KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ TINKLŲ KATEDRA

Deividas Grigas

Programinių modulių kokybės vertinimo sistema

Magistro darbas

Darbo vadovas
doc. B. Tamulynas

KAUNAS, 2004

Turinys

1 ĮVADAS

Dabartinėse mokymo įstaigose kompiuterizuotam žinių įvertinimui naudojami tik statiniai metodai, t.y. testai. Testuose pateikiami šabloniniai klausimai su galimų atsakymų pasirinkimu. Toks metodas negarantuoja objektyvaus įgytų žinių įvertinimo, nes galima atspėti teisingą atsakymo variantą arba atmintinai išmokti visus galimus atsakymus.

Praktinės studentų žinios tikrinamos darant programavimo laboratorinius darbus. Praktinio darbo atlikimui būtinas nagrinėjamos srities supratimas. Be to, vienam tokiam darbui atlikti reikia pasitelkti keleto temų žinias. Tai leidžia plačiau patikrinti studento teorines žinias ir sugebėjimus jas pritaikyti praktikoje. Taip užtikrinamas grįžtamasis ryšys tarp studento ir dėstytojo. Tačiau praktiniai darbai reikalauja daug laiko jų detaliam patikrinimui. Jų reikia atlikti keletą per teorijos kursą. Be to paprastai vienas dėstytojas tuo pačiu metu dėsto kelias disciplinas, kurių kiekvienoje jam tenka 30 ir daugiau studentų. Tad tampa fiziškai neįmanoma detaliai patikrinti visus studentų atliktus darbus. Lietuvoje nėra realizuotų kompiuterinių šių darbų tikrinimo ir įvertinimo priemonių, kurios galėtų palengvinti dėstytojo darbą.

Dažnas reiškinys studijose, o ypač moduliuose, susijusiuose su programavimu, yra plagijavimas. Tai didelė problema, kurią sunku pažaboti. 1992 m. Donald McCabe atliko apytiksliai 6000 studentų tyrimą, kurio metu paaiškėjo, kad 74% inžinerijos mokslų studentų prisipažino naudoję nesąžiningus atsiskaitymo metodus [1]. Kovai su plagijavimu yra sukurta ir kuriama nemažai programų, bet deja dauguma jų yra komercinės.

Atliekant šį magistrinį darbą, sukurta „Programinių modulių kokybės vertinimo sistema“ (PMKVS), kuri tiria studentų atliktų laboratorinių darbų kokybę šiais aspektais:

1. programų kokybės tyrimas – analizuojama, kaip programa tenkina keliamus reikalavimus remiantis tam tikrais kriterijais. Pateikta programa tiriama pagal programavimo kultūros, resursų naudojimo, saugumo, sudėtingumo ir panašius kriterijus. Atlikus analizę, pateikiamas laboratorinio darbo įvertinimas.
2. plagiato nustatymas – analizuojama, kiek studento pateiktas laboratorinis darbas yra panašus į kitų studentų pateiktus darbus.

Sistema gali įvertinti programos teksto kokybę bei atitikimą programavimo standartams ir nustatyti studentų laboratorinių darbų tarpusavio panašumo lygį. Programos teksto kokybės įvertinimo savybė gali pasitarnauti studentams, nes ji išryškina programavimo sritis, kuriose reikia pasitobulinti. Programavimo standartų, kultūros laikymasis tekstą padaro ne tik patrauklesniu žmogaus akiai, labiau įskaitomu, suprantamesniu, bet ir sumažina

tikimybę padaryti klaidas, kas taupo pačio studento laiką. Naudodamasis PMKVS plagiato nustatymu, dėstytojas gali nesunkiai identifikuoti nesąžiningus studentus. Ši programos galimybė turi ir prevencinę įtaką – žinodami, kad jų darbus tikrina kompiuteris, daugelis studentų vengs plagijuoti. Taip pat bus sudrausminami ir studentai leidžiantys kopijuoti savo darbus.

Kuriant PMKVS, jai buvo keliami šie pagrindiniai tikslai:

- Sistema turi įvertinti programų, parašytų JAVA programavimo kalba, kokybę ir nustatyti galimus plagijavimo atvejus.
- Dėstytojui turi būti suteikta galimybė kontroliuoti vertinimo procesą. Kokybė yra abstraktus terminas, todėl palikti įvertinimą išskirtinai sistemai negalima, nes pastaroji vertina pagal konkrečiai apibrėžtus ir suprogramuotus kriterijus. Dėstytojui reikia palikti galimybę eksperimentuoti keičiant naudojamus kriterijus, jų vertinimą bei įtaką galutiniam pažymiui.
- Rezultatai pateikiami programoje ir išvedami į išorines laikmenas. Tyrimų rezultatai gali būti pateikiami studentams internete. Tam programoje numatytas rezultatų išsaugojimas HTML formatu. Kadangi ateityje gali prireikti panaudoti rezultatus kitais tikslais, pasinaudota XML formato teikiamomis galimybėmis – dėl savo savybių XML yra plačiai naudojamas duomenų konvertavime (pvz. pasinaudojant XSL standartu, XML galima konvertuoti į HTML), kas suteikia galimybę pritaikyti rezultatus įvairiose sferose.
- Lengvai įsisavinamas, gerai dokumentuotas programos tekstas. Atsižvelgiant į tai, kad programa ateityje greičiausiai bus tobulinama kitų studentų, siekta, kad programos tekstas būtų lengvai suprantamas. Taisant ar papildant kitų programuotojų sukurtas sistemas, sunkiausia yra įsisavinti programos tekstą. Dažnai tai būna sunkiau, nei parašyti naują programą. Pagrindinė to priežastis – prasta dokumentacija arba jos nebuvimas. Kadangi sistema parašyta C# (tarimas: „si šarp“) programavimo kalba, pasinaudota jos galimybe dokumentuoti programinį kodą komentaruose. Ši iš JAVA kalbos pasiskolinta savybė gana sėkmingai išsprendžia daugelio programuotojų nemėgstamą dokumentacijos klausimą. Programiniame tekste parašytą dokumentaciją vėliau galima išgauti specialių įrankių pagalba (pvz. atviro kodo programa NDoc) ir sugeneruoti standartinę dokumentaciją. Pastaroji lengvai suprantama, nes joje prisilaikoma MSDN dokumentacijoje naudojamų standartų. Kitas būdas palengvinti programos teksto įsisavinimą yra naudoti programavimo standartus visame tekste. C# yra pakankamai jauna programavimo

kalba ir neturi nusistovėjusių programavimo standartų. Be to ji daugeliu savybių yra labai panaši į JAVA kalbą. Todėl pasirinktas Scott W. Ambler parašytas standartas JAVA programavimo kalbai [2].

PMKVS yra bakalaurinio darbo tuo pačiu pavadinimu [3] tęsinys. Dalis bakalauriniame darbe sukurtų principų sėkmingai panaudoti PMKVS sistemoje. Prototipinė bakalaurinio darbo programa nagrinėjo C++ programavimo kalba parašytus studentų laboratorinius darbus. Šiame magistriniame darbe sukurta sistema tiria JAVA kalba parašytus laboratorinius darbus. Savo sintakse JAVA yra panaši į C++, tačiau daugeliu kitų savybių labai skiriasi. Viena pagrindinių savybių turėjusių įtakos projektuojant PMKVS yra Javos buvimas visiškai objektine programavimo kalba, tuo tarpu C++ yra ir objektinė, ir funkcinė. Kita svarbi savybė – paprasta sintaksė (aprašyti JAVA kalbos sintaksę yra daug paprasčiau nei C++). Šie dalykai privertė į sprendžiamą problemą pažvelgti iš kitos pusės.

Sukurtą sistemą numatoma įtraukti į kompleksinę studentų žinių kontrolės sistemą TestTool, kuri jau keletą metų taikoma programavimo disciplinų moduliuose: „Duomenų struktūros ir algoritmų analizė“ (kodas P175B353) ir „Intelektualiųjų informacinių sistemų kursinis projektas“ (kodas P175S336).

2 SISTEMOS ANALIZĖ

Šiame skyriuje pateikiama problemos, kurią buvo bandoma spręsti projektuojant sistemą, analizė.

2.1 Rinkos analizė

Turimais duomenimis Lietuvoje analogiškų sistemų nėra. Užsienyje tokios sistemos egzistuoja. Pavyzdys galėtų būti sistema *Ceilidh* arba atnaujintas, visiškai perdarytas šios sistemos variantas – *CourseMaker*, kuri dabar naudojama Didžiosios Britanijos universitetuose. Abi sistemos yra komercinės ir platinamos už tam tikrą mokestį. Jos turi programų kokybės ir plagijavimo atvejų nustatymų galimybes. Kuriant PMKVS kokybės įvertinimo modulį, dalinai remtasi *Ceilidh* svetainėje pateikta dokumentacija.

Sistemų, kurios nagrinėja tik plagijavimo nustatymą yra daugiau – SIM, YAP, JPlag ir kitos. Jos skiriasi savo sudėtingumo lygiu ir nagrinėjamomis programavimo kalbomis.

2.2 Techninė sistemos analizė

Sistema veiks viename kompiuteryje, tad jai nereikalingas ryšys su internetu. Sistemai pakanka kompiuterio su x86 šeimos procesoriumi ir 64 MB RAM. Papildomų periferinių įrenginių sistema nenaudoja, tad jie nėra būtini.

Naudojama MS Windows šeimos (Win9x/NT/2000/XP) operacinė sistema. Remiantis tuo, reikalinga techninė įranga priklauso nuo naudojamos OS ir turi bent minimaliai atitikti naudojamos OS reikalavimus. Sąsajos realizavimui naudojami standartiniai Windows API elementai: mygtukai, meniu, lentelės ir t.t. Realizacijai pasirinkta C# programavimo kalba. Ji kaip ir JAVA yra visiškai objektinė, kas užtikrina sistemos lankstumą. Kadangi C# remiasi Microsoft .NET technologijomis, kompiuteryje, kuriame bus diegiama programa, turi būti instaliuota Microsoft .NET Framework 1.1.

2.3 Kokybės vertinimo modulis

2.3.1 Kokybės vertinimo modulio apibrėžimas

PMKVS kokybės modulis tiria studentų programavimo laboratorinius darbus, parašytus JAVA programavimo kalba, pagal pasirinktus kokybės kriterijus. Turint kiekvieno pasirinkto kriterijaus įvertinimą ir jų svorius, paskaičiuojamas bendras programos įvertinimas. Įvertinimai pateikiami dešimtbalėje vertinimo sistemoje.

2.3.2 Aktualumas

Kokybės vertinimo modulis padės dėstytojui įvertinti programos teksto kokybę. Dėstytojas neturėtų suteikti didelės įtakos kokybės įvertinimui laboratorinio darbo pažymyje. Tačiau šis įvertinimas būtų paskata studentams skirti daugiau dėmesio kokybiškam programų rašymui. Tyrimai rodo, kad kokybiškai parašytą programos tekstą yra lengviau suprasti, jame padaroma mažiau klaidų, sutaupomas laikas. Rekomenduotina įtaka studento laboratorinio darbo pažymiui – 10%. Pabrėžtina, kad studentų laboratorinių darbų kokybė turėtų būti vertinama tik tuo atveju, kai jiems yra išdėstyti kokybės kriterijai.

2.3.3 Modulio analizė

Vertinant kokybę, reikia apibrėžti norimus programų vertinimo kriterijus. Galima išskirti 3 kokybės kriterijų grupes:

- **Programos teisingumas.** Tikrinimas ar programoje nėra sintaksinių ir loginių klaidų. Patikrinti ar programoje nėra sintaksinių klaidų nebūtų sudėtinga, tereiktų JAVA kompiliatoriaus. Tačiau loginių klaidų suradimas yra labai sudėtingas. Norint jį realizuoti, tektų iš esmės pakeisti laboratorinių darbų organizavimą. Visi studentai turėtų gauti vienodas užduotis ir kiekvienai tokiai užduočiai turėtų būti suprogramuoti specialūs testai programos išvedamų rezultatų patikrinimui.Be to rezultatų išvedimas laboratoriniuose darbuose turėtų būti standartizuotas. Priešingu atveju būtų neįmanoma patikrinti programų loginį teisingumą. Dėl šių priežasčių PMKVS kokybės vertinimo modulyje nutarta nenagrinėti kokybės pagal programos teisingumo kriterijus.
- **Programos sudėtingumas.** Tikrinama ar sudėtinga suprasti tam tikrus programoje naudojamus kodo fragmentus. Programuojant, kaip ir kalbant, vieną ir tą patį dalyką galima apibrėžti skirtingais būdais. Tačiau vienus sakinius suprasti būna lengviau, o kitus sunkiau. Todėl daugelis programuotojų profesionalų pasaulyje laikosi tam tikrų programavimo standartų.
- **Programavimo kultūros laikymasis.** Programavimo kultūra yra vienas svarbiausių aspektų siekiant parašyti kokybišką programos tekstą. Tyrimai rodo, kad vis daugiau laiko yra skiriama ne programų kūrimui, o jų prižiūrėjimui, klaidų taisymui. Kadangi programavimo kultūra turi labai didelę įtaką kodo suvokimui, nustatytų programavimo kultūros normų laikymasis padeda išvengti klaidų, lengviau surasti padarytas klaidas ar greičiau suprasti kolegos parašytą programą.

Programavimo kultūra ne tik nereikalauja didelių pastangų iš programuotojo, bet dar ir taupo jo laiką.

Programos sudėtingumo ir programavimo kultūros laikymosi kriterijus galima tirti ieškant pasikartojančių programos kodo dalių (pvz. raktinių žodžių) ir prireikus nagrinėjant jų semantiką. Realizuojant programų *kokybės vertinimo modulį* reikia atkreipti dėmesį į šiuos aspektus:

- Kokybės tyrimui programos struktūra didelės įtakos neturi. Todėl kokybę galima tirti analizuojant programos kodą po eilutę.
- Kriterijai gali turėti nevienodą svarbą, todėl apskaičiuojant bendrą kokybės įvertinimą, reikia atsižvelgti į kriterijui suteiktą svorį.
- Esant reikalui, naujai sukurti kriterijai turėtų būti lengvai įterpiami tarp jau esančių, arba ankstesni lengvai pakeičiami naujais.
- Tyrimo procesą galima optimizuoti detaliai nenagrinėjant tuščių eilučių ir komentarų. Yra fiksuojamas tik faktas, kad tam tikras jų kiekis yra programos tekste.
- Pageidautina, kad būtų pateikiamos priežastys, kodėl tam tikras kriterijus yra įvertintas būtent tokiu balu.

2.3.4 Duomenų srautai

Tyrimams numatyti du režimai: failų tyrimas ir katalogų (projektų) tyrimas. Failų tyrimas skirtas detalesniam vieno studento laboratorinio darbo tyrimui (nes paprastai laboratorinį darbą sudaro keletas failų), pateikiant kiekvieno failo įvertinimą. Tuo tarpu katalogų tyrimas labiau tinka tiriant keleto studentų laboratorinius darbus.

Kokybės vertinimo modulis gauna tiriamų failų ar katalogų sąrašą ir tyrimui pasirinktų kriterijų sąrašą, o grąžina kiekvieno failo ar katalogo įvertinimus pagal visus kriterijus dešimtbalėje sistemoje. Bendras pažymys nesunkiai paskaičiuojamas, atsižvelgiant į kiekvieno kriterijaus svorį ir gautą įvertinimą. Dirbant katalogų tyrimo režimu reikia atsižvelgti ir į kiekviename faile esančių kodo dalių panaudojimo atvejų kiekį. Kitaip įvertinimas gali būti neobjektyvus, nes programinio teksto kiekis failuose gali būti netolygiai paskirstytas.

Rezultatai pateikiami lentelėse ir gali būti išsaugoti HTML ar XML formatu.

2.3.5 Vertinimo kriterijų apibrėžimas ir analizė

1. **Komentarų kiekis**. Programavimo kultūros grupei priklausantis kriterijus. Paprastai programos tekste komentarai turėtų sudaryti 10% – 60% bendro programos teksto ilgio. Jei programos kodas nėra komentuojamas, kitiems programuotojams ir net

pačiam kodo autoriui po tam tikro laiko tampa sunku jį suprasti. Padidėja tikimybė padaryti klaidų ir eikvojamas laikas. Kita vertus, jei programos kode komentarų yra per daug, jie užgožia pačią programą.

Informatyvumo požiūriu šis kriterijus būtų bene pats informatyviausias, tačiau norint automatizuoti testavimą ir rašant algoritmus komentarų paieškai, bent iš dalies reikėtų atsižvelgti į paties komentaro turinį. Pvz. neturėtų būti skaičiuojami tušti, ar neinformatyvūs komentarai, tokie kaip „// „„ ar „/* */“, „//***“, „/*****/“ ir pan. Tačiau šį procesą automatizuoti taip, kad būtų pasiektas 100% objektyvumas, praktiškai neįmanoma.

Realizuojant šį kriterijų, taip pat svarbu pastebėti, kad komentarų pradžios simbolius galima rasti simbolinėse konstantose, bet šiuo atveju jie nereikš komentaro pradžios. Be to komentarai gali tęstis per keletą eilučių, todėl, jei kodas nagrinėjamas po vieną eilutę, reikia įsiminti paskutinės tirtos eilutės būseną.

2. **Tuščių eilučių kiekis**. Programavimo kultūros grupei priklausantis kriterijus. Tuščios eilutės turėtų sudaryti 15% – 30% bendro programos teksto ilgio. Didelis tuščių eilučių kiekis apsunkina programos teksto skaitymą. Tačiau kuomet programos tekstas yra labai kompaktiškas, darosi sunku jį perskaityti.

   Šis kriterijus gana informatyvus. Reikėtų pamąstyti apie tai, jog kai kurie programuotojai savo programos kode mėgsta vietoj tuščių eilučių palikti komentuotas eilutes su žvaigždutėmis (//*****) arba, kaip yra kai kuriose sistemose – automatiškai tarp metodų įterpiama komentuota eilutė. Pvz. „C++ Builder“ įterpia (//-----) tarp metodų.

   Windows operacinėse sistemose eilutės pabaigai žymėti naudojami du simboliai ("\r\n"), tuo tarpu kitose OS naudojamas vienas simbolis ('\n'). Reikia atsižvelgti į abu variantus.

3. **Klasių ir metodų vardų, prasidedančių iš mažosios raidės, kiekis.** Jie turi prasidėti iš didžiosios raidės, nes tai leidžia geriau skirti kintamųjų vardus nuo klasių ar metodų vardų. Kintamųjų vardai turėtų prasidėti iš mažosios raidės. Taip palengvėja programos teksto skaitymas.

   Šis kriterijus neturėtų į skaičiavimus įtraukti iš sisteminių klasių paveldimų metodų, tačiau nustatyti ar metodas paveldėtas iš sisteminės klasės yra ganėtinai sudėtinga.

