino ir bandomiesiems vertinimams naudojama 6 metus. Kasmet egzamino vertinimo metu ją taikant įvertinama apie 2400 abiturientų parašytų programų. Visos programos įvertinamos bent po du kartus. Iškilus žymesniam (3 balų) skirtumui tarp skirtingų vertintojų, darbą pervertina trečiasis vertintojas. Galutiniu įvertinimu laikomas vėliausiai parašytas įvertinimas.

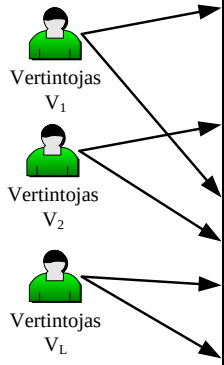
Vertintojų komandoje dirba apie 30 vertintojų. Vertinimo sesija trunka vieną savaitę (penkias darbo dienas). Vertintojais dirba vidurinių mokyklų ir gimnazijų IT mokytojai bei universitetų dėstytojai. Vertintojų komanda yra gana stabili – jau keletą metų kasmet pasikeičia ne daugiau trijų žmonių.

Vertinimo metu vertintojai gauna mokinių parašytų programų paketus (4.1 pav.), kuriuose būna po 10 darbų. Antrą kartą vertinant paketas negali būti duotas tam pačiam vertintojui.

4.1 Įdiegto pusiau automatinio vertinimo ir testavimo metodo eksperimentinis tyrimas

Eksperimentinis tyrimas rėmėsi ginamųjų teiginių tyrimu:

1. Naujas pusiau automatinio programų testavimo metodas leidžia efektyviau (sparčiau) įvertinti IT VBE praktinės užduoties metu abiturientų parašytas programas nei tai būtų galima atlikti rankinio vertinimo metodu. Šiai hipotezei eksperimentiškai ištirti buvo pasirinktas statistinis kiekybinis tyrimas – palyginta, kiek laiko užtrunka 50 mokinių programų testavimas pusiau automatiniu būdu ir



Paketas	Programa	Alternatyva		Rankinio vertinimo rezultatai	Vertinimo rezultatas
		Automatinio testavimo rezultatai	Pusiau automatinio testavimo rezultatai		
PK_1	P_1	AR_1	PAR_1	RR_1	VR_1
	...	...	...	...	...
	P_N	AR_N	PAR_N	RR_N	VR_N
PK_2	P_{N+1}	AR_{N+1}	PAR_{N+1}	RR_{N+1}	VR_{N+1}
	...	...	...	...	...
	P_{2N}	AR_{2N}	PAR_{2N}	RR_{2N}	VR_{2N}
...	...	...	...	...	...
PK_M	$P_{N(M-1)+1}$	$AR_{N(M-1)+1}$	$PAR_{N(M-1)+1}$	$RR_{N(M-1)+1}$	$VR_{N(M-1)+1}$
	...	...	...	...	...
	P_K	AR_K	PAR_K	RR_K	VR_K

4.1 pav. **Vertintojų santykis su programų paketais. Užtušuoti langeliai pildomi vertintojo interaktyvaus darbo metu**

rankiniu būdu. Taip pat vertintojai apklausti dėl laiko parinkimo bandomajam testavimui.

2. Nauju pusiau automatinio programų testavimo metodu paremtos IT VBE praktinių užduočių vertinimo sistemos formuojami įvertinimai objektyvumu nenusileidžia nei įprastiniam rankiniam vertinimui, nei automatiniam testavimui. Ginamasis teiginys buvo tiriamas pasinaudojant kokybinio tyrimo lyginamosios analizės ir apklausų metodais.

4.2 Pirmojo ginamojo teiginio eksperimentinis tyrimas

Naujas pusiau automatinio programų testavimo metodas leidžia efektyviau (sparčiau) įvertinti IT VBE praktinės užduoties metu abiturientų parašytas programas nei tai būtų galima atlikti rankinio vertinimo metodu.

Eksperimentiškai buvo tiriama, kiek vidutiniškai laiko užtrunka vertintojas tikrindamas 10 mokinių parašytų programų paketą besinaudodamas IT VBE praktinių užduočių vertinimo sistema ir kiek laiko užtrunka jis naudodamasis tik įprasta programavimo aplinka.

Nors turima duomenų iš įvairių metų egzaminų vertinimų iškilo abejonių dėl:

- vertintojų eksperimentui parinkimo;
- mokinių darbų paketų parinkimo.

4.2.1 Vertintojų ir jų darbo laiko parinkimas

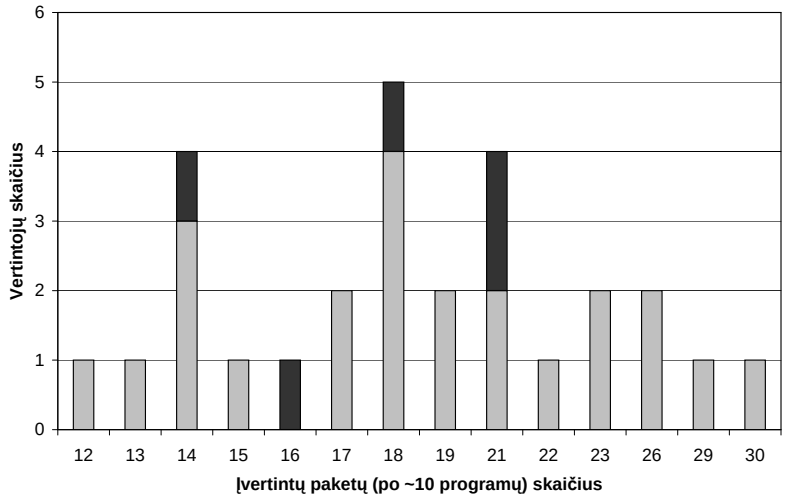
Vertintojų ir jų darbo laiko parinkimas nebuvo paprastas, nes:

- vertintojų darbo efektyvumas yra labai skirtingas. Per savaitės trukmės IT VBE tikrinimo sesiją tarp vertintojų lyderių ir lėčiau dirbančių vertintojų patikrintų programų skaičius skiriasi beveik dvigubai;
- vertintojų darbo efektyvumas beveik pastoviai auga vertinimo sesijos bėgyje. Tai sietina su apsimokymu naudotis testavmo sistema, prisitaikymu vertinti konkrečią užduotį sprendžiančias programas;
- egzamino vertinimo metu nėra galimybių daryti rankinio darbo eksperimentų – dirbama intensyviai, vertinimui skirtas laikas yra griežtai ribojamas, nes vertintojų grupė turi spėti įvertinti visus darbus per tam skirtą savaitę.

Vertintojų ir jų darbo laiko parinkimo būdas:

- parinkti 5 vidutinės darbo spartos vertintojai. Jų per savaitę įvertintų programų skaičius tarpusavyje skyrėsi ne daugiau kaip 70 (33%). Parinktų vertintojų darbo sparta iliustruota histograma 4.2. Vidutiniškai vertintojai įvertino po 190 programų;

- vertinimo spartai IT VBE praktinių užduočių vertinimo sistemai buvo parinkti 5 programų paketai, kurie buvo vertinami antros vertinimo dienos ryte. Tai pagrįsta tuo, kad pirmąją dieną žymi dienos dalis panaudojama vertinimo instrukcijos tobulinimui, bandomajam testavimui. Ritmingas testavimo darbas prasideda antrąją dieną;
- pasirinktų vertintojų prašyta atlikti bandomąjį rankinį testavimą, kai nuo vertinimo sesijos jau buvo praėjęs mėnuo. Tai garantavo, kad per savaitę įgyti įgūdžiai sparčiai vertinti nurodytos užduoties programas jau buvo kažkiek prarasti. Apklausus vertintojus apie tai, kiek jie prisiminė užduotis ir jų vertinimo instrukcijas jie konstatavo: „vėl reikėjo pradėti nuo pradžių“, „kažką atsimenu, bet į popierinę lentelę reikėjo pažiūrėti“, „ kriterijus daugmaž atsimenu, bet populiariausias mokinių klaidas primiršau“. Antra vertus, vertintojai jau nebebuvo tiek susirūpinę darbo terminais: „visai gerai prisiminti tuos uždavinius, nes ruduo jau ne už kalnų“.