4. **Vidutinis programos sakinių ilgis**. Nuo šio kriterijaus priklauso paties programos teksto sudėtingumas. Vartotojui yra sunkiai suprantami ilgi sakiniai, todėl klaidų

tikimybė didėja. Be to ilgi sakiniai gali netilpti į redaktoriaus ekraną. Patartina ilgus sakinius skaidyti į kelis trumpesnius.

Kriterijaus realizacija yra paprasta, o informatyvumas didelis. Kokybės nustatymo prasme šis kriterijus turi didelę reikšmę, nes nepatyrę programuotojai linkę rašyti labai ilgus kodo sakinius.

5. **Skaitinių išraiškų, kurios turi būti konstantos, kiekis**. Jei keičiasi uždavinio sąlyga (pavyzdžiui padidėja masyvo dydis), tuomet nereikės atlikti keleto pakeitimų visame programos tekste. Reikės tik pakeisti konstantos reikšmę.

   Kriterijus svarbus, tačiau laboratoriniuose darbuose paprastai panaudojami 1-2 masyvai, todėl kyla įvertinimo objektyvumo problema. Kriterijaus svoris (įtaka bendram pažymiui) turėtų būti mažas.

6. **Neteisingų intervalo rėžių aprašymų kiekis for cikluose.** Nurodant intervalo viršutinį rėžį, jis turi būti aprašomas neimtinai. Tada intervalo dydis yra lygus rėžių skirtumui. Taip yra aiškiau suprasti kiek kartų bus kartojamas ciklas.

   Pvz.: `for (int i = 0; i <= 98; i++)` turėtų būti `for (int i = 0; i < 99; i++)`

   Šis kriterijus nėra labai informatyvus, o jo įvertinimas labai priklauso nuo programoje panaudotų for ciklų kiekio. Kuo mažiau ciklų panaudota, tuo neobjektyvesnis tampa įvertinimas. Todėl šio kriterijaus svoris turėtų būti mažas.

7. ***switch* raktinių žodžių naudojimas**. Jų naudojimas blogina programos struktūrą, didina jos sudėtingumą. Naudoti *switch* galima, tačiau geriau sumažinti jo panaudojimų kiekį, keičiant *switch* į *if* sakinius ar kitomis priemonėmis.

   Kriterijus vėlgi labai priklauso nuo panaudojimų kiekio.

8. ***Tabuliacijos simbolių naudojimas.*** Įvairiuose teksto redaktoriuose tabuliacijos simbolį atitinka nevienodas tarpo simbolių skaičius. Todėl rekomenduojama atitraukiant tekstą nuo kairiojo krašto naudoti tarpo simbolius. Priešingu atveju programos kodas kitame redaktoriuje gali atrodyti tarsi bet kaip išmėtytas.

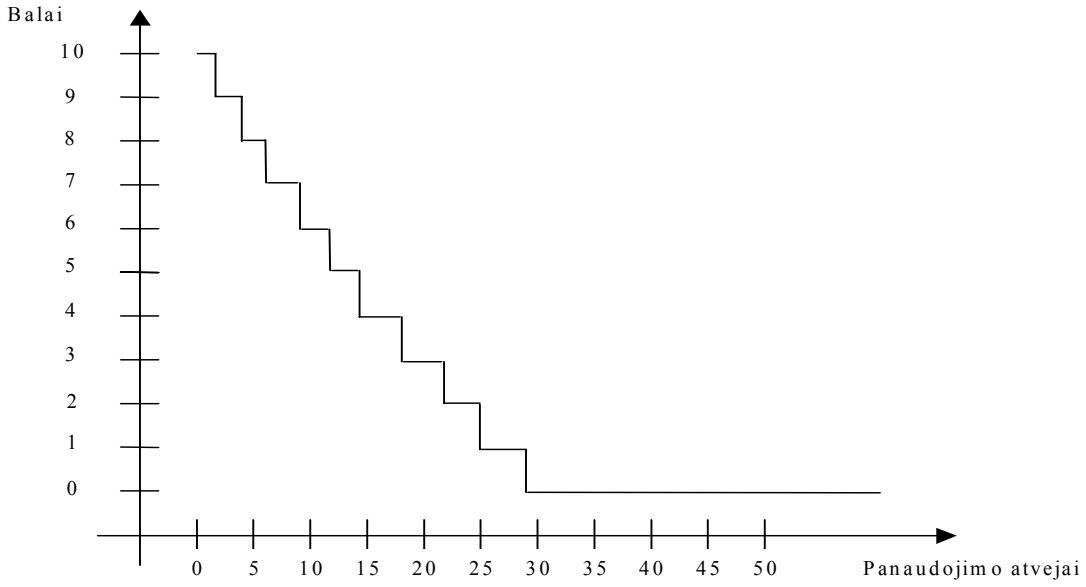
   Kriterijus svarbus, tačiau jei studentas savo laboratoriniame darbe panaudojo tabuliacijos simbolį bent vieną kartą, tai labai tikėtina, kad jis tiesiog nežino apie jų neigiamą įtaką programos teksto kokybei ir naudoja dažnai. Vėlgi kyla įvertinimo objektyvumo problema. Kriterijaus svoris turėtų būti mažas.

2.3.6 Rezultatų įvertinimo metodai

Atlikus kokybės tyrimą, gautus rezultatus galima vertinti dviem būdais:

1. Gautą rezultatą galima vertinti intervalų metodu. Kiekvienam įvertinimo balui sudaromi galimų rezultatų intervalai (vertinimo skalė). Vertinant yra tikrinama, į kurį vertinimo skalės intervalą patenka rezultatas, gautas tyrimo metu. Pvz.: raktinį žodį *switch* programoje galima naudoti, bet jei jis panaudotas nuo 2 iki 4 kartų, tuomet studento gaunamas įvertinimas už šį kriterijų bus 9. Jei *switch* panaudotas nuo 4 iki 6 kartų, įvertinimas – 8, *switch* panaudotas nuo 6 iki 9 kartų, įvertinimas – 7 ir t.t.

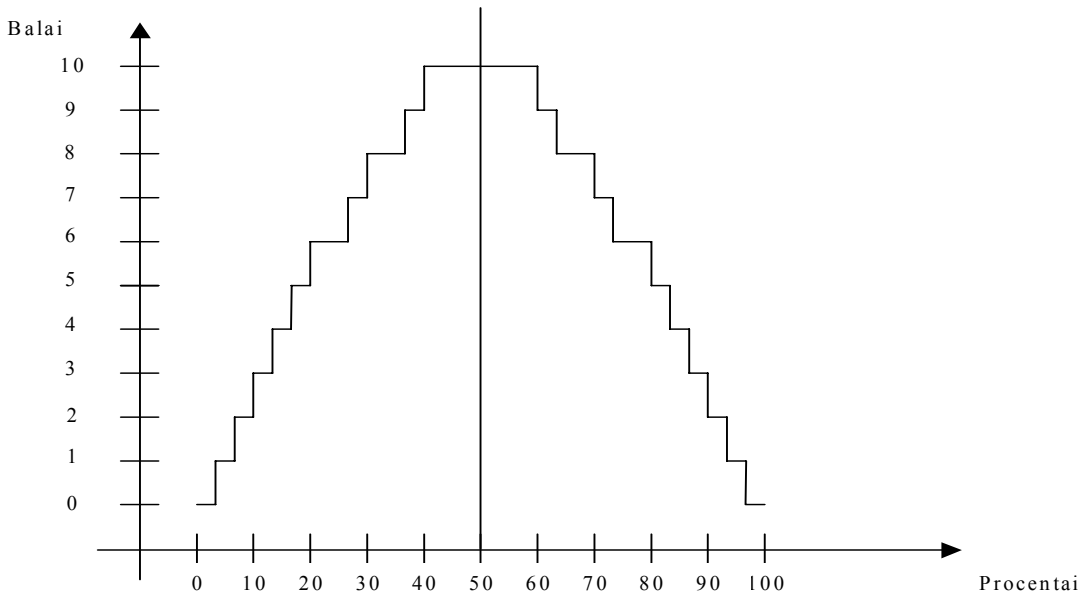
Vertinimo skalė: 10: 0 – 2; 9: 2 – 4; 8: 4 – 6; 7: 6 – 9; 6: 9 – 12; 5: 12 – 14; 4: 14 –18; 3: 18 – 22; 2: 22 – 25; 1: 28 – 29; 0: > 29.



1 pav. Kriterijų vertinimo skalė pagal intervalų metodą.

2. Kriterijaus rezultatą galima vertinti pagal Gauso dėsnį. Paprastai šiuo būdu yra vertinami kriterijai, kuriuose skaičiuojamas kažkoks santykis. Vertinimo skalės sudarymas yra panašus kaip ir intervalų metode. Skirtumas tik tas, kad šio metodo vertinimo skalė yra simetriška galimam tyrimo rezultatui, kuriam esant gaunamas maksimalus įvertinimas. Pvz.: tarkim, geriausias atvejis, kuomet komentarai programoje sudaro nuo 40 iki 60% teksto. Tada vertinimo skalė bus simetriška rezultatui, lygiam 50%.

Vertinimo skalė: 10: 50 – 40%; 9: 40 – 35%; 8: 35 – 29%; 7: 29 – 25%; 6: 25 – 21%; 5: 21 – 17%; 4: 17 – 14%; 3: 14 – 10%; 2:10 – 8%; 1: 8 – 2%; 0: 2 – 0%. Ši skalė yra simetriška 50% rezultatui.



2 pav. Vertinimo skalė pagal Gauso dėsnį

Šie vertinimo metodai aprašyti N. E. Fenton knygoje“Software Metrics. A rigorous approach“ ir yra paremti statistiniu programos teksto tyrimu.

Apskaičiavus kiekvieno kriterijaus įvertį, galima paskaičiuoti bendrą tiriamos programos kokybės įvertinimą. Jis paskaičiuojamas pagal formulę:

$$P = 10 - \left(\frac{\sum_{i=1}^{n} (X_i \times (10 - Q_i))}{\sum_{j=1}^{n} X_j} \right)$$

Čia:

P – bendras tiriamos programos įvertinimas;

X_i – i–tojo kriterijaus svoris. Vardiklyje yra visų kriterijų svorių suma;

Q_i – i–tojo kriterijaus įvertis;

Skaičius *10* šioje formulėje naudojamas todėl, kad programų vertinimui buvo pasirinkta dešimtbalė vertinimo sistema.

2.3.7 Apribojimai

Vertinamos programos turi būti sintaksiškai teisingos, t.y. JAVA kalbos kompiliatorius neturi rasti jose sintaksinių klaidų. Nors modulis nagrinės ir sintaksiškai klaidingas programas, tačiau gautas įvertinimas gali neatitikti tikrovės.

2.4 Plagiato nustatymo modulis

2.4.1 Plagiato nustatymo modulio apibrėžimas

PMKVS plagiato nustatymo modulis tiria studentų programavimo laboratorinių darbų, parašytų JAVA programavimo kalba, tarpusavio panašumo lygį. Iš nustatyto panašumo lygio matosi galimi programų plagijavimo atvejai. Laboratorinių darbų programos padarytos pagal vienodą užduotį, bet skirtingų studentų. Plagiato nustatymo modulis yra pagalbinė priemonė dėstytojui, tikrinančiam studentų darbus atliktus JAVA programavimo kalba. Jis skirtas padėti dėstytojui nustatyti galimus studentų sukčiavimo atvejus.

2.4.2 Aktualumas

Plagijavimo atvejų nustatymas dėstytojui yra daug laiko resursų reikalaujantis uždavinys. Pilnai patikrinti visus studentų darbus ir nustatyti sukčiaujančius studentus žmogui fiziškai beveik neįmanoma. PMKVS plagiato nustatymo modulis pasinaudoja kompiuterių teikiamomis galimybėmis atlikti rutininį darbą. To dėka sumažėja sukčiavimo atvejų skaičius tarp studentų, o dėstytojas gali daugiau laiko skirti kūrybiniam ar kitokiam darbui.

2.4.3 Užsienio literatūros šaltiniuose pateiktų sprendimų problemai spręsti lyginamoji analizė

Plagiatų nustatymo programos nėra retas reiškinys pasaulyje. Plagijavimo problema egzistuoja seniai, todėl natūralu, kad atsirado norinčių ją pažaboti. Sukurtos programos skiriasi savo sudėtingumo lygiu. Kai kurios jų naudoja sudėtingus matematinius metodus, kitos remiasi paprasta logika, semantiniu teksto nagrinėjimu. Pateiksime keletą kitose sistemose naudojamų šios problemos sprendimo būdų:

- **SIM (Software Similarity Testor).** SIM kūrėjas yra D. Grune. Jis teigia, kad SIM gali nustatyti vienodus kodo (arba tiesiog teksto, jei lyginant ne programas) fragmentus. SIM dirba trejais etapais. Pirmojo etapo metu nuskaitomas failas ir sugeneruojama vienodų kodo fragmentų lentelė. Sistema padalina kodą į mažus fragmentus (pvz. dalinant kodą „abcdefgh“ į 5 simbolių dydžio fragmentus, gautume „abcde“, „bcdef“, „cdefg“ ir „defgh“) ir suranda tuos, kurie sutampa abiejose programose. Antrojo etapo metu surandamos šių kodo fragmentų pozicijos failuose. Trečiojo etapo metu šie kodo fragmentai yra atspausdinami.
- **YAP (Yet Another Plague).** YAP gali nustatyti daugelį plagijavimo atvejų – komentarų, kintamųjų vardų pakeitimo, sakinių sukeitimą vietomis ir t.t. YAP panašumus tarp programų nustato dvejais etapais. Pirmojo etapo metu:

- Iš programų pašalinami komentarai ir išvedimo eilutės.
- Didžiosios raidės keičiamos mažosiomis.
- Pašalinamos raidės, nerastos apibrėžtuose kintamuosiuose.
- Suformuojamas kodo žymių (angl. token) sąrašas.
- Vienodą prasmę turintys kodo sakiniai perdaromi pagal vieną formą.

Antrojo etapo metu panašumams rasti skirtingos YAP versijos naudoja nevienodus metodus [12]. YAP1 naudoja keletą Unix įrankių ir Bourne-shell skriptą. YAP2 naudoja Hekelio algoritmą [16]. YAP3 – RKR-GST (Running-Karp-Rabin Greedy-String-Tiling) algoritmą [16].

- **JPlag** panašumų ieško tarp programinio kodo failų rinkinių. Ši sistema tiria JAVA, C, C++, Scheme programų kodus ir natūralių kalbų tekstą. JPlag dirba dvejais etapais. Pirmojo etapo metu sudaromas žymių sąrašas. Antrojo etapo metu žymių eilutės lyginamos tarpusavyje ieškant panašumų. Čia naudojamas panašus metodas kaip ir SIM sistemoje – žymių eilutės sudalinamos į mažus fragmentus ir ieškoma vienodų fragmentų. Panašumo lygis nustatomas pagal vienodų fragmentų kiekį.

2.4.4 Modulio analizė

Realizuojant *plagiato nustatymo* modulį, reikia atsižvelgti į tokias pagrindines problemas:

- Galima lengvai pakeisti programavimo kultūrą. Programą galima papildyti tarpais, tuščiomis eilutėmis, komentarais. Šitaip pakeista programa nuo pradinio varianto skiriasi tik vizualiai, tuo tarpu jos vykdymas yra identiškas pradinei programai.
- Programos modulius (metodus, klases) galima sukeisti vietomis, iš vieno failo perkelti į keletą ir pan. JAVA kalboje modulių išdėstymas neturi reikšminio skirtumo.
- Galima pakeisti kintamųjų bei metodų vardus. Čia reikia atsižvelgti į kintamųjų galiojimo sritį. Skirtinguose metoduose arba tiesiog skirtingose galiojimo srityse vienodai pavadintas kintamasis gali turėti skirtingą prasmę.
- Keletą sakinių galima išdėstyti vienoje eilutėje, arba vieną sakinį galima išdėstyti keliose eilutėse.
- Galima pakeisti duomenis, kuriais operuoja programa. Bene dažniausiai tarp studentų pasitaikantis plagijavimo atvejis yra programų pritaikymas darbui su kitais duomenimis. Beveik visada tam nereikia žinių apie programos arba atskirų jos modulių realizaciją. Šitaip pakeitus programą, ji skirtųsi nuo

pradinio varianto t.y. sukompiliuotas kodas nebūtų identiškas. Tačiau programos realizacija išlieka ta pati. Pagrindinis skirtumas būna pradiniai duomenys ir rezultatai.

Atsižvelgiant į paminėtas problemas, PMKVS plagiato nustatymo modulyje nuspręsta pasinaudoti kitose plagijavimo nustatymo sistemose naudojamų metodų mišiniu. Tačiau modulyje atsisakyta YAP ir JPlag sistemose naudojamo fragmentų palyginimo principo. Šis principas efektyvus, kai fragmentų dydis nėra per didelis ar per mažas, tačiau jis gali būti pakankamai lėtas. PMKVS plagiato nustatymo moduliui sukurtas naujas (žinių, kad kitose sistemose būtų naudojamas panašus metodas, tyrimo metu nerasta) metodas, kurio pagrindinis tikslas yra pašalinti iš programinio kodo neapibrėžtumus (komentarai, tarpai, konstantos, kintamųjų vardai), išskirti originalius (nepasikartojančius) kodo sakinius ir surasti sutapusių kiekį. Šis principas iš dalies panašus į naudojamą YAP sistemoje, tačiau turėtų būti greitesnis nors ir ne toks tikslus. Teoriškai 20 simbolių eilutę, padalinus į 5 simbolių fragmentus, gautume 16 fragmentų, kuriuos reiktų lyginti su kitos programos eilutėje rastais fragmentais. Lyginant dvi 20 simbolių eilutes, maksimalus palyginimų skaičius lygus 20*20=400. Tuo tarpu lyginant tas pačias eilutes naujai sukurtu metodu, palyginimų skaičius visada lygus vienam.

Atlikus bandomuosius tyrimus su bakalauriniame darbe padarytu prototipu (kuriame buvo naudojamas panašus principas kaip ir šiame magistro darbe), pastebėta, kad programa nustato daugelį plagijavimo atvejų.

2.4.5 Apribojimai

Vertinamos programos turi būti sintaksiškai teisingos, t.y. JAVA kalbos kompiliatorius neturi rasti jose sintaksinių klaidų.

2.4.6 Duomenų srautai

Tyrimams numatyti du režimai: failų tyrimas ir katalogų (projektų) tyrimas. Failų tyrimas analizuoja konkrečius failus, tuo tarpu katalogų tyrimas analizuoja visus nurodytuose kataloguose rastus **.java* failus. Paprastai tiriant studentų laboratorinius darbus labiau tinka katalogų tyrimas, nes dažniausiai vieną laboratorinį darbą sudaro keletas failų.

Plagiato nustatymo modulis gauna tiriamų failų ar katalogų sąrašą, o grąžina kiekvieno failo ar katalogo panašumo į kitus failus ar katalogus įvertinimus. Grąžinama reikšmė nurodo failų ar katalogų panašumo lygį ir gali kisti intervale 0%-100%. 100% reikšmė reiškia, kad plagijavimo tikimybė didžiausia, 0 – mažiausia.

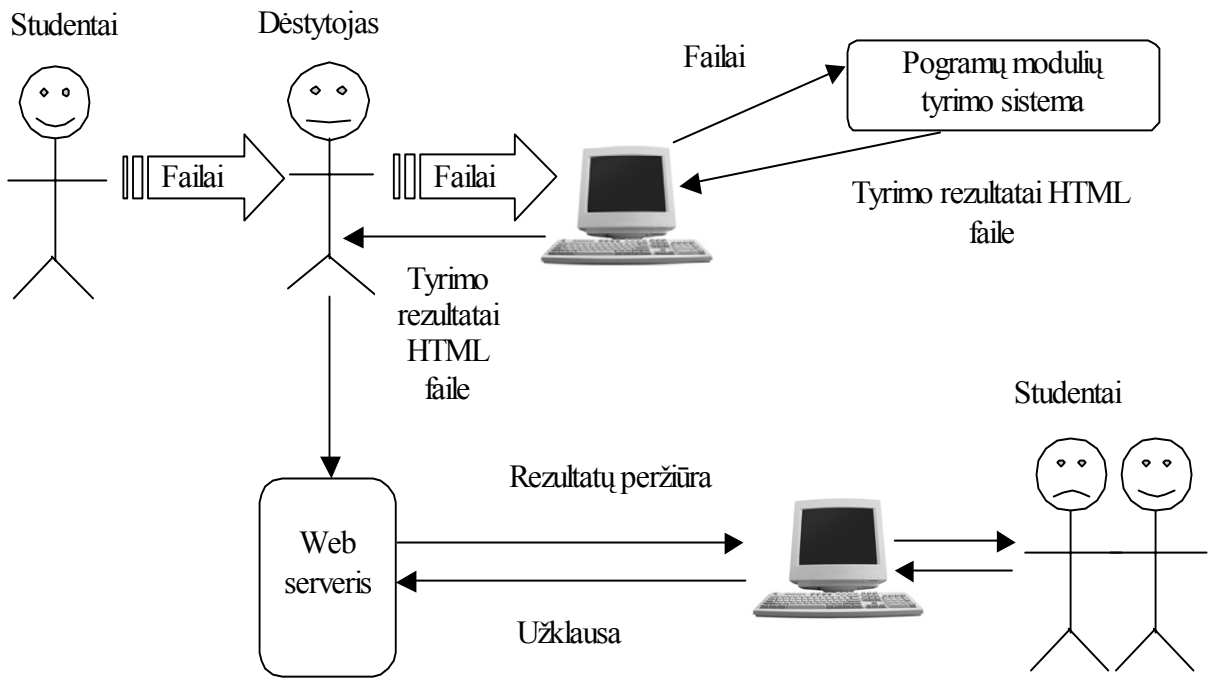
Rezultatai pateikiami lentelėse ir gali būti išsaugoti HTML ar XML formatu.

2.5 Eksploatavimo aplinka

Sukurtai sistemai nereikalingas tiesioginis ryšys su kompiuteriniu tinklu. Sistemos tyrimų objektas yra programų tekstai saugomi failuose. Šiuos failus dėstytojas surenka iš studentų, arba studentai atneša juos dėstytojui. Surinktus failus dėstytojas patalpina į kompiuterį kuriame instaliuota PMKVS sistema. Kiekvieno studento failus patartina išsaugoti atskiruose kataloguose.

Vartotojo patogumui, tiriamų katalogų pavadinimus patartina sudaryti pagal tokią schemą: pavadinimas susideda iš pirmų 4–ių studento pavardės ir pirmų 4–ių vardo raidžių. Pavyzdžiui: jei tiriamą programą parašė studentas *Jonas Petraitis*, tai katalogas turėtų vadintis *jonapetr.java*.

Rezultatus galima saugoti HTML faile, kurį dėstytojas gali patalpinti Web serveryje, suteikiant studentams galimybę gautus rezultatus peržiūrėti, naudojantis tinklo naršykle.



3 pav. Išoriniai sistemos duomenų srautai (studentų darbus surenka pats dėstytojas)

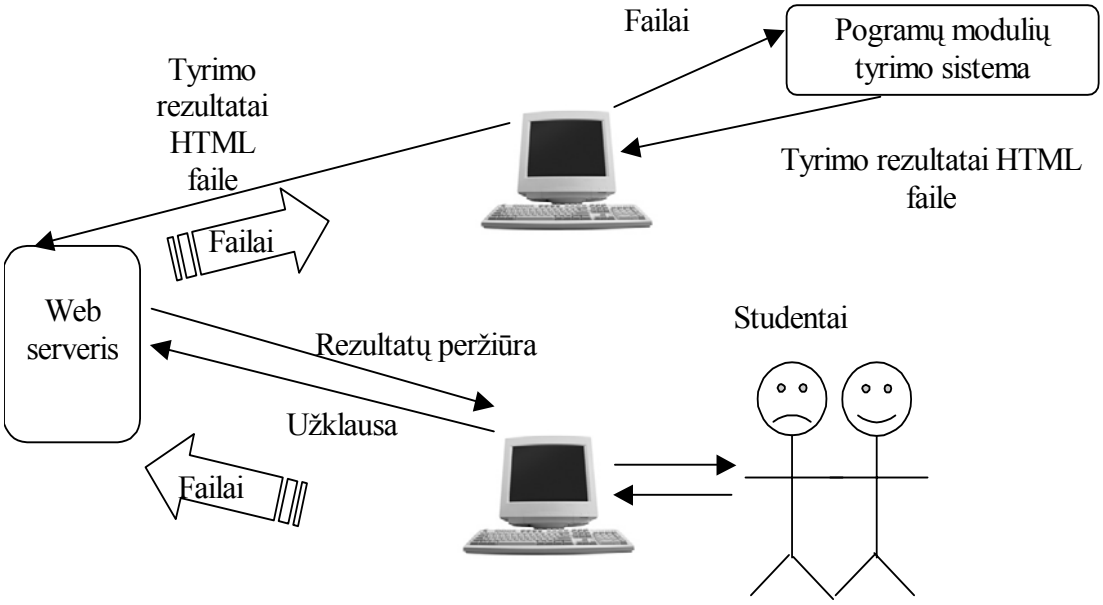
Aukščiau paminėtas studentų darbų surinkimo metodas turi tokias savybes:

- lengvai realizuojamas
- dėstytojas turi papildomo darbo surenkant ir pervadinant studentų failus
- sudėtingas pats surinkimo procesas (tikėtini variantai yra priimti darbus diskeliuose, siųsti darbus elektroniniu paštu ar patalpinti į FTP serverį)

Kad išspręsti dvi paminėtas problemas, galima panaudoti kitokį darbų surinkimo metodą – studentai turėtų siųsti savo darbus į specialiai tam tikslui sukurtą serverį. Atsiųsti failai

automatiškai pervadinami atitinkamais vardais ir perkeliami į nurodytą katalogą. Šis metodas turėtų tokias savybes:

- sunkiau realizuojamas (reikalauja daugiau laiko resursų)
- dėstytojui nereikia asmeniškai surinkinėti studentų darbų
- studentai turėtų užsiregistruoti sistemoje (registraciją turėtų patvirtinti dėstytojas) arba būti užregistruotais (šiuo atveju dėstytojas turėtų užregistruoti studentus ir pranešti jiems prisijungimo vardus bei slaptažodžius)



4 pav. Išoriniai sistemos duomenų srautai (studentų darbus atsiunčia į Web serverį)

3 SISTEMOS PROJEKTAS

Šiame skyriuje pateikiamas sistemos projekto aprašymas.

3.1 Reikalavimų projektuojamai sistemai specifikacija.

3.1.1 Užsakovo pateikti reikalavimai sistemai

- Sistema turi įvertinti programinio kodo kokybę bei nustatyti galimus plagijavimo atvejus.
- Sistema tiria JAVA programavimo kalba parašytas programas.
- Sistema turi būti lengvai modifikuojama.
- Tyrimo rezultatai pateikiami lentelių pavidalu.
- Gautų rezultatų išsaugojimas HTML formatu.
- Paprastas ir aiškus sistemos įsisavinimas ir eksploatavimas.

3.1.2 Funkciniai reikalavimai

- Sistema turi patikrinti programinio kodo kokybę bei įvertinti atskirų vartotojų programų panašumą.
- Rezultatai atvaizduojami lentelėse.

3.1.3 Nefunkciniai reikalavimai

Naudojamumas

- Aiškus ir suprantamas vartotojo interfeisas.
- Neilgas išmokimo naudotis programa laikas.

Patikimumas ir kokybė

- Nepatekti į kritines situacijas.
- Programa neturi leisti įvesti pradinių duomenų nesančių galimų įvedimų aibėje.

Mobilumas

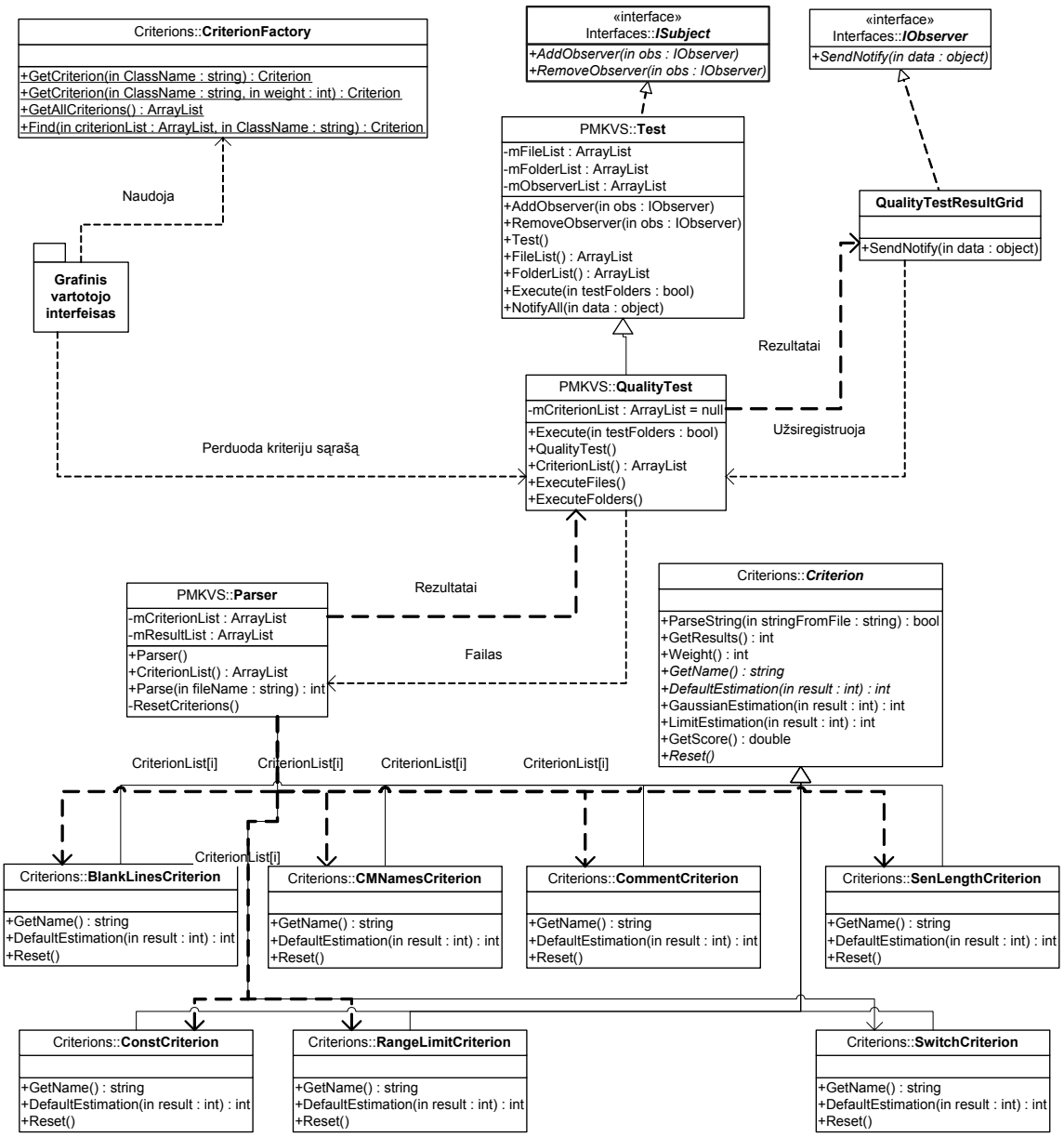
- Lengvas sistemos perkėlimas iš vieno kompiuterio į kitą.

Saugumas

- Programa neturi kenkti sistemos, kurioje vykdoma, stabilumui.

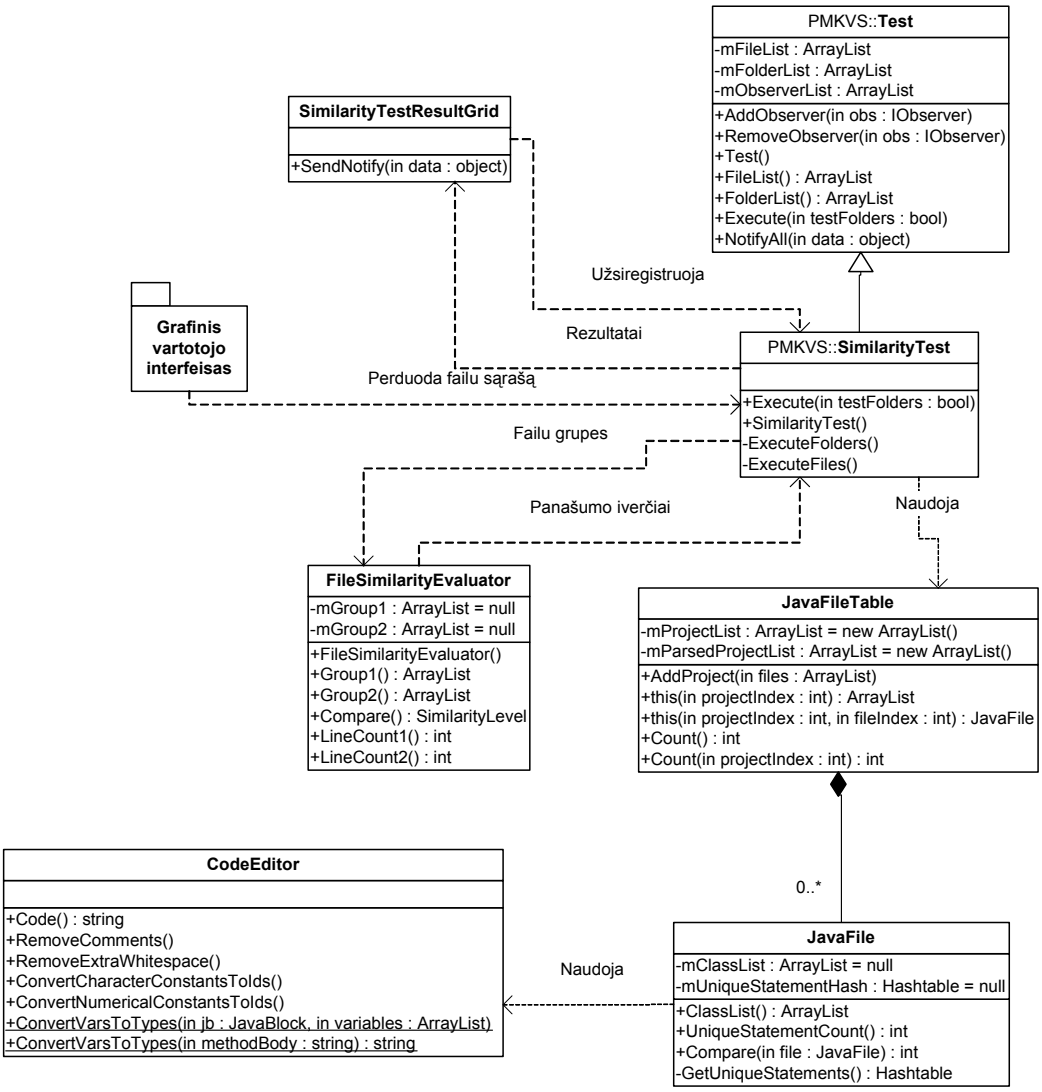
3.2 Duomenų struktūra.