4.2 pav. **Vertintojų darbo spartos histograma 2012 metais IT VBE vertinimo sesijoje. Tamsesne spalva pažymėti parinkti vertintojai.**

4.2.2 Mokinių darbų paketų parinkimas

Kaip jau minėta, mokinių darbų paketai buvo parenkami pagal tai, kuriuo metu jie pirmąjį kartą pakliuvo vertintojui. Buvo atrinkti antrą

dieną ryte vertinti darbai. Antra vertus, buvo atsižvelgiama į pakete esančių darbų įvertinimus. Mat statistiškai daugiausiai laiko sugaištama vertinant „vidutinius“ darbus. Tai galima pagrįsti tuo, kad sėkmingai testus įveikiančios programos, dažnai sparčiai įvertinamos artimais maksimaliems balais. Programos, panašios į pradinę programavimo aplinkos sugeneruojamą „Sveikas pasauli“ programą, taip pat vertinamos labai sparčiai. Atrinkti mokinių darbų paketai turėjo po 5–6 vidutiniškai įvertintas programas.

Dar vienas paketo parinkimo kriterijus buvo, kad kievienas iš parinktų vertintojų būtų taisęs po 1 iš nurodytų paketų. Tyrime specialiai buvo duodami jau vertinti tų pačių vertintojų paketai, nes labai sunku surasti analogiškas programas su to paties sudėtingumo vertinimu. Todėl efektyvumo palyginimas vertinant netapačius programų paketus būtų buvęs negalimas.

Mokytojai apklausoje konstatavo, kad konkrečiai tų darbų jie neprisiminė: „ką ten prisiminsi, jei patikrini šimtus panašių programų“, „viena programa pasirodė gana matyta, bet neprisiminiau, kaip taisiau ir ką dariau :(“.

4.2.3 Apibendrinti tyrimo rezultatai

Palyginus visų 5 vertintojų vertinimo laiko skirtumus (4.1 lentelė), matome žymiai didesnes laiko sąnaudas (43-117%) vertinant rankiniu būdu. Šis palyginimas darytas programų paketo atžvilgiu. Vertintojai klausimyne pažymėjo, kad „užtrukau ilgiau nei galvojau“, „manau, galėjau įvertinti greičiau, tik vasarą dirbasi lėčiau“, „niekur neskubėjau“. Šios pastabos kelia abejonių dėl laiko parinkimo bandomajam vertinimui. Tačiau sugaištas vertinimui laikas skiriasi tiek žymiai, kad abejoti aukštesniu pusiau automatinio testavimo sistemos efektyvumu netenka.

Atkreiptinas dėmesys į klausimo „Kam sugaišote daugiausiai laiko?“ atsakymus: „užtrunka laiko junginėtis tarp įvairių OS langų“, „nepatogu

kaitalioti failų vardus programavimo aplinkoje. Kai sugalvojau testų failus pervardinti – darbas pagreitėjo“, „susipainiojau su savo pataisymais. Teko ieškoti pradinio failo archyve“. Atrodo, kad vertintojai jau yra pripratę prie vertinimo sistemos kai kurių paslaugų ir grįžti prie rankinio vertinimo nebenori.

Vertintojas	Įvertino darbų per sesiją	Pusiau automatinis testavimas, val.	Rankinis vertinimas, val.	Laiko santykis, rank.v./p.aut.
A	140	01:38	02:20	1,43
B	210	01:15	01:50	1,47
C	180	01:20	02:35	1,94
D	210	00:50	02:00	2,40
E	160	00:53	01:55	2,17
Iš viso		05:56	10:40	1,80

4.1 lentelė. Verintojų darbo laiko sąnaudų palyginimas vertinant pusiau automatiniu ir rankiniu būdu dešimties programų paketą

4.3 Antrojo ginamojo teiginio eksperimentinis tyrimas

Nauju pusiau automatinio programų testavimo metodu paremtos IT VBE praktinių užduočių vertinimo sistemos formuojami įvertinimai kokybe nenusileidžia rankiniam vertinimui ir yra kokybiškesni už automatinį testavimą.

Šios teiginio tyrimui buvo parinkti keli skirtingi metodai, nes reikalingi palyginimai su dviem labai skirtingais vertinimo būdais.

Disertacijos autorius mano, kad pagal kokybę aukštos kvalifikacijos eksperto atliekamas rankinis vertinimas gali būti laikomas etaloniniu. Taip yra dėl jo prigimties – dar daug programų vertinimo kriterijų negali būti patikrinti automatiniu būdu. Be to, šiuolaikinės programavimo kalbos tobulinamos, kad būtų patogios žmogui. Todėl tik žmogus gali

iki galo išsiaiškinti kito žmogaus rašytos programos subtilumus. Laikydamasis šios pozicijos kokybiškiausiu programos vertinimo metodu laikau rankinį vertinimą.

Tyrimo metu buvo atlikta lyginamoji analizė tiriant 50 programų galutinius įvertinimus. Vieni jų buvo gauti jau anksčiau minėto tyrimo metu vertinant rankiniu būdu. Kiti rezultatai buvo gauti dėka IT VBE praktinių užduočių vertinimo sistemos egzamino sesijos metu. Taip pat buvo analizuoti apklausos atsakymai. IT VBE vertintojai tvirtino, kad IT VBE sistemos naudojimas jiems padeda „greičiau atlikti vertinimą“. Jie šio naujo vertinimo metodo neišskyrė į atskirą klasę ir sistemą vertino kaip pagalbinį įrankį rankiniam vertinimui. Tai reiškia, kad sąsaja ir teikiamos paslaugos neturi įtakoti vertinimo kokybės.

IT VBE metu trečiam vertinimui dažniausiai iškeliami darbai, kurių galutiniai įvertinimai skiriasi daugiau nei 2 balais. Lyginant rankinio ir pusiau automatinio vertinimo rezultatus nustatyta, kad didesnį balų skirtumą nei 2 balai tarp rankinio ir pusiau automatinio vertinimų turėjo 12% programų (6 atvejai). Tai yra žemesnis rodiklis nei vykstant IT VBE vertinimo sesijai, kurioje 2012 metais trečiam vertinimui buvo iškelta 22,8% programų. Darytina išvada, kad pusiau automatinio testavimo metu vertintojų parašyti įvertinimai skiriasi nežymiai nuo parašytų rankinio vertinimo būdu. Todėl teiginį, kad šio pusiau automatinio vertinimo ir testavimo metodu paremtos sistemos vertinimo kokybė nenusileidžia rankinio vertinimo kokybei laikau įrodytu.

4.3.1 IT VBE vertinimo sistemos vertinimo kokybės palyginimas su automatiniu vertinimu

Buvo analizuojamos 2010 m. IT VBE programos, kurios neįveikė testų, tačiau bent vienas iš vertintojų skyrė visus taškus už alternatyviuosius kriterijus. 4.2 lentelėje pateikiami duomenys apie egzamino apimtis ir šių tiriamų programų skaičių.

	Iš viso pateikta	Programų su tiriamu vertinimu
Pirmosios užduoties	1224	34
Antrosios užduoties	1019	62
Abiejų užduočių	2243	96

4.2 lentelė. 2010 m. IT VBE pateiktų ir tirtų programų skaičiaus palyginimas

Minimos programos buvo pasirinktos, nes jos turėjo būti pakankamai artimos veikiančioms (kitaip vertintojai nebūtų davę daug alternatyvių balų). Kadangi jos sudaro arti 5% visų pateiktų programų, tai patvirtina prielaidą, kad vien automatinio vertinimo neužtenka IT VBE vertinimui.

Šis tyrimas išryškino tam tikras vertintojų pozicijas, nes jei vienas vertintojas davė visus balus už alternatyvų vertinimą, tai kitas galėjo visų balų neduoti. Buvo pasigilinta į skirtingai įvertintus kriterijus. Dažniausiai galima buvo identifikuoti arba ne iki galo tikslų kriterijų taikymą, arba ne iki galo aiškią kriterijaus formuluotę taikant ją konkrečiam sprendimui.

Tokių kriterijų pavyzdys galėtų būti kriterijus „Nenaudojami globalieji kintamieji“. Skirtingi vertintojai, vertindami programą su globaliu masyvu, tai interpretavo skirtingai. Šios problemos iliustracijai tiktų ir kriterijus „Sumos kintamojo inicializavimas“. Kai kurie vertintojai laikė, kad globalius kintamuosius neprivaloma inicializuoti nuliu, nes Free Pascal juos inicializuoja automatiškai.