3.2.1 Kokybės vertinimo modulio klasių diagrama



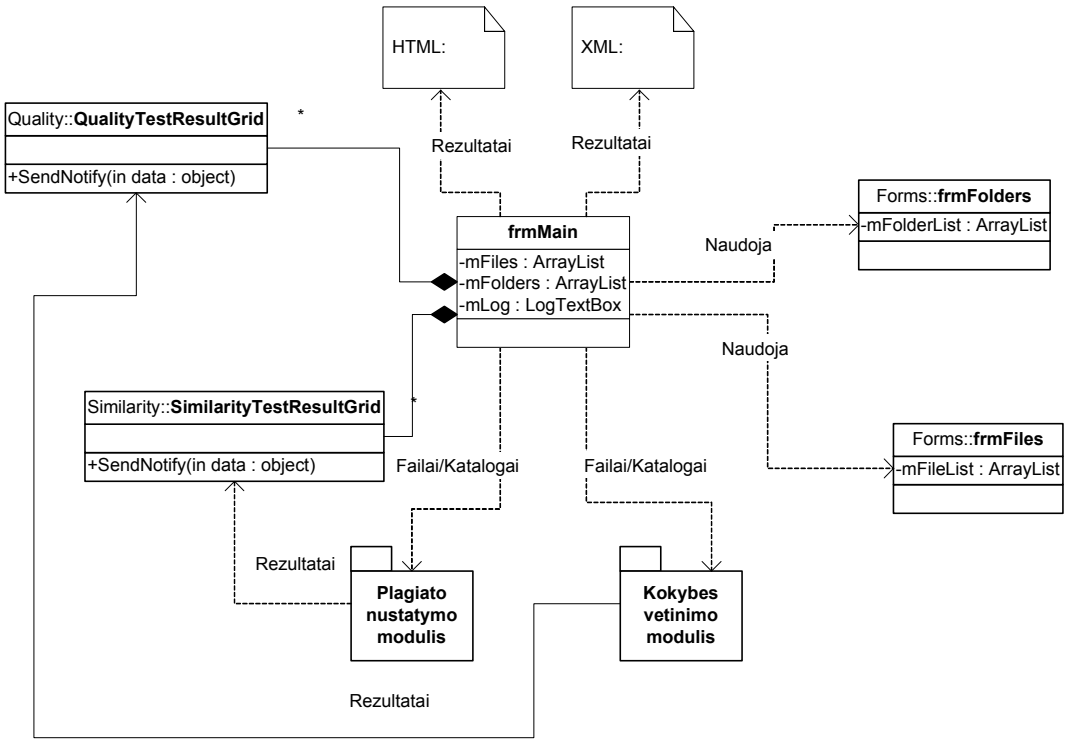
5 pav. Kokybės vertinimo modulio klasių diagrama

3.2.2 Plagiato nustatymo modulio klasių diagrama



6 pav. Plagiato nustatymo modulio klasių diagrama

3.3 Projektuojamos sistemos architektūra.



7 pav. Ryšiai tarp grafinės vartotojo sąsajos ir plagiato nustatymo bei kokybės vertinimo modulių

3.4 Plagiato nustatymo modulio realizacijos principas

Plagiato nustatymas vyksta šiais etapais:

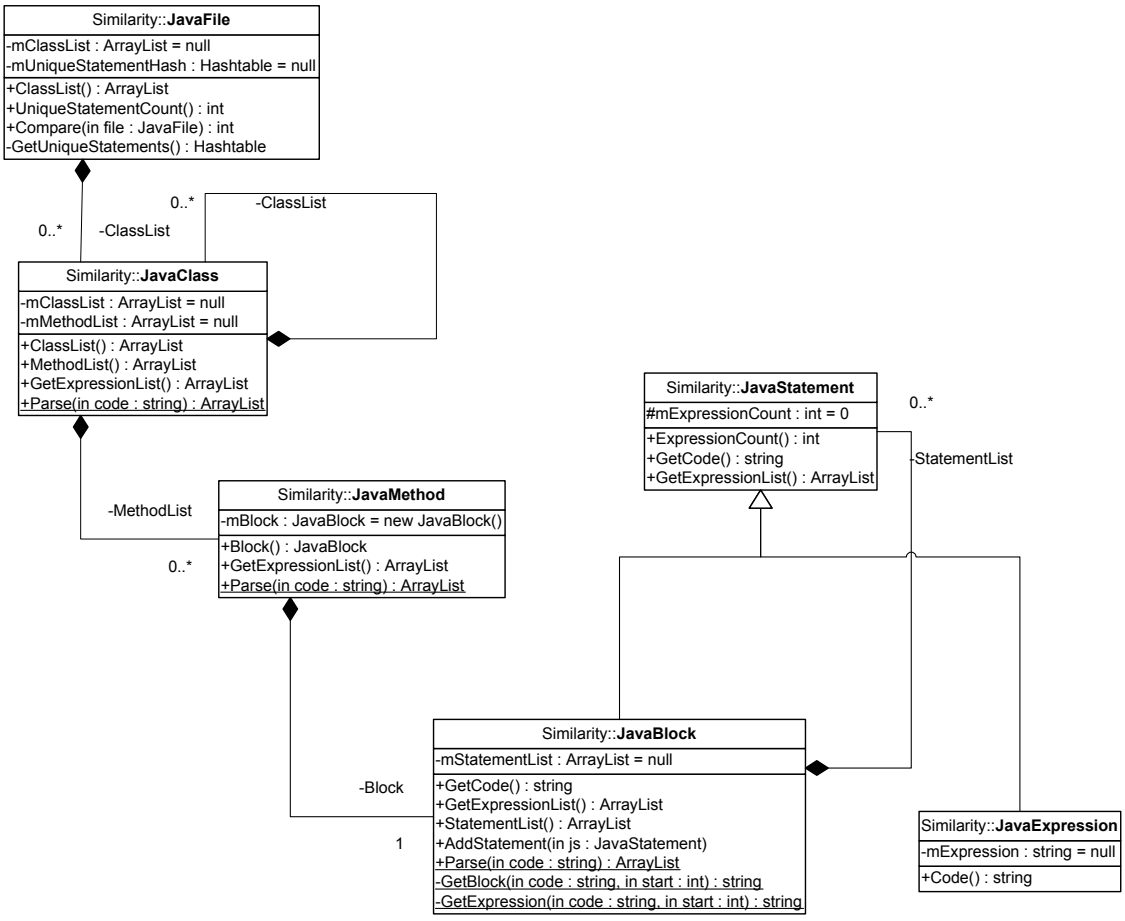
- **Programų kodo nuskaitymas iš failų.** Programos tekstas nuskaitomas iš failų į eilutės tipo kintamuosius.

- **Komentarų ir nereikalingų tarpų šalinimas.** Komentarai neturi jokios įtakos programos veikimui, todėl jie pašalinami. Programa, parašyta laikantis programavimo kultūros, paprastai turi daug tarpų, tabuliacijos ir naujų eilučių simbolių, kurie taip pat neturi jokios įtakos programos veikimui. Kai kurie tarpo simboliai yra naudojami kaip skyrikliai (pvz. sakinyje „int k=0;“ tarpas naudojamas kaip skyriklis kintamojo tipo ir kintamojo vardo atskyrimui), todėl jie paliekami.

- **Skaitinių ir simbolinių konstantų pakeitimas identifikatoriais.** Studentų laboratorinių darbų sąlygos dažnai būna panašios ar vienodos, o skiriasi pradiniai duomenys. Todėl plagijuojant dažniausiai pakeičiama skaitinės konstantos, simbolinės konstantos ir kintamųjų vardai (pastarieji keičiami identifikatoriais kitame

etape). Šiame etape pakeičiant skaitines ir simbolines konstantas identifikatoriais, pašaliname dalį neapibrėžtumo.

- **Programų gramatinis išnagrinėjimas. Programinio kodo perkėlimas į specialius objektus.** Siekiant pakeisti kintamųjų vardus jų tipais, iškyla problema – skirtingose programos vietose kintamasis, turintis tą patį vardą, gali būti nevienodo tipo. Pavyzdžiui vienoje programos dalyje kintamasis vardu *metai* gali būti apibrėžtas kaip *int* tipo kintamasis, o kitoje kaip *String* tipo kintamasis. Norint žinoti kintamųjų galiojimo sritis, programos kodas nuskaitomas į specialiai tam sukurtus objektus. Objektai sukuriami failams, klasėms, metodams, kodo sakinių blokams bei kodo sakiniams ir sudaro hierarchinę programos struktūrą (6 pav.). Failų objektai savyje gali turėti klasių objektus, pastarieji gali turėti vidinių klasių objektus ir metodų objektus. Metodas susideda iš kodo sakinių bloko. Kodo sakinių blokas gali susidėti iš kodo sakinių ir kitų kodo sakinių blokų. Kodo sakiniai yra žemiausia hierarchijos grandis.



8 pav. Hierarchinė objektų struktūra.

- **Kintamųjų vardų keitimas jų tipais.** Turint hierarchinę programos struktūrą, galima pakeisti kintamųjų vardus jų tipais. Kintamojo galiojimo sritis prasideda kintamo apibrėžimo vietoje, o baigiasi objekto, kuriame jis apibrėžtas, vidinių objektų sąrašo gale. Kitaip sakant, jei kintamasis apibrėžtas kodo sakinių bloke, tai jo galiojimo sritis yra visi šiame bloke esantys blokai ir sakiniai pradedant kintamojo apibrėžimo vieta.

- **Originalių (nevienodų) kodo eilučių paieška ir palyginimas.** Lyginant du failus tarpusavyje, kiekviename iš jų identifikuojamos originalios eilutės ir surandamas sutapusių eilučių kiekis. Failuose, kurių vienas yra plagiatas, sutapusių eilučių kiekis artėja link 100%. Šitaip lyginant failus panaikinamas dar vienas neapibrėžtumas, kuris gali atsirasti, kai studentas nuplagijuotoje programoje sukeičia vietomis sakinius, klases ar metodus. Reikia pastebėti, kad net ir nenuplagijuotuose darbuose greičiausiai bus vienodų eilučių, tačiau jų kiekis laboratorinio darbo apimties programose paprastai neviršys 10-15%.

3.5 Programinių modulių specifikacijos.

```
public class frmMain : Form
```

Pagrindinė forma.

```
public class Test : ISubject
```

Tyrimas. Apjungia pasirinktų failų tyrimą pagal pasirinktus kriterijus.

FileList	Tiriamų failų (FileItem) sąrašas.
FolderList	Tiriamų katalogų (FolderItem) sąrašas.
AddObserver	Registruoja objektą, kuriam pranešinėti apie ištirtus failus. Objektas turi realizuoti IObserver interfeisą.
Execute	Įvykdo tyrimą.
NotifyAll	Praneša visiems užsiregistravusiems objektams apie ištirtą failą. Kad žinotų koks failas buvo ištirtas, objektas turi turėti failų sąrašą ir sekti visus pranešimus.
RemoveObserver	Išregistruoja prisiregistravusį objektą. Objektas turi realizuoti IObserver interfeisą.

```
public class SimilarityTest : Test
```

Plagijavimo nustatymo tyrimo klasė.

FileList (inherited from **Test**)	Tiriamų failų (FileItem) sąrašas.
FolderList (inherited from **Test**)	Tiriamų katalogų (FolderItem) sąrašas.
AddObserver (inherited from **Test**)	Registruoja objektą, kuriam pranešinėti apie ištirtus failus. Objektas turi realizuoti IObserver interfeisą.
Execute	Įvykdo tyrimą.
NotifyAll (inherited from **Test**)	Praneša visiems užsiregistravusiems objektams apie ištirtą failą. Kad žinotų koks failas buvo ištirtas, objektas turi turėti failų sąrašą ir sekti visus pranešimus.
RemoveObserver (inherited from **Test**)	Išregistruoja prisiregistravusį objektą. Objektas turi realizuoti IObserver interfeisą.

```
public class QualityTest : Test
```

Kokybės įvertinimo klasė.

QualityTest Constructor	Inicializuoja naują klasės QualityTest objektą.
CriterionList	Kriterijų sąrašas.
FileList (inherited from **Test**)	Tiriamų failų (FileItem) sąrašas.
FolderList (inherited from **Test**)	Tiriamų katalogų (FolderItem) sąrašas.
AddObserver (inherited from **Test**)	Registruoja objektą, kuriam pranešinėti apie ištirtus failus. Objektas turi realizuoti IObserver interfeisą.
Execute	Įvykdo tyrimą.
ExecuteFiles	Įvykdo tyrimą su failais.
ExecuteFolders	Įvykdo tyrimą su katalogais.
NotifyAll (inherited from **Test**)	Praneša visiems užsiregistravusiems objektams apie ištirtą failą. Kad žinotų koks failas buvo ištirtas, objektas turi turėti failų sąrašą ir sekti visus pranešimus.
RemoveObserver (inherited from **Test**)	Išregistruoja prisiregistravusį objektą. Objektas turi realizuoti IObserver interfeisą.

```
public class Parser
```

Apjungia programos teksto eilutės tyrimą pagal visus pasirinktus kriterijus.

CriterionList	Kriterijų sąrašas.
Parse	Ištiria kodo eilutę pagal visus pasirinktus kriterijus.

public class FileItem

Elementas, nusakantis failą. Jį galima naudoti kaip sudėtinį kitų elementų (pvz. ListBox, Grid) komponentą, kuris gali rodyti pilną kelią iki failo arba tik failo pavadinimą.

Directory	Kelias iki failo (pvz. "D:\\failai").
FileName	Failo pavadinimas (pvz. "failas.java").
ShowDir	Rodyti pilną kelią ar tik failo pavadinimą.
Equals	Nustato ar duotas objektas yra toks pats kaip šis objektas.
GetHashCode	Grąžina objekto hash kodą.
GetPath	Grąžina pilną failo kelią (su failo pavadinimu).
ToString	Jei ShowDir yra **true**, grąžina pilną kelią. Priešingu atveju, grąžina failo pavadinimą. listbox objektas automatiškai iškviečia šį metodą, kad gautų elemento tekstą.

public class FolderItem

Elementas, nusakantis katalogą. Jį galima naudoti kaip sudėtinį kitų elementų (pvz. ListBox, Grid) komponentą, kuris gali rodyti pilną kelią iki katalogo arba tik katalogo pavadinimą.

FileLineCountList	Failuose esančių kodo eilučių kiekio sąrašas.
FileList	Kataloge esančių java failų sąrašas. Sąraše failai laikomi string tipo laukuose.
ShowFullPath	Nustato/grąžina ar rodyti pilną kelią ar tik katalogo pavadinimą.
GetName	Grąžina katalogo pavadinimą.
ToString	Jei ShowFullPath yra **true**, gražina pilną kelią. Priešingu atveju, grąžina katalogo pavadinimą.

public abstract class Criterion : ICriterion

Klasė aprašanti kriterijų. Kiekvienas kriterijus paveldi šią klasę.

Enabled	Nurodo ar kriterijus yra aktyvuotas. Atliekant tyrimą, naudojami tik aktyvuoti kriterijai.
Weight	Kriterijaus svoris.
DefaultEstimation	Grąžina ištirto failo įvertinimą pagal standartiškai šiam kriterijui naudojamą dėsnį.
GaussianEstimation	Tyrimo rezultatų įvertinimo pagal Gauso dėsnį metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
GetName	Grąžina kriterijaus vardą.

GetScore	Įvertina išanalizuotą kodą.
LimitEstimation	Tyrimo rezultatų įvertinimo pagal ribas metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
ParseString	Išnagrinėja eilutę.
Reset	Nustato kriterijų į pradinę padėtį (paruošia kito failo analizavimui).

public interface ICriterion

Kokybės vertinimo kriterijaus interfeisas

GetResults	Grąžina rezultatą.
ParseString	Apdoroja eilutę.

public class CriterionFactory

Kriterijų fabrikas. Klasė, apjungianti metodus kriterijų objektų gavimui.

Find	Nurodytame sąraše suranda kriterijų, kurio klasės pavadinimas atitinka pavadimą, nurodytą kintamajame ClassName.
GetAllCriterions	Sukuria visų programai žinomų kriterijų objektus ir grąžina jų sąrašą. Sąraše esantys pirmas ir antras kriterijai visada turi būti BlankLinesCriterion ir CommentCriterion.
GetCriterion	Overloaded. Grąžina nauja kriterijaus objektą pagal nurodytą kriterijaus klasės pavadinimą.

public class BlankLinesCriterion : Criterion

Tuščių eilučių kriterijus.

Weight (inherited from **Criterion**)	Kriterijaus svoris.
DefaultEstimation	Grąžina ištirto failo įvertinimą pagal standartiškai šiam kriterijui naudojamą dėsnį.
GaussianEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal Gauso dėsnį metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
GetName	Grąžina kriterijaus vardą.
GetScore (inherited from **Criterion**)	Įvertina išanalizuotą kodą.
LimitEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal ribas metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
ParseString	Išnagrinėja eilutę.

Reset	Nustato kriterijų į pradinę padėtį (paruošia naujo failo tyrimui).

```
public class CMNamesCriterion : Criterion
```

Kriterijus, kuris tiria tiria, kiek aprašytų klasių ir metodų vardų prasideda iš mažosios raidės.

Weight (inherited from **Criterion**)	Kriterijaus svoris.
DefaultEstimation	Grąžina ištirto failo įvertinimą pagal standartiškai šiam kriterijui naudojamą dėsnį.
GaussianEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal Gauso dėsnį metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
GetName	Grąžina kriterijaus vardą.
GetScore (inherited from **Criterion**)	Įvertina išanalizuotą kodą.
LimitEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal ribas metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
ParseString	Išnagrinėja eilutę.
Reset	Nustato kriterijų į pradinę padėtį (paruošia naujo failo tyrimui).

```
public class CommentCriterion : Criterion
```

Komentarų tikrinimo kriterijus.

Weight (inherited from **Criterion**)	Kriterijaus svoris.
DefaultEstimation	Grąžina ištirto failo įvertinimą pagal standartiškai šiam kriterijui naudojamą dėsnį.
GaussianEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal Gauso dėsnį metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
GetName	Grąžina kriterijaus vardą.
GetScore (inherited from **Criterion**)	Įvertina išanalizuotą kodą.
LimitEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal ribas metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
ParseString	Išnagrinėja eilutę.
Reset	Nustato kriterijų į pradinę padėtį (paruošia naujo failo tyrimui).

```
public class ConstCriterion : Criterion
```

Kriterijus nustatantis ar masyvu aprašyme naudojamos konstantos o ne skaičiai

Weight (inherited from **Criterion**)	Kriterijaus svoris.
DefaultEstimation	Grąžina ištirto failo įvertinimą pagal standartiškai šiam kriterijui naudojamą dėsnį.
GaussianEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal Gauso dėsnį metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
GetName	Grąžina kriterijaus vardą.
GetScore (inherited from **Criterion**)	Įvertina išanalizuotą kodą.
LimitEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal ribas metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
ParseString	Išnagrinėja eilutę.
Reset	Nustato kriterijų į pradinę padėtį (paruošia naujo failo tyrimui).

```
public class RangeLimitCriterion : Criterion
```

Kriterijus, tiriantis ar for cikle naudojamas intervalo aprašymo rėžis yra aprašytas teisingai.

Weight (inherited from **Criterion**)	Kriterijaus svoris.
DefaultEstimation	Grąžina ištirto failo įvertinimą pagal standartiškai šiam kriterijui naudojamą dėsnį.
GaussianEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal Gauso dėsnį metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
GetName	Grąžina kriterijaus vardą.
GetScore (inherited from **Criterion**)	Įvertina išanalizuotą kodą.
LimitEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal ribas metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
ParseString	Išnagrinėja eilutę.
Reset	Nustato kriterijų į pradinę padėtį (paruošia naujo failo tyrimui).

```
public class SenLengthCriterion : Criterion
```

Kriterijus, kuris skaičiuoja vidutinį tiriamos programos sakinių ilgį. Neskaičiuojami komentarai ir tuščios eilutės.

Weight (inherited from **Criterion**)	Kriterijaus svoris.
DefaultEstimation	Grąžina ištirto failo įvertinimą pagal standartiškai šiam kriterijui naudojamą dėsnį.

GaussianEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal Gauso dėsnį metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
GetName	Grąžina kriterijaus vardą.
GetScore (inherited from **Criterion**)	Įvertina išanalizuotą kodą.
LimitEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal ribas metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
ParseString	Išnagrinėja eilutę.
Reset	Nustato kriterijų į pradinę padėtį (paruošia naujo failo tyrimui).

```
public class SwitchCriterion : Criterion
```

Kriterijus, tiriantis kaip dažnai programoje naudojamas raktinis žodis switch.

Weight (inherited from **Criterion**)	Kriterijaus svoris.
DefaultEstimation	Grąžina ištirto failo įvertinimą pagal standartiškai šiam kriterijui naudojamą dėsnį.
GaussianEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal Gauso dėsnį metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
GetName	Grąžina kriterijaus vardą.
GetScore (inherited from **Criterion**)	Įvertina išanalizuotą kodą.
LimitEstimation (inherited from **Criterion**)	Tyrimo rezultatų įvertinimo pagal ribas metodas. Tikrinama, į kurį vertinimo skalės intervalą patenka tyrimo rezultatas result.
ParseString	Išnagrinėja eilutę.
Reset	Nustato kriterijų į pradinę padėtį (paruošia naujo failo tyrimui).

```
public class frmFiles : Form
```

Forma, leidžianti pasirinkti failus tyrimams.

GetFiles	Grąžina pasirinktų failų sąrašą.

```
public class frmFolders : Form
```

Forma, leidžianti pasirinkti katalogus tyrimams.

GetFolders	Grąžina pasirinktų katalogų sąrašą.

```
public class frmQualityTestSettings : Form
```

Kokybės tyrimo nustatymų forma.

CriterionList	Kriterijų sąrašas.
AddRange	Papildyti kriterijų sąrašą.

```
public interface IObserver
```

Stebėtojas. Stebi objektą (arba keletą objektų), kuris(ie) realizuoja ISubject interfeisą. Pastarasis(ieji) praneša stebėtojui apie tam tikrus įvykius iškviesdamas(i) stebėtojo viešą metodą SendNotify.

SendNotify	Gauna pranešimą iš stebimo objekto ir įvykdo atitinkamus veiksmus.

```
public interface ISubject
```

Klasė, kurią galima 'stebėti'. Apie įvykius stebėtojams pranešama per metodą SendNotify.

AddObserver	Papildo stebėtojų sąrašą.
RemoveObserver	Pašalina stebėtoją iš stebėtojų sąrašo.

```
public class Logger : ISubject
```

Pranešimų registravimo klasė. Objektai, norintys gauti pranešimus turi užsiregistruoti (AddObserver). Registruojantis reikia nurodyti kokio tipo pranešimus pageidaujama gauti.

AddObserver	Papildo stebėtojų sąrašą. Pranešimų tipas nustatomas MessageType.Main
Dump	Išsiunčiamas pranešimas stebėtojams, pageidaujantiems gauti type tipo pranešimus.
RemoveObserver	Pašalina stebėtoją iš stebėtojų sąrašo.

```
public class LogTextBox : RichTextBox, IObserver
```

Vartotojo interfeiso elementas, išvedantis teksto pavidalu registravimo pranešimus.

SendNotify	Metodas, per kurį stebiamieji objektai praneša apie įvykius.

```
public class LogWriter : IObserver
```

Stebėtojas, išvedantis gautus pranešimus į failą.

DumpToFile	Išveda teksto eilutę į failą.
SendNotify	Gauna pranešimą iš stebimo objekto ir išveda jo duomenis į failą.

```
public enum Logger.MessageType
```

Pranešimo tipas.

Kintamojo pavadinimas	Apibūdinimas
Main	Svarbiausi pranešimai.
File	Pranešimai, kurie išvedinėjami į failą.
Debug	Pranešimai programos darbo stebėjimui programos kūrimo metu.
UserDefined	Vartotojo nustatyti pranešimai.

```
public class QualityTestResultGrid : Grid, IObserver
```

Vartotojo interfeiso elementas, surenkantis ir atvaizduojantis kokybės testo rezultatus lentelės pavidalu.

CriterionList	Kriterijų sąrašas. Pagal jį nustatomi lentelės stulpelių pavadinimai. Tas pats kriterijų sąrašas turi būti paduodamas Test objektui.
FileList	Tiriamų failų sąrašas.
SendNotify	Apskaiciuoja ištirto failo kriterijų ivertinimus, bei galutinį įvertinimą. Šis metodas yra iškviečiamas iš Test objekto, kiekvieną kartą ištyrus failą. Kriterijų įvertinimai apskaičiuojami atsižvelgus į kriterijaus svorio santykį su visų pasirinktų kriterijų svorių suma. Kitaip sakant jei kriterijaus svoris yra 20, o visu pasirinktų kriterijų svorių suma yra 100, tai gautas rezultatas bus tarp 0 ir 2.0.

```
public class CriterionSettings
```

Klasė kriterijaus nustatymų saugojimui ir atstatymui.

ClassName	Kriterijaus klasės pavadinimas.
Enabled	Ar kriterijus aktyvuotas.
Weight	Kriterijaus svoris.
Update	Atnaujina kriterijaus reikšmes.

```
public class ResultSettings
```

Klasė tyrimo rezultatų saugojimui.

Projects	Katalogų (projektų) masyvas.
SimilarityLevels	Panašumo lygių masyvas.

```
public class Settings
```

Klasė programos nustatymų saugojimui.

Criterions	Kriterijų sąrašas.
LoadDefaultSettings	Nustato standartines visų nustatymų reikšmes.
settings	Išsaugomų kintamųjų "hešas".

public class SettingsManager

Klasė laikanti ir išsauganti į diską programos nustatymus XML formatu.

LoadSettingsFromFile	Nuskaito anksčiau faile išsaugoto klasės objekto duomenis. Jei nurodytas failas neegzistuoja arba jo nuskaitymo metu įvyksta klaida, grąžinami standartinės reikšmės.
SaveSettingsToFile	Išsaugo programos nustatymus į nurodytą failą.
SerializeToStream	Nurodyto objekto viešus parametrus išsaugo XML pavidalu į nurodytą srautą.
Settings	Grąžina nustatymų klasės objektą.
SettingsFile	Grąžina arba nustato programos nustatymų failo vardą.
LoadSettings	Nuskaito programos nustatymus iš failo.
RevertChanges	Panaikina pakeitimus atsiradusius nustatymų klasės objekte po paskutinio nustatymų nuskaitymo ar išsaugojimo.
SaveSettings	Išsaugo programos nustatymus faile.

public class CodeEditor

Atlieka pakeitimus programiniame kode.

ConvertVarsToTypes	Programiniame kode rastus kintamųjų vardus pakeičia tų kintamųjų tipais (pvz. "int j; j = 0; j++;" pakeičiamas į "int int; int = 0; int++;")
CodeEditor Constructor	Inicializuoja naują klasės CodeEditor objektą.
Code	Programinis kodas.
ConvertCharacterConstantsToIds	Programiniame kode rastas simbolių konstantas (pvz. "tekstas", 'A') pakeičia identifikatoriais.
ConvertNumericalConstantsToIds	Programiniame kode rastas skaičių konstantas (pvz. 52, 7.0) pakeičia identifikatoriais.
RemoveComments	Pašalina programiniame kode rastus komentarus.
RemoveExtraWhitespace	Pašalina programiniame kode programos sintaksei neturinčius įtakos tarpus bei eilučių pabaigas žyminčius simbolius (\r\n).

public class FileSimilarityEvaluator

Klasė, nustatanti dviejų java failų kodo panašumo lygį.

Group1	Java failų (JavaFile) grupė (pvz. vienas studento laboratorinis darbas susidedantis iš keleto java failų).
Group2	Java failų (JavaFile) grupė (pvz. vienas studento laboratorinis darbas susidedantis iš keleto java failų).
Compare	Palygina failų grupes tarpusavyje ir nustato jų panašumo lygį procentais.
LineCount1	Grąžina pirmame faile esančių kodo eilučių kiekį.
LineCount2	Grąžina antrame faile esančių kodo eilučių kiekį.

public class JavaBlock : JavaStatement

Kodo sakinių blokas.

Parse	Išnagrinėja duotą kodą ir grąžina formuluočių (blokų ir sakinių) sąrašą.
ExpressionCount (inherited from **JavaStatement**)	Programinio kodo sakinių kiekis formuluotėje.
StatementList	Formuluočių (JavaStatement) sąrašas
AddStatement	Papildyti bloką nurodyta formuluote.
GetCode	Grąžinti bloko programinį kodą.
GetExpressionList	Grąžina bloke esančių sakinių sąrašą.
mExpressionCount (inherited from **JavaStatement**)	Programinio kodo sakinių kiekis formuluotėje.

public class JavaClass

Nusako java klasės struktūrą.

Parse	Išnagrinėja pateiktą kodą, suranda jame klases, kurias sudeda į JavaClass objektus ir grąžina jų sąrašą.
ClassList	Klasėje rastų vidinių klasių sąrašas.
MethodList	Klasėje rastų metodų sąrašas.
GetExpressionList	Grąžina klasėje rastų kodo sakinių sąrašą.

public class JavaExpression : JavaStatement

Programinio kodo sakinys.

Code	Programinio kodo sakinys.
ExpressionCount (inherited from **JavaStatement**)	Programinio kodo sakinių kiekis formuluotėje.
GetCode	Grąžina sakinio programinį kodą.
GetExpressionList	Grąžina sąrašą su šiuo programinio kodo sakiniu.
mExpressionCount (inherited from **JavaStatement**)	Programinio kodo sakinių kiekis formuluotėje.

public class JavaFile

Nusako java failo struktūrą.

ClassList	Faile rastų klasių sąrašas.
Compare	Palygina šį JavaFile objektą su nurodytu parametruose ir grąžina failuose sutapusių kodo sakinių skaičių.
UniqueStatementCount	Apskaičiuoja unikalių (nevienodų) faile esančių kodo sakinių kiekis. Kadangi plagiato nustatymo testo metu pašalinami kodo neapibrėžtumai (konstantos, kintamųjų vardai ir t.t.), todėl daug prieš tai buvusių nevienodų sakinių šiame metode jau bus skaičiuojami kaip vienodi.

public class JavaFileTable

Saugo java failų nuorodas (kelius) ir pagal išnagrinėtą jų tekstą sukurtus JavaFile klasių objektus. Failai sugrupuoti į projektus. JavaFile objektus galima pasiekti nurodant projekto ir failo indeksus. Pavyzdys:

```
JavaFileTable table = new JavaFileTable();

table.AddProject({"file1.java", "file2.java"});

table.AddProject({"file3.java", "file4.java", "file5.java"});

// gauti išparsintą file5.java

JavaFile jf = table[1, 2];

// gauti išparsintą file1.java

JavaFile jf2 = table[0, 0];
```

Item	Grąžina projekto nurodytu indeksu išnagrinėtus failus (JavaFile).
AddProject	Papildo projektų (katalogų) sąrašą nauju projektu (katalogu).
Count	Grąžina projektų kiekį.

public class JavaMethod

Nusako java klasės metodo struktūrą.

Parse	Išnagrinėja duotą kodą, išrenka jame rastus metodus, sudeda į JavaMethod objektus ir grąžina.
Block	Programinio kodo blokas.
GetExpressionList	Grąžina metode rastų kodo sakinių sąrašą.

public class JavaStatement

Formuluotė (programinio kodo blokas arba sakinys).

JavaStatement Constructor	Inicializuoja naują klasės JavaStatement objektą.
ExpressionCount	Programinio kodo sakinių kiekis formuluotėje.
GetCode	Grąžina formuluotės programinį kodą.
GetExpressionList	Grąžina programinio kodo sakinių sąrašą formuluotėje.

public class SimilarityLevel : IComparable

Rodo katalogų panašumo lygį.

SimilarityLevel Constructor	Inicializuoja naują klasės SimilarityLevel objektą.
Count1	Java failų kiekis pirmame kataloge.
Count2	Java failų kiekis antrame kataloge.
LineCountArray1	Eilučių kiekių pirmo katalogo failuose masyvas.
LineCountArray2	Eilučių kiekių antro katalogo failuose masyvas.
MatchedLineCountTable	Sutapusių eilučių kiekių katalogų failuose lentelė.
CompareTo	Palygina šį objektą su duotuoju. Šis metodas reikalingas SimilarityLevel tipo objektų rūšiavimui.
GetMirror	Grąžina šio objekto veidrodinę kopiją. Veidrodinė kopija rodo antro katalogo panašumo su pirmu katalogu lygį. Palyginus pirmą katalogą su antru, tereikia iškviesti šį metodą, kad gautume antro katalogo panašumo lygį su pirmuoju.
ToString	Grąžina katalogų panašumo lygį procentais eilutės pavidalu. Lygis apskaičiuojamas, jei anksčiau nebuvo apskaičiuotas.
mLevel	Katalogų panašumas procentais.

public class SimilarityTestResultGrid : UserControl, IObserver

Komponentas, plagiavimo tyrimo rezultatus atvaizduojantis lentelės pavidalu.

FileList	Failų (FileItem) sąrašas.
FolderList	Katalogų (FolderItem) sąrašas.
SendNotify	Priima panašumo įverčius (SimilarityLevel) ir atvaizduoja juos lentelėje.

4 EKSPERIMENTINIS TYRIMAS

Sistemos plagiato nustatymo modulis išbandytas programavimo disciplinos „Objektinė technologija ir duomenų struktūros“ (kodas P175B312) egzamino metu. Keletas studentų patalpino ginamus laboratorinius darbus nurodytame FTP serveryje. Vėliau atlikto plagiato nustatymo tyrimo metu, rasti 3 nuplagijuoti darbai.

Iš 1 lentelės matome, kad studentų Breitmozero ir Minsevičiaus darbų panašumas viršija 93%. Čia rezultatas 93.02 reiškia, kad 93.02% studento Breitmozero programoje rastų originalių kodo eilučių taip pat rasta ir studento Minsevičiaus programoje. Analogiškai rezultatas 93.24 reiškia, kad 93.24% studento Minsevičiaus programoje rastų originalių kodo eilučių taip pat rasta ir studento Breitmozero programoje. Kuris darbas originalus, o kuris plagiatas deja paprastai iš programų teksto nustatyti neįmanoma.

Taip pat 1 lentelėje matome, kad studento Nekrosevičiaus darbo panašumas su kitais darbais pakankamai aukštas (37.64%, 38.2%, 24.16%). Atlikus papildomą testą, kuriame palyginome studentų Nekrosevičiaus ir Minsevičiaus laboratorinių darbų failus, paaiškėjo, kad didžiausias panašumas yra failų *Kolekcijos.java* ir *Lab1Class.java* (2 lentelė). Peržiūrėjus šiuos failus pastebėta, kad tai vartotojo interfeiso dalis, kurią automatiškai sugeneruoja JAVA programavimo aplinka JBuilder.

Iš 2 lentelės dar galime pamatyti, kad programa palygino ir to pačio laboratorinio darbo failus tarpusavyje. Taip atsitiko todėl, kad dirbant failų režimu, laikoma, kad kiekvienas failas yra atskira programa. Dirbant katalogų režimu, sistema nelygina viename kataloge esančių *.java failų tarpusavyje, o tik su kitų katalogų *.java failais. Tačiau katalogų režime rodomas bendras rezultatas, kuris ne visada yra pakankamai informatyvus.

1 lentelė. Pirmojo laboratorinio darbo plagiato nustatymo tyrimo rezultatai:

Plagiato nustatymo tyrimo rezultatai(pirmas laboratorinis darbas)

Katalogas1\Katalogas2	BreitmozerasIF2_10	MinseviciusIF2_10	NekroseviciusIF2_2	StratkauskasIF2_3	AKamantauskas
BreitmozerasIF2_10		93.02	15.81	3.72	15.35
MinseviciusIF2_10	93.24		15.62	3.73	14.69
NekroseviciusIF2_2	37.64	38.2		1.12	24.16
StratkauskasIF2_3	7.55	7.55	1.42		4.25
AKamantauskas	26.34	24.73	16.13	3.23	

2 lentelė. Studentų Nekrosevičiaus ir Minsevičiaus pirmojo laboratorinio darbo failų plagiato nustatymo tyrimo rezultatai (studento Nekrosevičiaus laboratorinio darbo failai paryškinti):

Plagiato nustatymo tyrimo rezultatai (studentų Nekrosevičiaus ir Minsevičiaus laboratorinių darbų failai)

Katalogas1\Katalogas2	**Kolekcijos.java**	**Dialogas.java**	**Frame1.java**	Elementas.java	Lab1Class.java	Lab1Frame.java	Lab1Frame_AboutBox.java	Sarasas.java	Lab1Sarasas.java
Kolekcijos.java		12.5	12.5	0	87.5	12.5	12.5	0	0
Dialogas.java	3.7		14.81	0	3.7	29.63	25.93	1.85	0
Frame1.java	1.85	7.41		0	1.85	34.26	6.48	2.78	0
Elementas.java	0	0	0		0	0	0	11.11	0
Lab1Class.java	87.5	12.5	12.5	0		12.5	12.5	0	0
Lab1Frame.java	0.69	5.54	12.8	0	0.69		2.42	2.08	0
Lab1Frame_AboutBox.java	5.88	41.18	20.59	0	5.88	20.59		0	0
Sarasas.java	0	1.37	4.11	2.74	0	8.22	0		0
Lab1Sarasas.java	0	0	0	0	0	0	0	0	

3 lentelėje matome, kad dviejų studentų darbai yra nuplagijuoti nuo kito studento.

3 lentelė. Antrojo laboratorinio darbo plagiato nustatymo tyrimo rezultatai:

Plagiato nustatymo tyrimo rezultatai (antras laboratorinis darbas)

Katalogas1\Katalogas2	ALikasIF2_14	MGestautasIF2_2	NekroseviciusIF2_2	TaunysIF2_1	TimofejevasIF2_10	ADervinskas
ALikasIF2_14		8. 56	8. 56	6. 31	7. 21	8. 56
MGestautasIF2_2	3. 83		98. 09	26. 56	2. 87	85. 89
NekroseviciusIF2_2	3. 83	98. 09		26. 56	2. 87	85. 89
TaunysIF2_1	4. 89	21. 26	21. 26		2. 87	21. 26
TimofejevasIF2_10	6. 9	6. 9	6. 9	5. 75		6. 9
ADervinskas	3. 7	83. 1	83. 1	25. 69	2. 78	

4 lentelėje matome, kad nuplagijuotų darbų nėra. Čia studentų Kamantausko ir Jackaus darbų panašumo pakankamai aukštas procentas atsirado vėlgi dėl JBuilder automatiškai sugeneruojamų vartotojo grafinio interfeiso failų.

4 lentelė. Trečiojo laboratorinio darbo plagiato nustatymo tyrimo rezultatai:

Plagiato nustatymo tyrimo rezultatai (trečias laboratorinis darbas)

Katalogas1\Katalogas2	BucinskasIF215	Jackus	Sinkevicius	AKamantauskas
BucinskasIF215		7.77	0.52	2.07
Jackus	5.13		0.96	27.24
Sinkevicius	2.04	2.04		4.08
AKamantauskas	2.16	39.22	0.86	

Sistemos kokybės vertinimo modulį, atlikdamas individualią užduotį, tyrė studentas J.Prelgauskas. Jis savo darbe turėjo teoriškai apskaičiuoti kokius įvertinimus turės grąžinti programa pagal nurodytus kriterijus ir palyginti teorinius bei praktinius rezultatus. Lentelėse 5, 6 ir 7 matome trijų kriterijų palyginimo rezultatus. Dviejuose testuose teoriniai rezultatai su praktiniais sutapo 92% tikslumu, trečiame – 100%. Dėl teorinių ir praktinių rezultatų neatitikimo patikrintas programos kodas ir pataisyta rasta klaida.

Pagal kitus kriterijus visi failai teoriškai ir praktiškai buvo įvertinti 10 (juose nerasta neteisingai panaudotų kodo fragmentų). Todėl studentas konstatavo, kad šie kriterijai neinformatyvūs.