Buvo analizuojamos tipingos klaidos bei priežastys, dėl kurių automatinis vertinimas nepateikė balų už testus.

Viena būdingiausių situacijų buvo skirtingo Pascal kalbos dialekto naudojimas. Vertinimo sistemoje programos testuojamos naudojant Turbo Pascal 7.0 dialektą, kuris yra standartas de facto mokymui ir įvairioms varžyboms. Free Pascal turi įvairių kalbos išplėtimų, kuriuose leidžiama funkcijos vardą naudoti kaip kintamąjį, deklaruojant funkcijas be argumentų galima naudoti tuščius skliaustelius ir pan. Vertintojai pertestuo-

davo minimaliai pataisę programas ir gaudavo testų teisingus atsakymus. Šioje situacijoje skirtingų vertintojų programų pataisymai galėjo skirtis – tai priklausė nuo vertintojų patirties. Programų įvertinimai dėl to buvo gauti taip pat skirtingi.

Kita būdinga programų klasė – tai programos su įvairiomis skubėjimo, neatidumo klaidomis: netikslūs failų vardai, dokumentuojant programą netyčia užkomentuoti svarbūs fragmentai, neinicializuoti kintamieji. Galima spėti, kad šių klaidų gali būti sunku išvengti dėl egzamine patiriamo kandidatų streso ir riboto laiko.

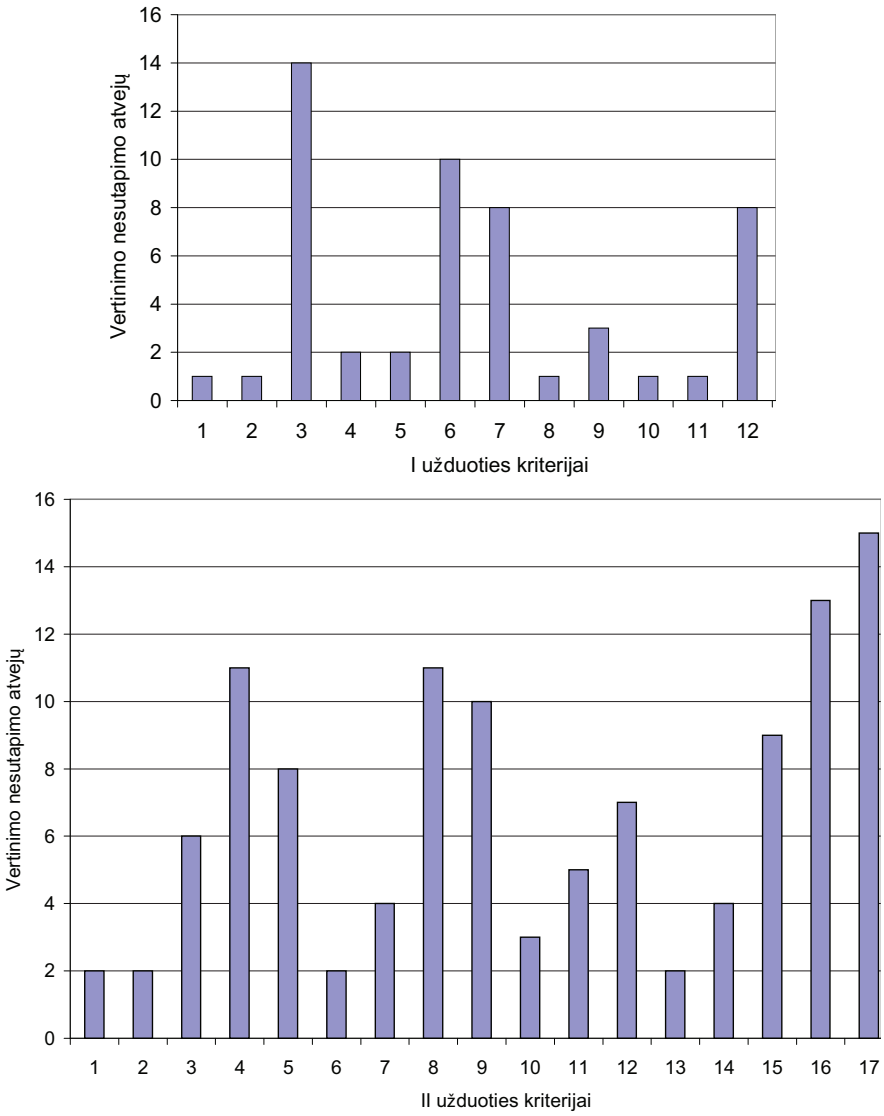
4.3 paveiksle pateikiamos diagramos demonstruojančios, kad skirtingų vertintojų vertinimai įvairiuose kriterijuose skyrėsi netolygiai. Be jau minėtų bendrų problemų labiau skyrėsi vertintojų vertinimai apie funkcijų grąžinamos reikšmės inicializavimą, skaičiavimą, rezultato grąžinimą. Tai greičiausiai rodo, kad šių kriterijų vertinimas vertintojams sukelia problemų analizuojant pateiktas programas.

Histogramose taip pat išsiskiria ir grynai techniniai kriterijai, pvz. teisingas failo skaitymas. Galbūt vertintojams jie kėlė abejones, nes programos pilnai nefunkcionavo, o klaidos vertė nebuvo lengvai įvertinama.

Taip pat teigtina, kad trečiasis vertinimas mažiau skyrėsi nuo antrojo vertinimo, nei antrasis nuo pirmojo. Tai galima paaiškinti kylančia vertintojų kvalifikacija duotai užduočiai.

4.3.2 IT VBE vertinimo sistemos vertinimo kokybės palyginimas su klasikiniu pusiau-automatiniu vertinimu

Buvo analizuojamos 2010 m. IT VBE rezultatai. Kadangi naujo pusiau automatinio vertinimo metodo balus sudaro trys dalys (automatinio vertinimo rezultatai, rankinio vertinimo rezultatai ir alternatyviojo pusiau automatinio vertinimo rezultatai), buvo nesunku įvertinti, kaip

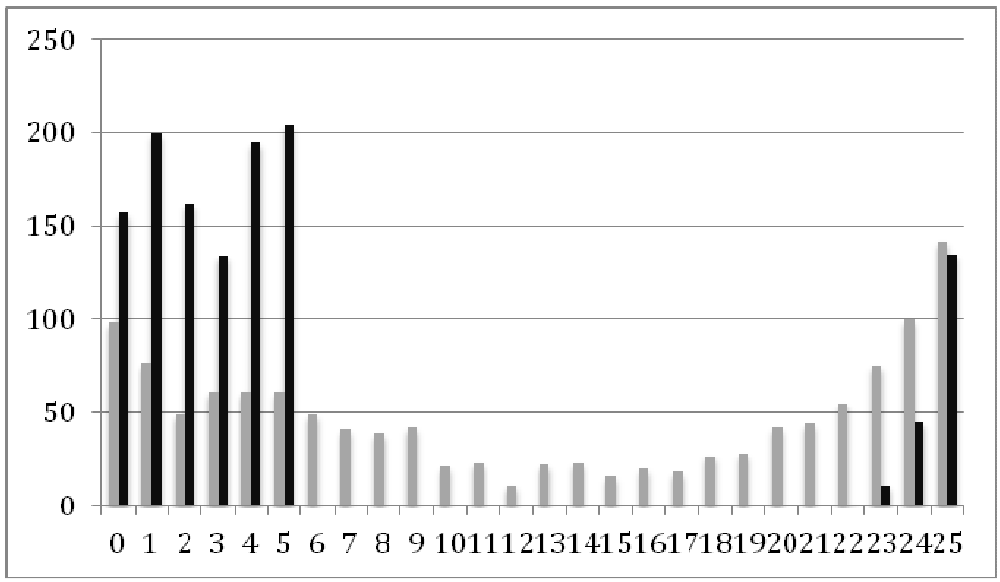


4.3 pav. Alternatyvių vertinimo kriterijų nesutampančių vertinimų histogramos (tiriamų programų aibėje) [76]

būtų pasiskirstę balai, jei balus sudarytų tik automatinio ir rankinio vertinimo balai. Pateikiama histograma 4.4 iliustruoja problemą, kad nemažai atvejų įprastinis pusiau automatinis vertinimas veikia ne visoje balų skalėje. Tuo tarpu pusiau automatinis vertinimas leidžia išnaudoti visą balų skalę ir tuo būdu leidžia pasiekti kokybiškesnį įvertinimą.