5 lentelė. Kokybės įvertinimo pagal tuščių eilučių kriterijų teorinių ir praktinių rezultatų palyginimas:

Failo pav.	Programos įvertinta	Teoriškai apskaičiuota
10MantoKlase.java	5	8.3% = 5
1MantoKlase.java	3	5.4% = 3
2MantoKlase.java	9	17.2% = 9
3MantoKlase.java	3	5.3% = 3
4MantoKlase.java	3	5.3% = 3
5MantoKlase.java	3	5.1% = 3
6MantoKlase.java	4	6.8 = 4
7MantoKlase.java	4	6.6% = 4
8MantoKlase.java	5	8.3% = 5
9MantoKlase.java	2	3.4% = 2
AvangardoKlase.java	2	3.6% = 2
KostoElementas.java	0	0% = 0
KostoFailai.java	1	2.2% = 2
KostoFreimas.java	0	1.5% = 1
KostoKlase.java	3	5.9% = 3
KostoKolekcijos.java	0	0.5% = 0
KostoLab4.java	3	4.6% = 3
KostoMasyvas.java	2	3.1% = 2
KostoMeniu.java	0	0% = 0
KostoSarasas.java	0	0% = 0
VaidosElementas.java	10	23% = 10
VaidosKlase.java	3	5.9% = 3
VaidosMeniu.java	3	5.6% = 3
VaidosParduotuve.java	5	8.6% = 5
VaidosSarasas.java	4	6.7% = 4

6 lentelė. Kokybės įvertinimo pagal kriterijų „Vidutinis sakinių ilgis“ teorinių ir praktinių rezultatų palyginimas:

Failo pav.	Programos įvertinta	Teoriškai apskaičiuota
10MantoKlase.java	0	0
1MantoKlase.java	0	0
2MantoKlase.java	0	0
3MantoKlase.java	0	0
4MantoKlase.java	0	0
5MantoKlase.java	7	7
6MantoKlase.java	7	7
7MantoKlase.java	5	5
8MantoKlase.java	0	0
9MantoKlase.java	10	10
AvangardoKlase.java	10	10
KostoElementas.java	9	9
KostoFailai.java	9	9
KostoFreimas.java	8	8
KostoKlase.java	10	10
KostoKolekcijos.java	8	8
KostoLab4.java	10	10
KostoMasyvas.java	10	10
KostoMeniu.java	10	10
KostoSarasas.java	10	10
VaidosElementas.java	8	8
VaidosKlase.java	10	10
VaidosMeniu.java	10	10
VaidosParduotuve.java	9	9
VaidosSarasas.java	10	10

7 lentelė. Kokybės įvertinimo pagal kriterijų „Komentarų kiekis“ teorinių ir praktinių rezultatų palyginimas:

Failo pav.	Programos įvertinta	Teoriškai apskaičiuota
10MantoKlase.java	9	35.6% = 9
1MantoKlase.java	8	65.5% = 8
2MantoKlase.java	8	66.7% = 8
3MantoKlase.java	10	53.8% = 10
4MantoKlase.java	10	53.2% = 10
5MantoKlase.java	3	11.7% = 3
6MantoKlase.java	3	13.3% = 3
7MantoKlase.java	6	23.8% = 6
8MantoKlase.java	10	41.0% = 10
9MantoKlase.java	4	14.2% = 4
AvangardoKlase.java	1	4.0% = 1
KostoElementas.java	7	27.9% = 7
KostoFailai.java	3	11.9% = 3
KostoFreimas.java	1	5.6% = 1
KostoKlase.java	2	9.2% = 2
KostoKolekcijos.java	1	8.3% = 2
KostoLab4.java	5	19.4% = 5
KostoMasyvas.java	1	3.2% = 1
KostoMeniu.java	1	6.1% = 1
KostoSarasas.java	5	17.8% = 5
VaidosElementas.java	2	7.7% = 1
VaidosKlase.java	2	9.2% = 2
VaidosMeniu.java	1	3.9% = 1
VaidosParduotuve.java	1	7.6% = 1
VaidosSarasas.java	3	13% = 3

Apibendrinant reikia pastebėti, kad nors tyrimas parodė, kad teoriniai ir praktiniai rezultatai daugeliu atvejų sutapo, tačiau tiriamas programas įvertinti teisingai pagal kai kuriuos kriterijus sistemos pagalba kartais tiesiog neįmanoma. Pavyzdžiui tuščių eilučių kriterijus turėtų įvertinti ir komentarus, kurie naudojami vietoj tuščių eilučių, tačiau sukurti algoritmą, kuris nustatytų ar komentaras naudojamas kaip tuščia eilutė yra labai sudėtinga.

5 IŠVADOS

1. Pagal pateiktus studentų žinių analizės ir vertinimo kriterijus pasirinkta programų kokybės vertinimo kompiuterizavimo sistemos struktūra ir principai.
2. Kadangi sistemą ketinama naudoti daug metų, pagrindinį dėmesį nuspręsta sutelkti į galimybes lengvai modifikuoti, bei išplėsti sistemą.
3. Projektas pristatytas tarptautinėje konferencijoje „Pažangios informacinės technologijos ir jų taikymas studijų procese“, bei publikuotas konferencijos metu platintame leidinyje.
4. Realizuojant sistemą, atsižvelgta į studentų, testavusių prototipinę sistemą, pastebėtus trūkumus.
5. PMKVS sistemą numatoma įtraukti į kompleksinę studentų žinių kontrolės sistemą (TestTool), kuri jau keletą metų taikoma programavimo disciplinų moduliuose „Duomenų struktūros ir algoritmų analizė“ (kodas P175B353) ir „Intelektualiųjų informacinių sistemų kursinis projektas“ (kodas P175S336). Pats principas gali būti pritaikytas ir kitos srities praktinių darbų kokybės vertinimui, jei yra aiškiai apibrėžti darbų vertinimo ir jų palyginimo kriterijai. Sistema gali būti panaudota ir mokyklose, informatikos kurse, mokant mokinius programavimo pagrindų.

6 LITERATŪRA

1. Meade, J. Cheating: Is Academic Dishonesty Par for the Course? ASEE Prism, 1992, 30-32p.
2. Ambler, S. Writing Robust Java Code [interaktyvus]. 2000. Prieiga per internetą: http://www.ambysoft.com/javaCodingStandards.html.
3. Programinių modulių kokybės vertinimo sistema. Bakalaurinis darbas. / D. Grigas, A. Kaušinis, A. Petrikas; KTU. Kaunas, 2001.
4. N. E. Fenton: Software Metrics. A rigorous approach, 1992 Chapman & Hall, London.
5. B. Eckel: Thinking in JAVA, 2000 MindView, USA.
6. J.Gosling: The Java Language Specification, Addison-Wesley, 1996. Prieiga per internetą: http://java.sun.com/docs/books/jls/
7. M. Lelevičius –"Sintaksinė ir semantinė tekstų analizė mokymosi sistemose". Tiriamasis darbas, KTU 2001, Kaunas;
8. Ceilidh documentation on the World Wide Web [interaktyvus]. Prieiga per internetą: http://www.cs.nott.ac.uk/~ceilidh/papers.html
9. CourseMaker Documentation [interaktyvus]. Prieiga per internetą: http://www.cs.nott.ac.uk/CourseMarker/cm_com/html/More_Info.html
10. XML Schema, World Wide Web Consortium [interaktyvus]. 2001. Prieiga per internetą: http://www.w3.org/XML/Schema
11. XSLT Tutorial [interaktyvus]. Prieiga per internetą: http://www.w3schools.com/xsl/default.asp
12. P. Clough, Plagiarism in natural and programming languages: an overview of current tools and technologies [interaktyvus]. Department of Computer Science, University of Sheffield, UK, 2000. Prieiga per internetą: http://www.dcs.shef.ac.uk/~cloughie/plagiarism/HTML_Version/
13. J. Prelgauskas: Duomenų struktūrų individualusis darbas. KTU. Kaunas, 2003.
14. J .Friedl: Mastering Regular Expressions. O'Reilly, 1997.
15. M. J. Wise: Detection of Similarities in Student Programs: YAP'ing may be preferable to Plague'ing. SIGCSE Technical Symposium, Kansas City, 1992. 268-271p.
16. M. J. Wise: YAP3: Improved detection of similarities in computer program and other text. 27th SIGCSE Technical Symposium, PA, 1996. 130-134p.
17. Noh, Seo-Young and Gadia, Shashi: An XML Plagiarism Detection Model for Procedural Programming Languages. Technical Report 14-03, Computer Science, Iowa State University. 2003.

7 TERMINŲ IR SANTRAUKŲ ŽODYNAS

- **API** (Application Programming Interface) – realizuotų operacinės sistemos priemonių (mygtukų, dialogų, lentelių, darbo su procesais ir duomenų srautais priemonių) rinkinys, skirtas tiesioginiam naudojimui rašomose programose.
- **CASE** (Computer Aided Software Engineering) – kompiuterinės programinės priemonės, skirtos palengvinti projektavimą.
- **Ceilidh** – komercinė programų kokybės vertinimo sistema veikianti kai kuriuose Didžiosios Britanijos universitetuose.
- **HTML** (Hyper Text Markup Language) – programavimo kalba skirta kurti internetines svetaines.
- **IDE** (Integrated Development Environment) – Integruota projektavimo aplinka. Paprastai būna specializuota tam tikrai kalbai. Apima programų teksto redaktorių, konkrečios kalbos kompiliatorių, programų derinimo priemones ir pagalbos sistemą. Gali turėti specialias priemones vartotojo aplinkos projektavimui.
- **Kriterijaus svoris** – kokią balo dalį galutiniame įvertinime sudaro kriterijaus įvertinimas.
- **Plagijavimas** – svetimos programos autorystės pasisavinimas pakeitus dalį (arba nekeitus visai) programinio kodo.
- **PMKVS** (Programinių modulių kokybės vertinimo sistema)
- **Tuščia eilutė** – eilutė kuri susideda tik iš tarpo, tabuliacijos ar eilutės pabaigos simbolių.
- **UML** (Unified Modelling Language) – priemonė skirta modeliuoti sistemą aukštesniame nei programavimas lygmenyje.
- **Žymė –** elementarus, gramatiškai nedalus kalbos vienetas pvz. kintamasis ar operatorius.

8 SANTRAUKA ANGLŲ KALBA

Software Quality Assessment System

Summary

Modern educational institutions use static methods e.g. tests in computer-based knowledge assessment systems. Tests have pattern questions with limited number of answers. Use of such methods does not guarantee objective knowledge assessment – numbers of the right answers could be guessed or memorized in advance.

Practical skills can be tested by assigning programming tasks. In order to complete practical tasks students need good knowledge in the fields examined. This leads to better assessment of student knowledge in theory and practice. Teachers get better feedback. However practical tasks need considerable amount of time to be examined in detail which most teachers do not have. When assignments are tested superficially students tend to plagiarize and so the problem gets even deeper. A survey by Donald McCabe of approximately 6000 students revealed that 74% of engineering students reported engaging in some form of academic dishonesty.

The goal of this work was to build a computer-based system to assess software quality and detect plagiarism. The system was designed and implemented. Experimental tests have shown good results. The system is intended to be used in future courses and gradually improved by other students.

9 PRIEDAI

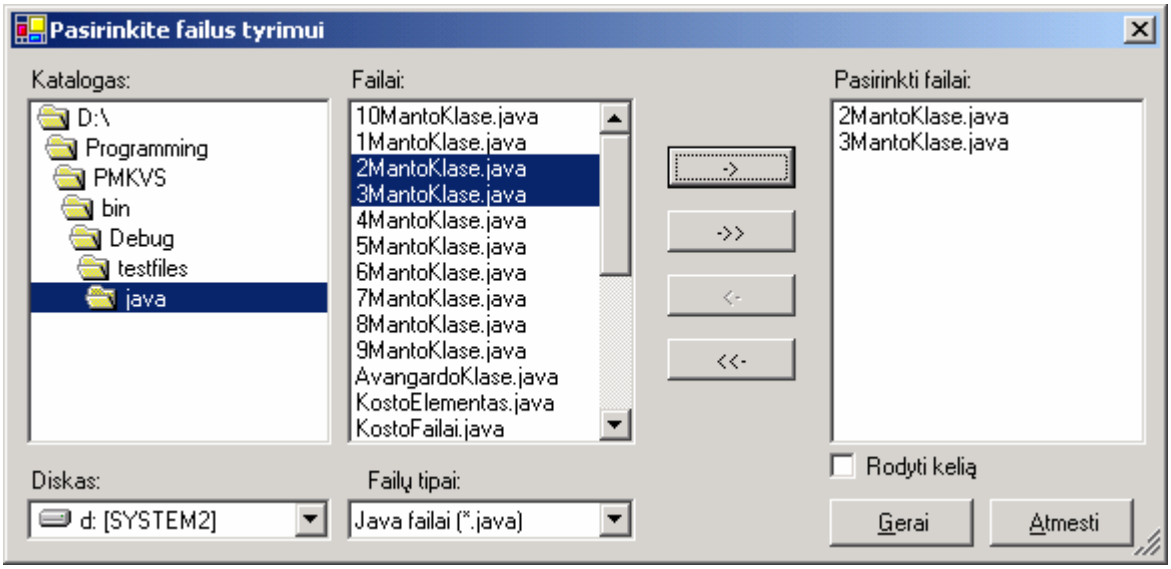
9.1 Sistemos vartotojo vadovas

Sistemos funkcinis aprašymas

„Programinių modulių kokybės vertinimo sistema“ (PMKVS) skirta studentų programavimo laboratorinių darbų, parašytų JAVA programavimo kalba, tyrimui. Sistema gali įvertinti šių darbų kokybę pagal pasirinktus kriterijus. Be to sistema turi galimybę tarp kelių studentų laboratorinių darbų nustatyti nuplagijuotus (tuos, kurių tarpusavio panašumas labai didelis). PMKVS gali tirti pasirinktus failus arba katalogus. Tiriant pasirinktus katalogus, programa ištirs visus kataloguose rastus *.java failus.

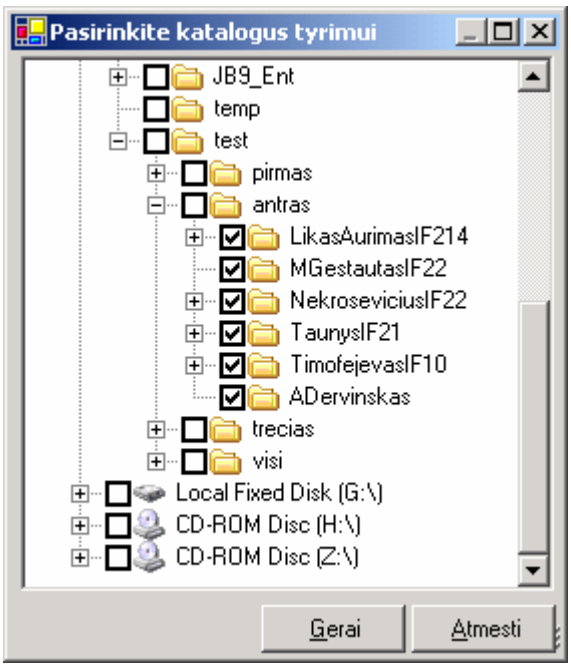
Sistemos vadovas

Pradėjus darbą su sistema reikia nuspręsti ką tirsime – failus ar katalogus. Tiriant failus, pasirinkite meniu punktą 'Failai->Pasirinkti failus tyrimui...'. Atsidarys langas, kuriame galite pasirinkti failus tyrimui (1 pav.). Nurodę reikiamą diską ir katalogą, laukelyje 'Failai' pamatysite visus tame kataloge esančius *.java failus. Jei norite tirti kitus failus, pakeiskite laukelio 'Failų tipai' reikšmę į 'Visi failai'. Failus, kuriuos norite tirti pažymėkite pelės pagalba ir spauskite mygtuką '->'. Jei norite tirti visus laukelyje esančius failus, galite spausti mygtuką '->>' – šiuo atveju pažymėti failus nebūtina. Mygtuku '<-' galite atmesti laukelyje 'Pasirinkti failai' pažymėtus failus, o mygtuku '<<-' galite atmesti visus laukelyje 'Pasirinkti failai' esančius failus. Žymėdami 'Rodyti kelią' galite pasirinkti ar rodyti failus su pilnais keliais ar tik failų vardus. Nusprendę, kad jau pasirinkote reikiamus failus, savo sprendimą patvirtinkite paspausdami mygtuką 'Gerai', o nutarę, kad tyrimo atlikti nebenorite, spauskite mygtuką 'Atmesti'.



1 pav. Failų pasirinkimo langas.

Tiriant katalogus, pasirinkite meniu punktą 'Failai->Pasirinkti katalogus tyrimui...'. Atsidarys langas, kuriame galite pasirinkti katalogus tyrimui (2 pav.). Pažymėkite norimus tirti katalogus ir patvirtinkite savo pasirinkimą mygtuku 'Gerai'. Persigalvoję spauskite mygtuką 'Atmesti'.

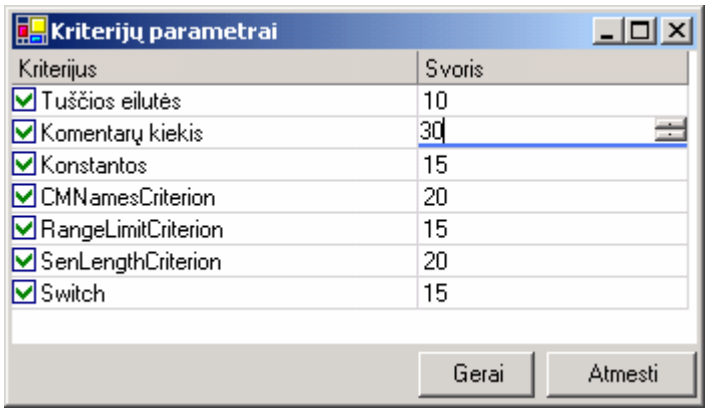


2 pav. Katalogų pasirinkimo langas.