4.4 Išvados

- Sukurta IT VBE praktinių užduočių vertinimo sistema realizuoja naują pusiau automatinio testavimo ir vertinimo metodą.



4.4 pav. **2010 m. ITVBE vienos užduoties įvertinimų histograma. Taškai (0-25) yra atidėti horizontaliai, vertikaliai pažymėti abiturientų kiekis, gavusių nurodytą balų skaičių. Juoda histograma rodo įprastinio pusiau automatinio vertinimo rezultatus, pilka rodo balų pasiskirstymą taikant naują metodą.**

- IT VBE praktinių užduočių vertinimo sistema naudojama egzamino vertinimui jau keletą metų. Per šį laikotarpį, naudojant paminėtą sistemą, įvertinta apie 10000 programų. Ginamiesiems teiginiams tirti naudoti šių egzaminų metu rašytos programos.

- Ištirtas vertinimo spartos pagerėjimas taikant naują pusiau automatinio testavimo ir vertinimo metodą. Jis rodo, kad vertinimo sparta yra 1,4–2,2 karto didesnė nei taikant rankinio vertinimo metodą.

- Ištirtas vertinimo kokybė taikant naują pusiau automatinio testavimo ir vertinimo metodą. Jo kokybė nenusileidžia rankinio vertinimo metodui, bet pasižymi aukštesne kokybe už automatinį ir įprastą pusiau automatinį vertinimą (ypač nepilnai funkcionuojančioms programoms).

5 Sukurto pusiau automatinio vertinimo ir testavimo metodo galimi taikymai

Kadangi patobulintas pusiau automatinio vertinimo ir testavimo metodas yra tik patobulinimų rinkinys, lyginant su įprastinėmis pusiau automatinėmis programų testavimo sistemomis, galima tvirtinti, kad jis nėra sunkiai įdiegiamas į žymią dalį automatinių ir pusiau automatinių sistemų, kuriose yra numatyta mokytojo-vertintojo rolė.

Siekiant tuo įsitikinti, buvo atlikti eksperimentai su *Edujudge* projekte kurtu nuotolinės mokymo aplinkos *Moodle* įskiepiu. Bandyta įdiegti siūlomus patobulinimus. Įsitikinta, kad tai nėra sudėtinga užduotis. Daugiausia problemų kelia tai, kad iš dalies keičiasi duomenų panaudojimas iš mokytojo pusės. Mokytojui nebuvo numatyta galimybė redaguoti mokinio sprendimus, taip pat nebuvo numatyta galimybė kreiptis į automatinio programų testavimo modulį. Todėl eksperimentiniame *Edujudge* modulio tobulinimo sprendime iš dalies persipynė mokytojo ir mokinio rolės.

Išskaidytos atsakymo vertinimo programos įdiegimas problemų nesukėlė. Taip buvo iš dalies todėl, kad tai yra labai izoliuotas modulis, kuris neturi tiesioginių ryšių su visa sistema, tiesiogiai nekeičia duomenų bazės ir dirba tik pagal centrinės testavimo sistemos dalies poreikį. Pagrindinė problema liko tik užtikrinti detalizuoto grįžtamojo pranešimo pristatymą mokiniui bei mokytojui.

Šio pusiau automatinio vertinimo metodo taikymas suteikia naujų galimybių programavimo mokymui. Mokymosi efektyvumui labai svarbi besimokančiajam grąžinamos informacijos kokybė. Dažnai, vertinant studentų ar mokinių programas, pasitenkinama tik įveiktų testų suvestine, suteiktais balais ar fraze iš fiksuoto frazyno. Tuo tarpu, taikant šį metodą,

galima būtų besimokančiajam grąžinti funkcionuojančią programą su ištaisytomis klaidomis. Taip pat svarbi programos taisymų eiga, kol programa pradeda funkcionuoti. Mokytojo vykdomas programos taisymas bandymų ir klaidų metodu padėtų mokiniams išmokti klaidų paieškos metodų, parodytų mokytojo mąstymo eigą.

6 Bendrosios išvados ir rezultatai

- Ištirtos ir susistemintos šiuolaikinių automatinių ir pusiau automatinių programų testavimo sistemų galimybės. Įsitikinta, kad šioje programinės įrangos klasėje vis dar tebėra daug problemų, vyksta sistemų tobulinimo procesas. Nustatyta, kad mažiausiai ištirta pusiau automatinio programų testavimo kryptis. Iki šiol nebuvo ištirta galima automatinio programų testavimo rezultatų sąveika su vertintoju ir jo rankinio vertinimo būdu parašytais įverčiais.

- Sukurtas naujas pusiau automatinio vertinimo ir testavimo metodas, kuriuo remdamasis vertintojas gali interaktyviai naudotis automatinio programų testavimo sistema, atlikti eksperimentus su mokinio programa ir analizuoti programoje esančių klaidų sukeliamas pasekmes, įvertinti jų svorius.

- Sukurta Lietuvos informacinių technologijų valstybinio brandos egzamino praktinių užduočių vertinimo sistema, kurioje panaudotas naujas pusiau automatinio vertinimo ir testavimo metodas. Sistema sėkmingai naudojama Nacionaliniame egzaminų centre, ji nuolat tobulinama, adaptuojama pagal kintančio egzamino poreikius.

- Atlikti sukurto pusiau automatinio vertinimo ir testavimo metodo bandymai lyginant vertintojų darbo efektyvumą su rankiniu vertinimu. Nustatytas žymus efektyvumo pagerėjimas (1,4 – 2,2 karto). Taip pat įsitikinta, kad sukurto metodo pusiau automatinio vertinimo objektyvumas nenusileidžia rankiniam vertinimui.

- Atlikta kokybinė analizė darbų, kurių vertinimas, taikant naują pusiau automatinio vertinimo ir testavimo metodą, iš esmės skyrėsi

nuo automatinio testavimo rezultatų. Nustatyta, kad naujo metodo taikymas leido kokybiškiau įvertinti šiuos darbus.

- Pateikto pusiau automatinio vertinimo ir testavimo metodo įdiegimas į kitas pusiau automatines sistemas yra nesudėtingas.

Priedas A: Eksperimentinio vertinimo vertintojų apklausos klausimynas

Sveiki.

Atliekama Informacinių technologijų valstybinio brandos egzamino praktinių užduočių vertinimo sistemos analizė. Jūs jau keletą metų dirbate IT VBE vertintoju, todėl kaip eksperto prašau pagalbos.

Įvertinkite pateiktas programas. Vertinimui naudokite 2012 m. IT VBE vertinimo instrukciją, testus ir įprastines programavimo aplinkas (FPS, Dev C++, Codeblocks). Darbų įvertinimus pateikite prisegtoje MS Excel lentelėje (kiekvienam darbui skirtas atskiras lakštas).

Svarbu pažymėti darbo pradžios ir pabaigos laiką. Įvertinę darbus atsakykite į žemiau pateiktus klausimus.

---

1. Kiek sugaišote laiko šių programų įvertinimui?
2. Ar tikėjotės atlikti šį darbą sparčiau?
3. Kam sugaišote daugiausiai laiko? Ar egzamino vertinimo sistema būtų padėjusi įvertinti sparčiau?
4. Ar greitai prisiminėte užduotį ir jos vertinimo instrukciją? Gal reikėjo viską pakartotinai išanalizuoti?
5. Ar tarp darbų buvo „matytų“ programų? Gal atsiminėte kaip ją taisėte per IT VBE vertinimo sesiją?
6. Ar egzamino sistema būtų padėjusi vertinti greičiau?
7. Gal turite pastabų apie pateiktas programas ar šį tyrimą?

Ačiū už pagalbą.

Priedas B: IT VBE praktinės užduoties ir vertinimo schemos pavyzdys

Paveiksluose B.1, B.2, B.3, B.4 pateikiamos Lietuvos informacinių technologijų brandos egzamino 2010 metų praktinių užduočių salygos.

Paveiksluose B.5, B.6 pateikiamos šių užduočių vertinimo instrukcijos. Jose galima atkreipti dėmesį į alternatyviojo vertinimo galimybę, kuri išnaudojama, jei pateiktos programos neįveikia testų.

Šios sąlygos ir vertinimo instrukcijos yra Nacionalinio egzaminų centro (`http://www.egzaminai.lt`) intelektualinė nuosavybė. Čia pateikiamos užduoties ištraukos yra paskelbtos Nacionalinio egzaminų centro svetainėje ir buvo 2012-09-01 pasiekiamos internetu adresais:

- `http://www.egzaminai.lt/failai/1723_IT_-1-vert-2010.pdf`;
- `http://www.egzaminai.lt/failai/1602_IT-pagr-2010.pdf`.

II. PRAKTINĖS UŽDUOTYS

Trukmė – 90 min.

1. Šachmatų turnyras

Maksimalus vertinimas – 25 taškai

Mokykloje organizuojamas šachmatų turnyras, tačiau trūksta šachmatų žaidimo komplektų. Paaiškėjo, kad dalis mokinių turi namuose šachmatų žaidimo komplektus, kuriuose trūksta kai kurių baltų figūrų (juodų figūrų netrūksta). Jie turimus komplektus atnešė į mokyklą.

Parašykite programą, kuri suskaičiuotų, kiek pilnų šachmatų žaidimo komplektų galima sudaryti iš mokinių atneštų figūrų.

Vienos spalvos figūrų komplektą sudaro 8 pėstininkai, 2 bokštai, 2 žirgai, 2 rikiai, 1 karalius ir 1 valdovė.

Duomenys

Tekstiniame faile `U1.txt` yra kelios eilutės su sveikaisiais skaičiais.

- Pirmoje eilutėje užrašytas mokinių skaičius N ($1 \le N \le 100$).
- Toliau yra N eilučių, kuriose surašyti mokinių atneštų baltų figūrų skaičiai. Kiekvieno mokinio figūrų sąrašui skiriama viena eilutė. Kokių ir kiek mokinys atnešė baltų figūrų, surašyta tokia tvarka: pėstininkai, bokštai, žirgai, rikiai, karaliai ir valdovės. Jeigu kurios nors figūros mokinys neatnešė, toje vietoje parašytas nulis. Duomenų failo pavyzdyje parašyta, kad pirmas mokinys atnešė 22 pėstininkus, 3 bokštus, 5 žirgus, 6 rikius ir 2 karalius, o valdovių neatnešė.

Rezultatas

Tekstiniame faile `U1rez.txt` pateikite, kiek šachmatų žaidimo komplektų galima sudaryti iš mokinių atneštų figūrų.

Duomenų failo pavyzdys	Paaiškinimas	Rezultatų failo Pavyzdys	Paaiškinimas
4	Mokinių skaičius	4	Iš mokinių atneštų baltų figūrų galima sudaryti 4 šachmatų komplektus
22 3 5 6 2 0	Pirmo mokinio atneštos baltos figūros		
1 1 1 1 1 1	Antro mokinio atneštos baltos figūros		
8 4 4 4 1 2	Trečio mokinio atneštos baltos figūros		
5 3 3 3 0 2	Ketvirto mokinio atneštos baltos figūros		

Nurodymai

- Programoje **būtinai** naudokite vienmačius sveikųjų skaičių masyvus.
- Parašykite funkciją, kuri skaičiuotų, kiek šachmatų komplektų galima sudaryti iš mokinių atneštų baltų figūrų.
- Programoje nenaudokite sakinių, skirtų darbui su ekranu.

B.1 pav.: 2010 m. IT VBE I praktinės užduoties sąlyga

Programos vertinimas

Vertinimo kriterijai	Taškai	Pastabos
Testai	20	Visi taškai skiriami, jeigu programa pateikia teisingus visų testų rezultatus
Teisingai skaitomi duomenys iš failo	4	Vertinama tada, kai neskiriama taškų už testus
Teisingai spausdinamas rezultatas	2	
Sukurta funkcija, kuri suskaičiuoja, kiek šachmatų žaidimo komplektų galima sudaryti iš mokinių atneštų baltų figūrų	5	
Teisingos kitos procedūros ir funkcijos, jeigu jų yra, ir pagrindinė programa	9	
Teisingai aprašyti vienmačio masyvo duomenų tipas (tipai) ir kintamieji	1	Visada vertinama
Sukurta nurodytus skaičiavimus atliekanti funkcija	1	
Prasmingai pavadinti kintamieji. Komentuojamos programos dalys, laikomasi rašybos taisyklių	1	
Išlaikomas vientisas programos rašymo stilius, nėra sakinių, skirtų darbui su ekranu	2	
Iš viso taškų	**25**	

JUODRAŠTIS

B.2 pav.: 2010 m. IT VBE I praktinės užduoties sąlyga

2. Gimtadienis

Maksimalus vertinimas – 25 taškai

Martynas savo gimtadienio proga užsakė pietus visai klasei. Pietus sudaro vienodas patiekalų komplektas kiekvienam svečiui. Žinoma, kiek ir kokių produktų reikia kiekvienam patiekalui pagaminti ir kiek kuris produktas kainuoja.

Parašykite programą, kuri suskaičiuotų, kiek kainuos **kiekvienas patiekalas** ir kiek kainuos **vieno svečio** pietūs.

Duomenys

Duomenys surašyti tekstiniame faile `U2.txt`. Visi skaičiai yra sveikojo tipo.

- Pirmoje eilutėje pateikiamas visų produktų, reikalingų patiekalams gaminti, skaičius N ($1 \le N \le 10$) ir pietų komplekto patiekalų skaičius P ($1 \le P \le 12$).
- Antroje eilutėje pateikiamos visų produktų kiekio vienetų kainos centais.
- Toliau atskirose P eilučių pateikiami duomenys apie patiekalus: patiekalo pavadinimas (pirmos 15 pozicijų) ir produktų, reikalingų patiekalui pagaminti, kiekių sąrašas. Patiekalų sąraše produktai išdėstyti tokia pat tvarka, kaip ir kainų sąraše. Jeigu kuris nors produktas nenaudojamas, rašomas nulis.

Pavyzdžiui, duomenų faile užrašas

```
Salotos         5 1 0 0 2 1
```

reiškia, kad salotoms pagaminti reikia keturių produktų (galėtų būti pomidorai, svogūnai, grietinė ir druska):

- pirmo produkto, kurio kiekio vieneto kaina 12 centų, reikia 5 kiekio vienetų,
- antro produkto, kurio kiekio vieneto kaina 25 centai, reikia 1 kiekio vieneto,
- trečio ir ketvirto produktų nereikia,
- penkto produkto, kurio kiekio vieneto kaina 3 centai, reikia 2 kiekio vienetų,
- šešto produkto, kurio kiekio vieneto kaina 9 centai, reikia 1 kiekio vieneto.

Rezultatai

Rezultatai pateikiami tekstiniame faile `U2rez.txt`.

- Pirmose P eilučių reikia išvardyti visus patiekalus po vieną eilutėje: patiekalo pavadinimas ir kiek tas patiekalas kainuos centais. Pavadinimą (jam skirta 15 simbolių) nuo kainos reikia skirti vienu tarpu.
- Paskutinėje eilutėje reikia parašyti, kiek iš viso kainuos vieno svečio pietūs. (Turi būti išspausdinti du skaičiai: kiek litų ir kiek centų, atskirti vienu tarpu.)

B.3 pav.: 2010 m. IT VBE II praktinės užduoties sąlyga

Duomenų failo pavyzdys	Paaiškinimai

`6 5`
`12 25 35 2 3 9`
`Salotos 5 1 0 0 2 1`
`Kepsnys 6 3 12 9 0 0`
`Gaiva 0 0 1 15 1 0`
`Ledai Miau 0 0 5 5 5 1`
`Tortas 1 2 1 1 1 1`
	Produktų ir patiekalų skaičiai
	Produktų kiekio vienetų kainos centais
	Pirmam patiekalui reikalingų produktų sąrašas
	Antram patiekalui reikalingų produktų sąrašas
	Trečiam patiekalui reikalingų produktų sąrašas
	Ketvirtam patiekalui reikalingų produktų sąrašas
	Penktam patiekalui reikalingų produktų sąrašas

Rezultatų failo pavyzdys	**Paaiškinimai**

`Salotos 100`
`Kepsnys 585`
`Gaiva 68`
`Ledai Miau 209`
`Tortas 111`
`10 73`
	Pirmo patiekalo kaina centais
	Antro patiekalo kaina centais
	Trečio patiekalo kaina centais
	Ketvirto patiekalo kaina centais
	Penkto patiekalo kaina centais
	Vieno svečio pietūs kainuos 10 litų ir 73 centus

Nurodymai

- Programoje **būtinai** naudokite įrašo duomenų tipą ir masyvus su įrašo tipo elementais.
- Parašykite funkciją, kuri suskaičiuotų vieno patiekalo kainą centais.
- Parašykite funkciją, kuri suskaičiuotų vieno svečio pietų kainą centais.
- Programoje nenaudokite sakinių, skirtų darbui su ekranu.