Priklausomai nuo to kokį tyrimą norite atlikti ir nuo to ar tirsite failus, ar katalogus pasirinkite meniu punktus:

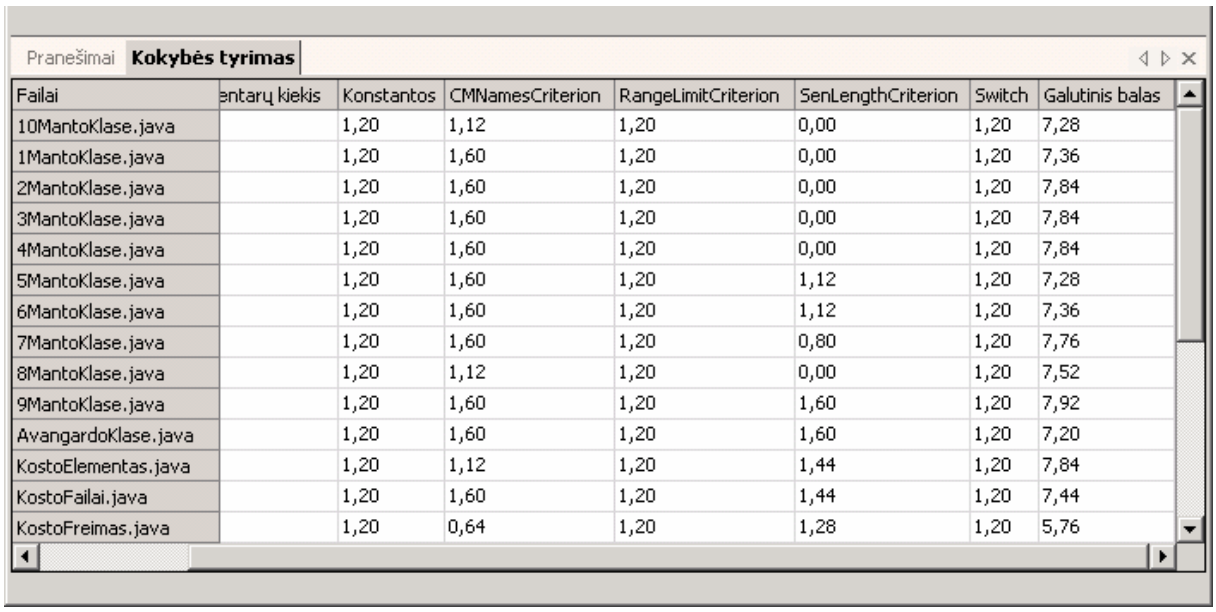
- 'Tyrimas->Kokybės įvertinimas->Tirti pasirinktus failus...',
- 'Tyrimas->Kokybės įvertinimas->Tirti pasirinktus katalogus...'
- 'Tyrimas->Plagiato nustatymas->Tirti pasirinktus failus'
- 'Tyrimas->Plagiato nustatymas->Tirti pasirinktus katalogus'

Jei pasirinkote kokybės tyrimą, atsidarys kriterijų parametrų langas (3 pav.). Pažymėkite kriterijus, kuriuos norite tirti. Stulpelyje svoris galite nustatyti kiekvieno kriterijaus svorį – tereikia pelės mygtuku du kartus paspausti ant norimo langelio ir klaviatūros pagalba arba spaudžiant atsiradusius mygtukus pasirinkite norimą svorį. Kriterijaus svoris nurodys kokią įtaką bendram pažymiui turės šis kriterijus. Savo pasirinkimus patvirtinkite mygtuku 'Gerai'.



3 pav. Kriterijų parametrų nustatymo langas.

Tyrimo rezultatus pamatysite lentelėje (4 pav.).

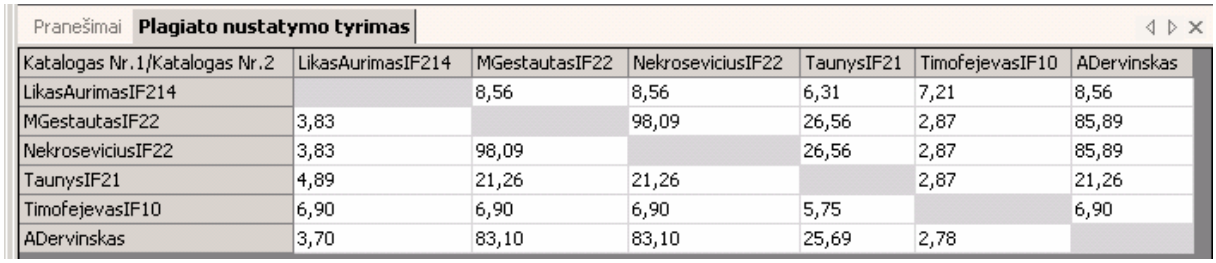


Pranešimai | **Kokybės tyrimas**

Failai	entarų kiekis	Konstantos	CMNamesCriterion	RangeLimitCriterion	SenLengthCriterion	Switch	Galutinis balas
10MantoKlase.java		1,20	1,12	1,20	0,00	1,20	7,28
1MantoKlase.java		1,20	1,60	1,20	0,00	1,20	7,36
2MantoKlase.java		1,20	1,60	1,20	0,00	1,20	7,84
3MantoKlase.java		1,20	1,60	1,20	0,00	1,20	7,84
4MantoKlase.java		1,20	1,60	1,20	0,00	1,20	7,84
5MantoKlase.java		1,20	1,60	1,20	1,12	1,20	7,28
6MantoKlase.java		1,20	1,60	1,20	1,12	1,20	7,36
7MantoKlase.java		1,20	1,60	1,20	0,80	1,20	7,76
8MantoKlase.java		1,20	1,12	1,20	0,00	1,20	7,52
9MantoKlase.java		1,20	1,60	1,20	1,60	1,20	7,92
AvangardoKlase.java		1,20	1,60	1,20	1,60	1,20	7,20
KostoElementas.java		1,20	1,12	1,20	1,44	1,20	7,84
KostoFailai.java		1,20	1,60	1,20	1,44	1,20	7,44
KostoFreimas.java		1,20	0,64	1,20	1,28	1,20	5,76

4 pav. Kokybės tyrimo rezultatų langas.

Pasirinkus katalogų tyrimą, tyrimas bus pradėtas iškart – jokių pasirinkimų nebus. Rezultatų lentelėje duomenys reiškia failo/katalogo eilutėje panašumo su failu/katalogu stulpelyje lygį procentais.



Pranešimai | **Plagiato nustatymo tyrimas**

Katalogas Nr.1/Katalogas Nr.2	LikasAurimasIF214	MGestautasIF22	NekroseviciusIF22	TaunysIF21	TimofejevasIF10	ADervinskas
LikasAurimasIF214		8,56	8,56	6,31	7,21	8,56
MGestautasIF22	3,83		98,09	26,56	2,87	85,89
NekroseviciusIF22	3,83	98,09		26,56	2,87	85,89
TaunysIF21	4,89	21,26	21,26		2,87	21,26
TimofejevasIF10	6,90	6,90	6,90	5,75		6,90
ADervinskas	3,70	83,10	83,10	25,69	2,78	

Norint išsaugoti gautus rezultatus faile, pasirinkite meniu punktą 'Failas->Išsaugoti rezultatus...'. Atsiradusiame lange, nurodykite failo vardą ir tipą. Tipas gali būti HTML – interneto puslapis arba XML – specialus duomenų saugojimo formatas.

Baigę darbą, galite išjungti programą meniu punktu 'Failas->Baigti darbą'.

9.1.3 Sistemos instaliavimo dokumentas

Sistemai pakanka kompiuterio su x86 šeimos procesoriumi ir 64 MB RAM. Papildomų periferinių įrenginių sistema nenaudoja, tad jie nėra būtini. Naudojama MS Windows šeimos (Win9x/NT/2000/XP) operacinė sistema. Kompiuteryje, kuriame bus diegiama programa, turi būti instaliuota Microsoft .NET Framework 1.1 (parsisiųsti galima iš http://windowsupdate.microsoft.com/).

Instaliavimas atliekamas paleidus failą Setup.exe. Instaliavimo metu sukuriami šie failai:

PMKVS.exe	programa
MagicLibrary.dll	biblioteka
Microsoft.VisualBasic.Compatibility.dll	biblioteka

SourceGrid.dll	biblioteka
SourceLibrary.dll	biblioteka
stdole.dll	biblioteka
TreeViewExplorer.Source.dll	biblioteka
xslt\similarity.xsl	failas XML vertimui į HTML
xslt\quality.xsl	failas XML vertimui į HTML

9.2 Straipsnio "Programinių modulių kokybės vertinimas kompiuterizuotoje žinių testavimo sistemoje" medžiaga. Projektas pristatytas tarptautinėje konferencijoje „Pažangios informacinės technologijos ir jų taikymas studijų procese“, bei publikuotas konferencijos metu platintame leidinyje.

Programinių modulių kokybės vertinimas kompiuterizuotoje žinių testavimo sistemoje

Deividas Grigas, Bronius Tamulynas
Kauno technologijos universitetas, Kompiuterių tinklų katedra,
Studentų 50, LT-3028 Kaunas, Lietuva

Straipsnyje pristatoma kompleksinė studentų žinių kontrolės posistemė, kurios pagalba galima įvertinti suderintų ir korektiškai veikiančių programinių modulių kokybę. Ji apima dviejų tipų kriterijus: plagijato nustatymo ir programos kodų racionalų naudojimą. Sistema gali būti panaudota kompiuterizuotose programavimo disciplinų studijose bei moksleivių informatikos mokyme.

Įvadas

Studijų kompiuterizuoto žinių testavimo sistemos testų rinkinius sudaro klausimai, kurių turinys dažniausiai siejamas su teorinių žinių vertinimu: sąvokos, teiginiai, metodai ir pan. Klausimų ir atsakymų pateikimo formos būna įvairios: atsakymas lyginamas su vienu ar keliais teisingais etaloniniais šablonais, kontekstinės užduotys ir pan. Suprantama, toks principas negarantuoja objektyvaus įgytų žinių įvertinimo, nes galima atspėti teisingą atsakymo variantą arba išmokti visus galimus teisingus atsakymus. Dar daugiau, specialiosios studentų žinios ir ypač programavimo įgūdžiai įgyjami atliekant praktines užduotis ir programavimo laboratorinius darbus. Tam, kad studentas sugebėtų savarankiškai dirbti būtinas nagrinėjamos srities supratimas ir kompleksinis modulio temų medžiagos įsisavinimas. Studijose siekiama, kad studentas visapusiškai sugebėtų manipuliuoti teorinėmis žiniomis ir lavintų sugebėjimus kaip taikyti jas praktikoje. Toks žinių ugdymo principas gali būti realizuotas per grįžtamąjį studento ryšį su dėstytoju. Jei grupėje studentų skaičius yra gerokai per 100, žinių kontrolė dėl objektyvių priežasčių tampa paviršutiniška, nes praktinio darbo rezultatų įvertinimas reikalauja daug laiko ir rezultatas neduoda tikėtino efekto. Kol kas praktiškai nėra realizuotų kompiuterizuotų laboratorinių ir praktinių darbų tikrinimo ir vertinimo priemonių, kurios galėtų palengvinti dėstytojo darbą.

Užsienyje panašią sistemą naudoja kai kurie Didžiosios Britanijos universitetai. Viena iš jų, komercinė **sistema Ceilidh** [3], platinama už tam tikrą mokestį ir orientuota **Java** kalba sukurtų programinių modulių kokybės vertinimui. KTU Informatikos fakultete buvo sukurtas programinių modulių kokybės vertinimo sistemos prototipas, kuris leidžia tikrinti programavimo disciplinos laboratorinius darbus (programinius modulius, parašytus C/C++ kalba). Ji analizuoja studentų atliktų laboratorinių darbų kokybę keliais aspektais:

3. programų kokybės tyrimas – analizuoja ar studento programa tenkina programavimo kokybei keliamus reikalavimus (programavimo kultūros, resursų naudojimo, saugumo, sudėtingumo ir panašius kriterijus);
4. plagijato nustatymas – analizuoja ar pagal tas pačias užduotis studento atlikti laboratoriniai darbai yra panašūs į kitų studentų.

Sukurtą sistemos prototipą numatoma įtraukti į kompleksinę studentų žinių kontrolės ir vertinimo sistemą TestTool, kuri jau keletą metų taikoma programavimo disciplinų moduliuose “Informatika” bei “Duomenų struktūros ir algoritmų analizė”.

1. Sistemos funkcinė paskirtis

Programinių modulių kokybės vertinimo sistema skirta studentų programavimo laboratorinių darbų kokybės tyrimui ir plagijavimo atvejų nustatymui. Ši sistema yra pritaikyta programų, parašytų C/C++ programavimo kalba, tyrimui. Programą sudaro šie moduliai: kokybės vertinimas, plagijato nustatymas ir vartotojo sąsaja. Sistemai pateikiamos programos turi būti sutransliuotos ir testavimo rezultatai teisingi. Išeities tekstų failus dėstytojas surenka iš studentų arba patys studentai persiunčia juos saugojimui atitinkamam kataloge. Vartotojo patogumui, failų pavadinimai sudaromi pagal iš anksto nustatytą schemą: failo pavadinimas - pirmosios keturios studento pavardės ir pirmų keturios vardo raidės. Pavyzdžiui: jei programą sukūrė *Jonas Petraitis*, tai failas turėtų vadintis *jonapetr.cpp*. Rezultatai saugomi HTML faile Web serveryje ir studentai per tinklo naršyklę gali su jais susipažinti. Funkcinė sistemos schema ir duomenų srautai pavaizduoti 1 paveikslėlyje. Sistemos programinei įrangai taikomi šie bendri aprobojimai:

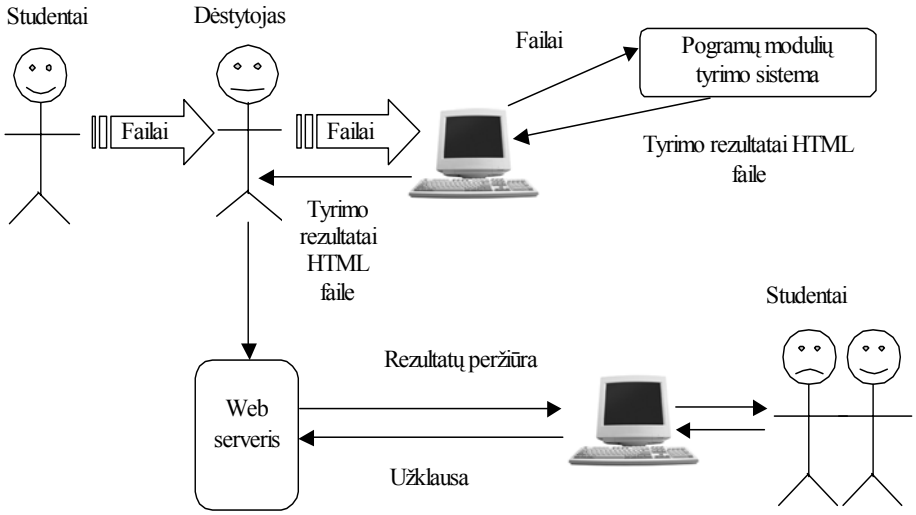
- programos turi būti sintaksiškai teisingos, t.y. C/C++ kalbos kompiliatorius neturi rasti jose sintaksinių klaidų;

- plagijato nustatymas yra paremtas pagrindinėmis C/C++ kalbos kodų rašymo taisyklėmis.Tačiau dėl C/C++ kalbos įvairiapusiškumo, duomenų aprašymo būdų ir komandų rašymo įvairovės bei kitų specifinių priežasčių plagijato nustatyme gali atsirasti netikslumų. Pagaliau programos gali būti įvertintos kaip nesulyginamos (plagijato nustatymo modulis nustatė kažkokius neapibrėžtumus ir dėl to negali palyginti programų);
- programų išeities tekstai yra viename faile. Programos vertinimui neįtraukiami ***#include*** direktyvoje nurodyti failai;
- kokybės tyrimo modulis vertina tik tuos kriterijus, kurie yra įtraukti į testuotinų kriterijų sąrašą;
- tyrimo moduliai realizuoti naudojant DLL failus, todėl abiejų vertinimo rezultatų failai *SoftEval.dll* ir *SimilEval.dll* turi būti tame pačiame kataloge kaip ir pagrindinė programa;
- ypatingų ribojimų dėl kompiuterio operatyvinės atminties ir procesoriaus nėra.

2. Programų kokybės vertinimo modulis

Programų kokybės vertinimo modulis pagal iš anksto nustatytą kriterijų sąrašą vertina programimius modulius. Bendras programos kokybės įvertis skaičiuojamas pagal kiekvienam kriterijui subjektyviai priskirtą svorinį koeficientą bei konkretų jo įvertį ir turi tenkinti šiuos reikalavimus:

- kokybės tyrimui programos struktūra didelės įtakos neturi, todėl kokybę galima tirti analizuojant programos kodą po eilutę;
- reikia apibrėžti programų kokybės vertinimo kriterijus ir juos deklaruoti;
- naujai sukurti kriterijai turėtų būti lengvai įterpiami į jau esančių sąrašą arba netinkami lengvai pakeičiami;
- tyrimo procesą galima paspartinti praleidžiant tuščias eilutes ir komentarus, fiksuojant būtiną jų kiekį programos tekste;
    - pageidautina, kad būtų nurodytos priežastys, kodėl kriterijus yra vertinamas būtent tokiu balu.



1 pav. Išoriniai sistemos duomenų srautai

Kokybės vertinimo kriterijai parinkti remiantis ***Ceilidh*** dokumentacija [Ceilidh reference] [3] ir tarnybiniais įvairių programavimo firmų dokumentacija [1,2]. Programų kokybės vertinimo modulis tuo tarpu naudoja rinkinį kriterijų, kurie leidžia įvertinti šiuos kokybinius parametrus:

1. *Komentarų kiekis*. Programavimo kultūrą ir programos dokumentavimą apibrėžiantis kriterijus. Paprastai programos tekste komentarai turėtų sudaryti 10% – 60% bendro programos teksto ilgio.
2. *Tuščių eilučių kiekis.* Tai programavimo kultūros rodiklis. Tuščios eilutės turėtų sudaryti 15%–30% bendro programos teksto ilgio. Nors perdidelis tuščių eilučių kiekis nėra geras rodiklis, tačiau perdaug kompaktiškas programos tekstas taip pat sunkiai skaitomas.
3. *Klasių ir metodų vardų, prasidedančių iš mažosios raidės, kiekis.* Rekomenduojama vardus rašyti iš didžiosios raidės, nes tai leidžia geriau skirti kintamųjų vardus nuo klasių ar metodų vardų. Kintamųjų ar objektų vardai turėtų prasidėti mažąja raide, nes lengviau skaityti programos tekstą.

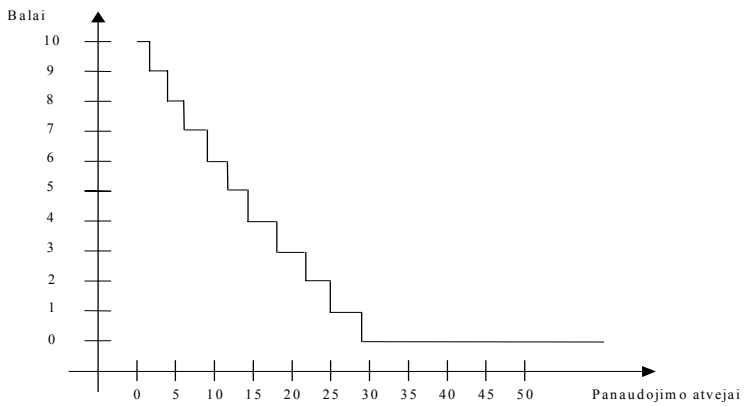
4. *Vidutinis programos sakinių ilgis.* Šis kriterijus charakterizuoja programos teksto sudėtingumą. Vartotojas sunkiau supranta ilgus sakinius, todėl klaidų tikimybė didėja. Patartina ilgus sakinius skaidyti į kelis trumpesnius.
5. *Masyvo rėžiai yra aprašyti, naudojant konstantas bet ne jų vardus*. Jei keičiasi uždavinio sąlyga (pavyzdžiui padidėja masyvo apimtis), tuomet pakanka pakeisti tik konstantos reikšmę.
6. *Neteisingų intervalo rėžių kiekis **for** cikluose.* Nurodant intervalo viršutinį rėžį, jis turi būti aprašomas neimtinai. Tada intervalo dydis yra lygus rėžių skirtumui. Taip yra aiškiau suprasti kiek kartų bus kartojamas ciklas. Be to, naudojant masyvus, sumažėja tikimybė "išeiti" už masyvo ribų.