Programos vertinimas

Vertinimo kriterijai	Taškai	Pastabos
Testai	20	Taškai skiriami, jeigu programa pateikia teisingus visų testų rezultatus
Teisingai skaitomi duomenys iš failo	4	Vertinama tada, kai neskiriama taškų už testus
Teisingai spausdinami rezultatai	4	
Teisingai parašytos nurodytos funkcijos	8	
Teisingos kitos procedūros ir funkcijos, jeigu jų yra, ir pagrindinė programa	4	
Teisingai aprašyti vienmačio įrašų masyvo duomenų tipas (tipai) ir kintamieji	1	Visada vertinama
Sukurtos nurodytus skaičiavimus atliekančios funkcijos	2	
Prasmingai pavadinti kintamieji. Komentuojamos programos dalys, laikomasi rašybos taisyklių	1	
Išlaikomas vientisas programos rašymo stilius, nėra sakinių, skirtų darbui su ekranu	1	
Iš viso taškų	**25**	

JUODRAŠTIS

B.4 pav.: 2010 m. IT VBE II praktinės užduoties sąlyga

II. PRAKTINĖS UŽDUOTYS

1. Šachmatų turnyras

Programos vertinimas

Vertinimo kriterijai	Taškai	Pastabos
Testai.	20	Visi taškai skiriami, jeigu programa pateikia teisingus visų testų rezultatus.
Teisingai skaitomi duomenys iš failo: • failo paruošimas skaitymui, uždarymas, baigus skaityti; • mokinių skaičius; • pirmasis ciklas, skirtas mokinių skaičiui nusakyti; • mokinio atneštų figūrų skaičių skaitymas.	4 (1) (1) (1) (1)	Vertinama tada, kai neskiriama taškų už testus.
Teisingai spausdinamas rezultatas: • failo paruošimas spausdinimui, uždarymas, baigus rašyti; • rezultato spausdinimas.	2 (1) (1)	
Sukurta funkcija, kuri suskaičiuoja, kiek šachmatų žaidimo komplektų galima sudaryti iš mokinių atneštų baltų figūrų: • pradinė minimumo reikšmė (jei `min := 100;` įvertinimas mažinamas 1 tašku); • mažiausio skaičiaus paieškos ciklas, sąlyginis sakinys arba sąlyginiai sakiniai be ciklo (jei veiksmuose naudojamas funkcijos vardas, įvertinimas mažinamas 1 tašku); • naujos minimumo reikšmės priskyrimas. *Pastaba*. Mažiausiai reikšmei rasti gali būti naudojamas rikiavimo algoritmas.	5 (2) (2) (1)	
Pėstininkų, bokštų, žirgų, rikių kiekių skaičiavimas (gali būti funkcijoje arba kitoje programos vietoje, pvz., įvedime): • pradinės sumų reikšmės; • ciklas; • sumos kaupimas (jei yra klaidų, įvertinimas mažinamas 1 tašku).	4 (1) (1) (2)	
Teisingas komplektų skaičiavimas kiekvienai figūrai (jei sprendime yra klaidų, bet iš esmės jis teisingas, skiriamas 1 taškas).	2	
Teisingos kitos procedūros ir funkcijos, jeigu jų yra, ir pagrindinė programa: • naudojamas aprašytas duomenų masyvo tipas (jei masyvas nenaudojamas arba naudojamas neprasmingai, taškas neskiriamas); • nenaudojami pagalbiniai globalieji kintamieji; • nėra sintaksės klaidų.	3 (1) (1) (1)	
Teisingai aprašyti vienmačio masyvo duomenų tipas (tipai) ir kintamieji.	1	Visada vertinama.
Sukurta nurodytus skaičiavimus atliekanti funkcija.	1	
Prasmingai pavadinti kintamieji, komentuojamos programos dalys, laikomasi rašybos taisyklių. *Pastaba*. Pakanka dviejų kriterijų.	1	
Išlaikomas vientisas programos rašymo stilius, nėra sakinių, skirtų darbui su ekranu.	2	
Iš viso taškų	**25**	

B.5 pav.: 2010 m. IT VBE I praktinės užduoties vertinimo schema

2. Gimtadienis

Programos vertinimas

Vertinimo kriterijai	Taškai	Pastabos
Testai.	20	Taškai skiriami, jeigu programa pateikia teisingus visų testų rezultatus.
Teisingai skaitomi duomenys iš failo: • teisingai skaitomi pirmos eilutės duomenys; • teisingai skaitomos produktų kainos (ciklas ir skaitymo sakinys); • teisingai skaitomas patiekalų sąrašas (ciklas, skaitymo sakinys).	4 (1) (1) (2)	
Teisingai spausdinami rezultatai: • teisingai spausdinamas patiekalų sąrašas (ciklas, spausdinimo sakinys); • centų vertimo į litus ir centus veiksmai; • teisingai spausdinama vieno svečio pietų kaina (litai, centai).	4 (2) (1) (1)	
Teisinga funkcija, kuri suskaičiuoja patiekalo kainą centais: • yra kintamasis (ne funkcijos vardas) sumai kaupti; • kainos kaupimo pradinė reikšmė nulis; • ciklas; • kainos kaupimo sakinys; • grąžinamas rezultatas.	5 (1) (1) (1) (1) (1)	Vertinama tada, kai neskiriama taškų už testus
Teisinga funkcija, kuri suskaičiuoja vieno svečio pietų kainą centais: • pradinė sumos reikšmė nulis; • skaičiavimo ciklas; • kainos kaupimo sakinys.	3 (1) (1) (1)	
Teisingos kitos procedūros ir funkcijos, jeigu jų yra, ir pagrindinė programa: • naudojamas aprašytas įrašo duomenų masyvo tipas (jei masyvas nenaudojamas arba naudojamas neprasmingai, taškas neskiriamas); • nenaudojami pagalbiniai globalieji kintamieji; • nėra sintaksės klaidų; • teisingi kreipiniai į abi funkcijas.	4 (1) (1) (1) (1)	
Teisingai aprašyti vienmačio įrašų masyvo duomenų tipas (tipai) ir kintamieji.	1	Visada vertinama
Sukurtos nurodytus skaičiavimus atliekančios funkcijos (ne tuščios).	2	
Prasmingai pavadinti kintamieji, komentuojamos programos dalys, laikomasi rašybos taisyklių. *Pastaba*. Pakanka dviejų kriterijų.	1	
Išlaikomas vientisas programos rašymo stilius, nėra sakinių, skirtų darbui su ekranu.	1	
Iš viso taškų	**25**	

B.6 pav.: 2010 m. IT VBE II praktinės užduoties vertinimo schema

Literatūra

[1] ACM Journal of Educational Resources in Computing (2005) vol. 5 n. 3. 30

[2] Lietuvos bendrojo lavinimo mokyklos bendrosios programos ir bendrojo išsilavinimo standartai XI-XII klasei. (2002). 68

[3] Švietimo ir mokslo ministro įsakymas dėl Švietimo ir mokslo ministro 2005 m. liepos 21 d. įsakymo nr. isak–1524 „dėl informacinių technologijų brandos egzaminų programos patvirtinimo“ pakeitimo, 2010 m. gruodžio 13 d., nr. v-2309. 68

[4] Ahoniemi, T. Reinikainen, T. (2006). ALOHA - a grading tool for semi-automatic assessment of mass programming courses. In *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*, p. 139–140, Uppsala, Sweden. 30

[5] Ala-Mutka, K. (2005). A Survey of Automated Assessment Approaches for Programming Assignments. *Computer Science Education*, 15(2):83–102. 17, 26, 31

[6] Anido, R. O. Menderico, R. M. (2007). Brazilian olympiad in informatics. *Olympiads in Informatics*, 1:5–14. 30

[7] Benford, S. D., Burke, E. K., Foxley, E., Higgins, C. A. (1995). The Ceilidh system for the automatic grading of students on programming courses. In *Proceedings of the 33rd annual on Southeast regional conference*, p. 176–182. 28

[8] Blonskis, J., Burbaitė, R., Dagienė, V. (2009). Informacinių technologijų valstybinio brandos egzamino praktinių užduočių ypatumai. *Informacijos mokslai*, 50. 51

[9] Blumenstein, M., Green, S., Nguyen, A., Muthukkumarasamy, V. (2004). An experimental analysis of game: a generic automated marking environment. *SIGCSE Bull.*, 36(3):67–71. 31

[10] Burton, B. A. (2008). Breaking the routine: Events to complement informatics olympiad training. *Olympiads in Informatics*, 2:5–15. 30

[11] Cadar, C., Godefroid, P., Khurshid, S., Păsăreanu, C. S., Sen, K., Tillmann, N., Visser, W. (2011). Symbolic execution for software testing in practice:

preliminary assessment. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, p. 1066–1071, New York, NY, USA. ACM. 31

[12] Carter, J., Ala-Mutka, K., Fuller, U., Dick, M., English, J., Fone, W., Sheard, J. (2003). How shall we assess this? *ACM SIGCSE Bulletin*, 35(4):107–123. 26

[13] Cheang, B., Kurnia, A., Lim, A., Oon, W. C. (2003). On automated grading of programming assignments in an academic institution. *Computers & Education*, 41:121–131. 31

[14] Colton, D., Fife, L., Thompson, A. (2006). A web-based automatic program grader. In *Proceedings of ISECON, the Conference for Information Systems Educators*, vol. 23, p. 1–9, Dallas, USA. 26

[15] Cormack, G. (2006). Random factors in IOI 2005 test case scoring. *Informatics in Education*, 5(1):5–14. 30

[16] Cormack, G., Kemkes, G., Munro, I., Vasiga, T. (2006). Structure, scoring and purpose of computing competitions. *Informatics in Education*, 5(1):15–36. 30

[17] Dagiene, V. Skupiene, J. (2004). Learning by competitions: olympiads in informatics as a tool for training high-grade skills in programming. In *Information Technology: Research and Education, 2004. ITRE 2004. 2nd International Conference on*, p. 79 – 83. 30

[18] Dijkstra, E. W. (1972). The humble programmer. *Communications of the ACM, ACM Turing Lecture*, 15(10):859–866. `http://www.cs.utexas.edu/users/EWD/transcriptions/EWD03xx/EWD340.html` [žiūrėta 2012-09-01]. 34

[19] Diks, K., Kubica, M., Stencel, K. (2007). Polish olympiads in informatics – 14 years of experience. *Olympiads in Informatics*, 1:50–56. 30

[20] Douce, C., Livingstone, D., Orwell, J. (2005). Automatic test-based assessment of programming: A review. *ACM Journal of Educational Resources in Computing*, 5(3):1–13. 27, 28, 29, 30

[21] Engels, S., Lakshmanan, V., Craig, M. (2007). Plagiarism detection using feature-based neural networks. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, SIGCSE '07, p. 34–38, New York, NY, USA. ACM. 31

[22] Fisher, M. Cox, A. (2006). Gender and programming contests: Mitigating exclusionary practices. *Informatics in Education*, 5(1):47–62. 30

[23] Forišek, M. (2006). On the suitability of programming tasks for automated evaluation. *Informatics in Education*, 5(1):63–76. 30, 74

[24] Forsythe, G. E. Wirth, N. (1965). Automatic grading programs. *Communications of the ACM*, 8(5):275–278. 27, 30

[25] Foxley, E., Higgins, C., Gibbon, C. (1996). The Ceilidh System: A General Overview 1996. 30

[26] Foxley, E., Higgins, C., Hrgazy, T., Symeonidis, P., Tsintisfas, A. (2004). *The CourseMaster CBA System: Improvements over Ceilidh.* The University of Nottingham, School of Computer Science and IT. `http://www.cs.nott.ac.uk/~cah/pdf/EIT_ImprovementsOverCeilidh_Paper.pdf` [žiūrėta 2012-09-01]. 29, 30

[27] Gottesdiener, E. (1995). Rad realities: Beyond the hype to how rad really works. *Application Development Trends*, August.:p.:28–38. 9, 52

[28] Harris, J. A., Adams, E. S., Harris, N. L. (2004). Making program grading easier (but not totally automatic). *Journal of Computing Sciences in Colleges*, 20(1):248–261. 30

[29] Helmick, M. T. (2007). Interface-based programming assignments and automatic grading of java programs. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, ITiCSE '07, p. 63–67, New York, NY, USA. ACM. 30

[30] Hext, J. B. Winings, J. W. (1969). An automatic grading scheme for simple programming exercises. *Communications of the ACM*, 12(5):272–275. 27, 30

[31] Higgins, C., Hegazy, T., Symeonidis, P., Tsintsifas, A. (2003). The CourseMarker CBA System: Improvements over Ceilidh. *Education and Information Technologies*, 8(3):287–304. 29, 30

[32] Higgins, C., Symeonidis, P., Tsinsifas, A. (2002a). Diagram-based cba using datsys and coursemaster. In *Proceedings of the International Conference on Computers in Education*, p. 367–372. 29

[33] Higgins, C., Symeonidis, P., Tsintsifas, A. (2002b). The marking system for coursemaster. *ACM SIGCSE Bulletin*, 34(3):46–50. 29

[34] Higgins, C. A., Gray, G., Symeonidis, P., Tsintsifas, A. (2005). Automated assessment and experiences of teaching programming. *J. Educ. Resour. Comput.*, 5(3). 31

[35] Hollingsworth, J. (1960). Automatic graders for programming classes. *Communications of the ACM*, 3(10):528–529. 27, 30

[36] Horvath, G. Verhoeff, T. (2002). Finding the Median under IOI conditions. *Informatics in Education*, 1(1):73–92. 30

[37] Ihantola, P., Ahoniemi, T., Karavirta, V., Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, p. 86–93, New York, NY, USA. ACM. 27

[38] Isaacson, P. C. Scott, T. A. (1989). Automating the execution of student programs. *ACM SIGCSE Bulletin*, 21(2):15–22. 28

[39] Jackson, D. (2000). A semi-automated approach to online assessment. *ACM SIGCSE Bulletin*, 32(3):164–167. 30

[40] Jackson, D. Usher, M. (1997). Grading student programs using ASSYST. *ACM SIGCSE Bulletin*, 29(1):335–339. 28, 30

[41] Joy, M., Griffiths, N., Boyatt, R. (2005). The BOSS online submission and assessment system. *ACM Journal of Educational Resources in Computing*, 5(3). 28, 30

[42] Joy, M. Luck, M. (1995). On-line submission and testing of programming assignments. In *Innovations in Computing Teaching, SEDA*, London. 28, 30

[43] Joy, M. Luck, M. (1996). A user-friendly on-line submission system. In *Proceeding of the 4th Annual Conference on Teaching of Computing*, p. 92–95. Dublin City University. 28, 30

[44] Kasanen, E., Lukka, K., Siitonen, A. (1993). The constructive approach in management accounting research. *Journal of Management Accounting Research*, 5:243 – 264. 9, 18, 48

[45] Kiryukhin, V. M. (2007). The modern contents of the Russian National Olympiads in Informatics. *Olympiads in Informatics*, 1:90–104. 30

[46] Kolstad, R. Piele, D. (2007). USA computing olympiad (USACO). *Olympiads in Informatics*, 1:105–111. 30, 42

[47] Lassenius, C., Soininen, T., Vanhanen, J. (2001). Constructive research. methodology workshop. 9, 49

[48] Leal, J. P., Queirós, R., Ferreira, D. (2010). Specifying a programming exercises evaluation service on the e-framework. In Luo, X., Spaniol, M., Wang, L., Li, Q., Nejdl, W., Zhang, W., editors, *Advances in Web-Based Learning – ICWL 2010*, vol. 6483 of *Lecture Notes in Computer Science*, p. 141–150. Springer Berlin Heidelberg. 31

[49] Leal, J. P. Silva, F. (2003). Mooshak: a web-based multi-site programming contest system. *Softw. Pract. Exper.*, 33(6):567–581. 31

[50] Lewis, S. Davies, P. (2004). Automated peer-assisted assessment of programming skills. In *Information Technology: Research and Education, 2004. ITRE 2004. 2nd International Conference on*, p. 84 – 86. 29

[51] Liang, Y., Liu, Q., Xu, J., Wang, D. (2009). The recent development of automated programming assessment. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, p. 1 –5. 27

[52] Luck, M. Joy, M. (1999). A secure on-line submission system. *Software-Practice and Experience*, 29:721–740. 30

[53] Malmi, L., Karavirta, V., Korhonen, A., Nikander, J. (2005). Experiences on automatically assessed algorithm simulation exercises with different resubmission policies. *J. Educ. Resour. Comput.*, 5(3). 31

[54] Malmi, L., Korhonen, A., Saikkonen., R. (2002). Experiences in automatic assessment on mass courses and issues for designing virtual courses. *ACM SIGCSE Bulletin*, 34(3):55–59. 31

[55] Mareš, M. (2007). Perspectives on grading systems. *Olympiads in Informatics*, 1:124–130. 30

[56] Nacionalinis egzaminų centras (2009). Informacinių technologijų valstybinio brandos egzamino programa. 68

[57] Naur, P. (1964). Automatic grading of students' algol programming. *BIT Numerical Mathematics*, 4:177–188. 10.1007/BF01956028. 27, 30

[58] Opmanis, M. (2006). Some ways to improve olympiads in informatics. *Informatics in Education*, 5(1):113–124. 30

[59] Pisan, Y., Richards, D., Sloane, A., Koncek, H., Mitchell, S. (2003). Submit! a web-based system for automatic program critiquing. In *Proceedings of the fifth Australasian conference on Computing education - Volume 20*, ACE '03, p. 59–68, Darlinghurst, Australia, Australia. Australian Computer Society, Inc. 30

[60] Pohl, W. (2006). Computer science contests for secondary school students: Approaches for classification. *Informatics in Education*, 5(1):125–132. 30

[61] Pohl, W. (2007). Computer science contests in Germany. *Olympiads in Informatics*, 1:141–148. 30

[62] Pohl, W. (2008). Manual grading in an informatics contest. *Olympiads in Informatics*, 2:122–130. 30

[63] Poranen, T., Dagiene, V., Eldhuset, A., Hyro, H., Kubica, M., Laaksonen, A., Opmanis, M., Pohl, W., Skupiene, J., Soderhjelm, P., Truu, A. (2009). Baltic olympiads in informatics: Challenges for training together. *Olympiads in Informatics*, 3:112–131. 30

[64] Queirós, R. Leal, J. P. (2012). Programming exercises evaluation systems - an interoperability survey. In *CSEDU 2012 - Proceedings of the 4th International Conference on Computer Supported Education, Porto, Portugal, 16-18 April, 2012*, vol. 1, p. 83–90. SciTePress. 9, 27, 29, 31

[65] Rahman, K. A., Ahmad, S., Nordin, M. J. (2007). The design of an automated c programming assessment using pseudo-code comparison technique. In *Proceedings of National Conference on Software Engineering and Computer Systems 2007*, Pahang, Malaysia. 30

[66] Rajaguru, D., Rajeswari, A., Bhuvaneshwari, V., Vagheesan, K. (2012). E-assessment of programming assignments in web service. In *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, p. 484 –489. 42

[67] Redwine, Jr., S. T. Riddle, W. E. (1985). Software technology maturation. In *Proceedings of the 8th international conference on Software engineering*, ICSE '85, p. 189–200, Los Alamitos, CA, USA. IEEE Computer Society Press. 9, 26, 27

[68] Reek, K. A. (1989). The TRY system -or- how to avoid testing student programs. In *Proceedings of the 20th SIGCSE technical symposium on Computer science education*, p. 112–116, USA. 28, 30

[69] Revilla, M. A., Manzoor, S., Liu, R. (2008). Competetive learning in informatics: The UVa Online Judge experience. *Olympiads in Informatics*, 2:131–148. 42

[70] Ribeiro, P. Guerreiro, P. (2009). Improving the automatic evaluation of problem solutions in programming contests. *Olympiads in Informatics*, 3:132–143. 30

[71] Romli, R., Sulaiman, S., Zamli, K. (2010). Automatic programming assessment and test data generation a review on its approaches. In *Information Technology (ITSim), 2010 International Symposium in*, vol. 3, p. 1186 –1192. 27, 31

[72] Shaw, M. (2001). The coming-of-age of software architecture research. In *Proceedings of the 23rd International Conference on Software Engineering*, ICSE '01, p. 656–, Washington, DC, USA. IEEE Computer Society. 48

[73] Skūpienė, J. (2004). Testing in informatics olympiads (in Lithuanian). In *Information Technologies Conference Proceedings*, p. 37–41, Kaunas. Technologija. 29

[74] Skūpienė, J. (2007). Automatinio sprendimų vertinimo informatikos olimpiadose raida ir perspektyvos. *Informacijos mokslai*, 42-43:43–49. 30, 43

[75] Skūpienė, J. (2010). *Evaluation of Algorithm-code Complexes in Informatics Contests*. PhD thesis, Matematikos ir informatikos institutas. 29

[76] Skupas, B. Dagiene, V. (2010). Observations from semi-automatic testing of program codes in the high school student maturity exam. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, p. 31–36, New York, NY, USA. ACM. 10, 74, 88

[77] Skupas, B., Dagiene, V., Revilla, M. (2009). Developing classification criteria for programming tasks. *ACM SIGCSE Bulletin*, 41(3):373–373. 9, 46

[78] Spacco, J., Strecker, J., Hovemeyer, D., Pugh, W. (2005). Software repository mining with Marmoset: An automated programming project snapshot and testing system. In *Proceedings of the 2005 international workshop on Mining software repositories*, p. 1–5. 30

[79] Striewe, M. Goedicke, M. (2010). Visualizing data structures in an e-learning system. In Cordeiro, J. A. M., Shishkov, B., Verbraeck, A., Helfert, M., editors, *CSEDU 2010 - Proceedings of the Second International Conference on Computer Supported Education, Valencia, Spain, April, 7-10*, vol. 1, p. 172–179. INSTICC Press. 31

[80] van der Wegt, W. (2009). Using subtasks. *Olympiads in Informatics*, 3:144–148. 30

[81] Vasiga, T., Cormack, G., Kemkes, G. (2008). What do olympiads tasks measure? *Olympiads in Infomatics*, 2:181–191. 30

[82] Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P., Queirós, R. (2012). A distributed system for learning programming on-line. *Comput. Educ.*, 58(1):1–10. 30

[83] Verhoeff, T. (2006). The IOI is (not) a science olympiad. *Informatics in Education*, 5(1):147–158. 30

[84] Verhoeff, T. (2009). 20 years of IOI competition tasks. *Olympiads in Informatics*, 3:149–166. 30

[85] von Matt, U. (1994). Kassandra: the automatic grading system. *ACM SIGCUE Outlook*, 22(1):26–40. 28, 30

[86] Wang, T., Su, X., Wang, Y., Ma, P. (2007). Semantic similarity-based grading of student programs. *Information and Software Technology*, 49(2):99–107. 30

[87] Woit, D. M. Mason, D. V. (1998). Lessons from on-line programming examinations. In *Proceedings of the 6th annual conference on the teaching of computing and the 3rd annual conference on Integrating technology into computer science education: Changing the delivery of computer science education*, p. 257–259, Ireland. Dublin City University. 30

Bronius Skūpas

PUSIAU AUTOMATINIO PROGRAMAVIMO UŽDUOČIŲ VERTINIMO IR TESTAVIMO METODAS

Daktaro disertacija
Technologijos mokslai, Informatikos inžinerija (07 T)

Bronius Skūpas

A METHOD FOR SEMI-AUTOMATIC EVALUATION AND TESTING OF PROGRAMMING ASSIGNMENTS

Doctoral Dissertation
Technological Sciences, Informatics Engineering (07 T)

2013 01 07, 5 sp. l. Tiražas 50 egz.
Parengė spaudai ir išleido
Vilniaus universiteto Matematikos ir informatikos institutas
Akademijos g. 4, LT-08663 Vilnius
Interneto svetainė: http://www.mii.lt
Spausdino Vilniaus universiteto Leidybos direkcija
Universiteto g. 3, 01513 Vilnius