   Pvz.: `for (int i = 0; i <= 98; i++)` turėtų būti `for (int i = 0; i < 99; i++)`
7. *Trinant duomenis, į kuriuos rodė rodyklė, jos reikšmė turi būti nustatyta į **NULL**.* Jei to nėra, padidėja galimybė įsilaužti į sistemą. Saugumas yra vienas iš svarbiausių PĮ kokybės reikalavimų. Patariama rodyklę nustatyti į ***NULL*** tuoj pat po duomenų ištrynimo.
8. ***#define*** direktyvų naudojimas konstantų aprašymui. Konstantos turi būti deklaruojamos raktiniu žodžiu *const.* Aprašant konstantas direktyva ***#define***, nenurodomas konstantos tipas, todėl kompiliatorius tipą priskiria pagal nutylėjimą ir naudoja papildomai daugiau atminties.

   Pvz.: `#define MAX_LENGTH 15000` turėtų būti `const unsigned int MAX_LENGTH = 15000`
9. ***goto** raktinių žodžių naudojimas.* Jų naudojimas blogina programos struktūrą, didina jos teksto sudėtingumą. Todėl ***goto*** sakinio naudoti nerekomenduojama.
10. ***switch*** raktinių žodžių naudojimas. Jų naudojimas taip pat blogina programos struktūrą, didina jos sudėtingumą. ***switch*** naudoti galima, tačiau reikia mažinti jų kiekį, keičiant ***switch*** į ***if*** sakinius ar kitomis priemonėmis.

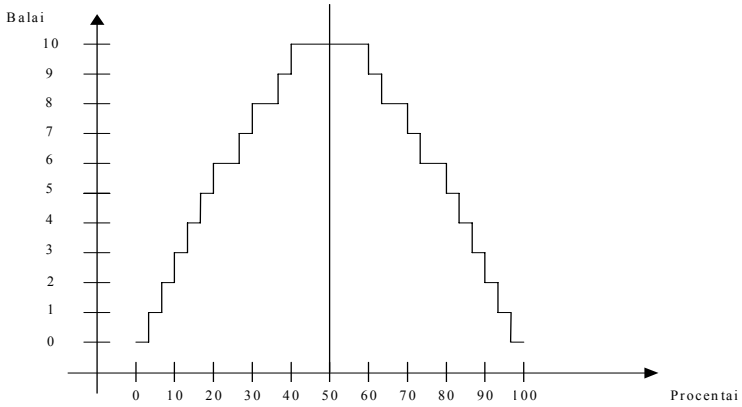
Kriterijai yra vertinami dešimties balų skalėje. Tyrimo rezultatai surašomi du masyvus: kriterijų tyrimo rezultatų ir tyrimo rezultatų įvertinimų. Atlikus programos tyrimą, gautus rezultatus galima vertinti dviem būdais:

3. Kiekvienam įvertinimo balui sudaromi galimų rezultatų intervalai (vertinimo skalė žr. 2 pav.). Vertinant yra tikrinama, į kurį vertinimo skalės intervalą patenka rezultatas, gautas tyrimo metu. Pvz.: raktinį žodį ***switch*** programoje galima naudoti, taciau, jei jis programoje buvo surastas 2-4 kartus, tuomet įvertinimas pagal šį kriterijų bus 9, jei- nuo 4 iki 6 kartų - įvertinimas – 8, nuo 6 iki 9 kartų - įvertinimas – 7 ir t.t. Tokiu būdu, vertinimo skalė: 10: 0–2; 9: 2–4; 8: 4–6; 7: 6–9; 6: 9–12; 5: 12–14; 4: 14–18; 3: 18–22; 2: 22–25; 1: 28–29; 0: > 29.



2 pav. Kriterijų intervalinė vertinimo skalė

4. Kriterijaus rezultatus galima vertinti pagal Gauso dėsnį. Paprastai šiuo būdu yra vertinami kriterijai, kuriuose skaičiuojamas santykinis įvertis (žr. 3 pav.). Vertinimo skalės sudarymas yra panašus kaip ir intervalų metode. Skirtumas tik tas, kad šio metodo vertinimo skalė yra simetriška tyrimo rezultatui, kuriam esant buvo gautas maksimalus įvertis. Tarkime, geriausias atvejis, kuomet komentarai programoje sudaro nuo 40 iki 60% teksto. Tada vertinimo skalė bus simetriška rezultatui, lygiam 50%. Vertinimo skalė: 10: 50–40%; 9: 40–35%; 8: 35–29%; 7: 29–25%; 6: 25–21%; 5: 21–17%; 4: 17–14%; 3: 14–10%; 2: 10–8%; 1: 8–2%; 0: 2–0%. Kaip matome, skalė yra simetriška 50% rezultatui.



3 pav. Kriterijų vertinimo skalė pagal Gauso dėsnį

Pagal kriterijų įverčius yra apskaičiuojami bendri įverčiai ir pagal juos suformuojamas programų kokybės įvertinimų grafikas. Apskaičiavus kiekvieno kriterijaus įvertį, galima nustatyti bendrą tiriamos programos kokybės įvertinimą. Jis randamas pagal formulę:

$$P = 10 - \left(\frac{\sum_{i=1}^{n} (X_i \times (10 - Q_i))}{\sum_{j=1}^{n} X_j} \right)$$

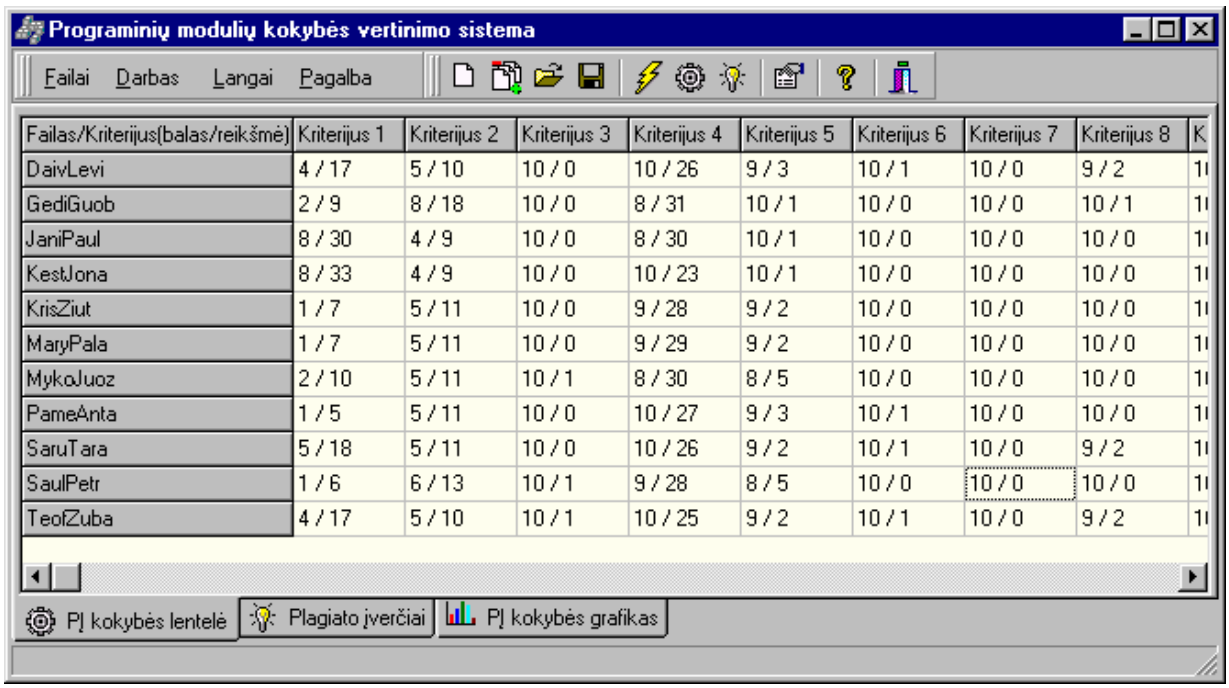
Čia:

P – bendras tiriamos programos įvertinimas;

X_i – i – tojo kriterijaus svoris. Vardiklyje yra visų kriterijų svorių suma;

Q_i – i – tojo kriterijaus įvertis;

Skaičius *10* šioje formulėje reiškia, kad programų vertinimui buvo pasirinkta dešimties balų vertinimo skalė.



4 pav. Kokybės vertinimo sąsajos langas

Ketvirtajame paveikslėlyje parodytas kokybės vertinimo sąsajos langas ir kriterijų reikšmės, suskaičiuotos pagal pateiktą formulę.

3. Plagijavimo atvejai programos tekste

Plagijato nustatymo modulis turi paporiui sulyginti keletą programų ir nustatyti programų panašumo (plagijavimo) lygį. Suprantama, kad programos turi spręsti identiškas užduotis, tačiau jose turėtų atsispindėti "autorystė", kadangi tikimybė, jog skirtingi autoriai idealiai vienodai koduotų vidutinio sudėtingumo panašaus algoritmo žingsnius nėra didelė. Taigi sulyginimo programa turi identifikuoti labai panašias programas, į kurias dėstytojas turėtų atkreipti dėmesį. Tokiu būdu sistema, įvertinusi konkrečią programą, pateikia rezultatus ir dėstytojas priima galutinį sprendimą pagal objektyvų plagijavimo fakto įvertį. Plagijavimo nustatymo moduliui nurodomi keliai iki lyginamųjų išeities tekstų failų. Vertinimo rezultatai pateikiami lentelėse. Nustačius, kad plagijavimo tyrimai identifikuoja labai panašias programas, t.y. tas, kurių įvertinimas yra didesnis už tam tikrą fiksuotą slenkstį, jos surašomos į atskirą sąrašą. Plagijavimas vertinamas trimis etapais: programų kodo formatavimas, modulinės programų struktūros analizė, programų modulių palyginimas. Plagijato identifikavimo modulis turi atsižvelgti į tai, kad autorius:

- gali lengvai keisti programavimo kultūrą: programą galima papildyti tarpais, tuščiomis eilutėmis, komentarais. Taip pakeista programa nuo kito jos varianto skiriasi tik vizualiai, tuo tarpu jos veikimas yra identiškas tikrajam autoriniam programos variantui.
- programos modulius (funkcijas) galima sukeisti vietomis, t.y. C/C++ kalba parašytose programose pradžioje aprašius modulius, tolesnis jų išsidėstymas neturi esminio skirtumo.
- gali būti pakeisti kintamųjų bei funkcijų vardai. Tokiu atveju, žinoma, reikia atsižvelgti į kintamųjų galiojimo sritį. Atskirose funkcijose vienodai pavadintas kintamasis gali turėti skirtingą prasmę.
- keletą sakinių galima išdėstyti vienoje eilutėje, arba vieną sakinį galima išdėstyti keliose eilutėse. Yra tik keletas apribojimų dėl sakinių išsidėstymo (pvz. tekstas tarp kabučių ("") turi baigtis tame pačiame sakinyje arba turi būti atskirtas simboliu \ ir tiesiai po jo einančiu eilutės pabaigos simboliu).
- galima pakeisti duomenis, kuriais manipuliuoja programa. Tai dažniausiai pasitaikantis plagijavimo atvejis, kai programos pritaikomos dirbti su kitais duomenimis. Taip pakeitus programą, ji skirtųsi nuo originalo t.y. sukompiliuotas kodas nebūtų identiškas. Tačiau programos realizacija išlieka ta pati. Skiriasi tik pradiniai duomenys ir gaunami rezultatai.

Programų kodo formatavimas. Formatavimo tikslas yra pašalinti iš programinio kodo neapibrėžtumus. Šiame etape programų koduose išlieka pagrindiniai C/C++ kalbos sintaksės elementai, tačiau didelę reikšmių apibrėžimo aibę turintys elementai (kintamųjų reikšmės, kintamųjų vardai ir pan.) yra pakeičiami. Jokios įtakos programai neturintys elementai (kai kurie tarpai, tuščios eilutės, komentarai) pašalinami visai.

Veiksmai, atliekami programų kodo formatavimo metu:

- pašalinami visi komentarai;
- pašalinami visi nereikalingi tarpai ir tuščios eilutės, pavyzdžiui, sakinyje ***int *k;*** tarpas yra nereikalingas, nes kintamojo tipą ir jo vardą atskiria simbolis *;
- simbolinės reikšmės (esančios tarp kabučių ("" arba ' ')) pakeičiamos į identifikatorius;
- kiekvienas programos kodo sakinys yra perkeliamas į atskirą eilutę;
- tipai turintys modifikatorius (pvz.: short int) pakeičiami į identifikatorius;
- skaitinės reikšmės (pvz.: 5.2) pakeičiamos į identifikatorius;
- kintamųjų vardai pakeičiami jų tipais.

Modulinės programų struktūros analizė. Šiame etape nustatoma kokia eilės tvarka vykdant programą yra iškviečiamas kiekvienas modulis. Pavyzdžiui, turime tokį atvejį: funkcija *f1* yra iškviečiama programos pagrindinės funkcijos *main* pirmoje eilutėje; funkcija *f2* yra iškviečiama antroje eilutėje; funkcija *f3* yra iškviečiama funkcijoje *f1*. Taigi šioje programoje funkcijos bus iškviečiamos tokia tvarka: *f1, f3, f2.*

Programų modulių palyginimas. Šiame etape palyginamos formatavimo etape suformatuotos programos arba tiksliau jų moduliai. Pirmiausia palyginamos programų pagrindinės funkcijos *main.* Toliau moduliai lyginami pagal tvarką, gautą modulinės programų struktūros analizės etape. Failų turiniai nuskaitomi į eilutes ir perduodami programų palyginimo objektui. Programų palygintojas paeiliui perduoda abi programinio kodo eilutes kodo tvarkytojui. Kodo tvarkytojas gautas eilutes sutvarko ir grąžina atgal. Programų palyginimui reikia žinoti sutvarkytų kodų modulinę struktūrą. Todėl jis sutvarkytus kodus paeiliui perduoda programų modulinės struktūros analizatoriui. Iš pastarojo programų palygintojas gauna po objektą, kuriame surašyti programos moduliai pagal jų pasirodymo tvarką programos vykdymo metu ir šių modulių vieta faile bei ilgis. Pagal šiuos duomenis programų lygintojas išsirenka atskirus modulius ir fiksuoja programinio kodo eilučių kiekį bei sutapusių programinio kodo eilučių kiekį. Palyginus visus modulius, programų panašumo įverčiai gaunami pagal eilučių kiekį visuose moduliuose ir sutapusių eilučių kiekį visuose moduliuose. Įverčių reikšmes galima interpretuoti taip: X (pirmoje ar antroje) programoje yra n procentų vienodą prasmę turinčio programinio kodo iš Y (antros arba pirmos) programos. Moduliai palyginimi taip:

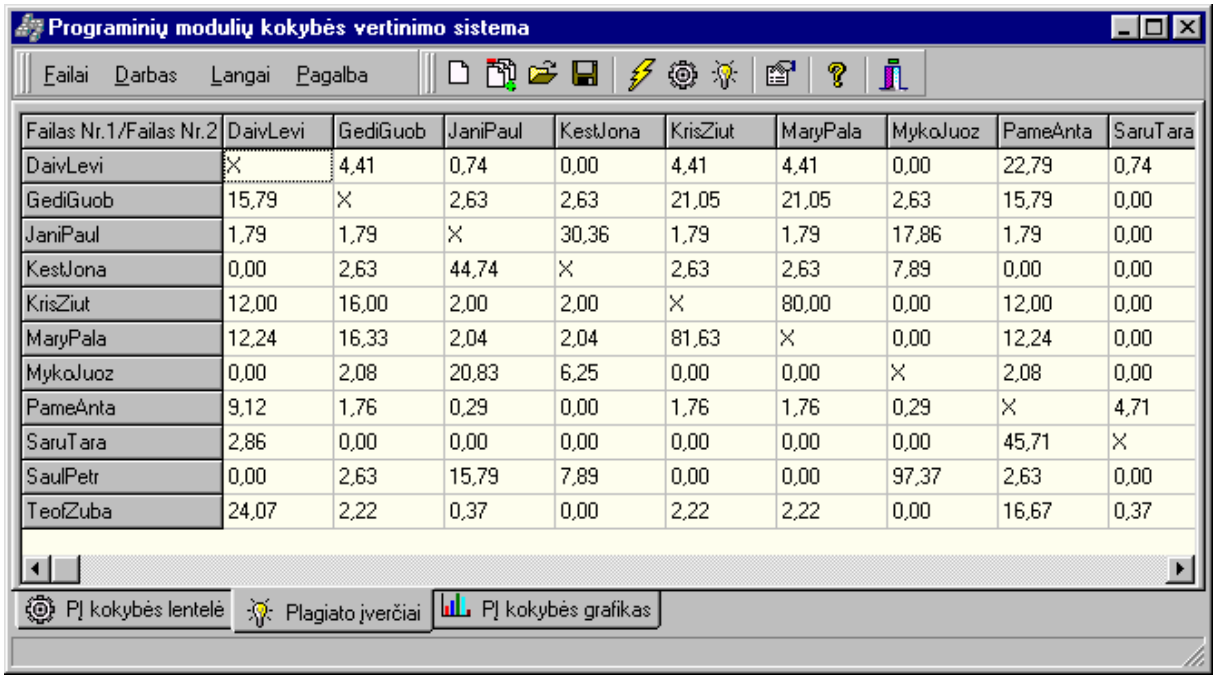
- iš pirmojo modulio nuskaitoma ir ištrinama pirma programinio kodo eilutė;
- kintamasis, rodantis eilučių kiekį pirmoje programoje, padidinimas vienetu;
- antrajame modulyje ieškoma punkte (1) nuskaitytos eilutės;
- jei antrajame modulyje tokia eilutė randama, ji ištrinama, kintamieji, rodantys eilučių kiekį antroje programoje bei sutapusių eilučių kiekį, padidinami vienetais. Priešingu atveju grįžtama į punktą (1);

- į šį punktą pereinama, kai ištuštėja pirmasis arba antrasis moduliai. Jei kažkuris iš modulių dar turi programinio kodo eilučių, jos yra ištrinamos ir atitinkamai padidinamas kintamasis, rodantis tos programos eilučių kiekį.

Taip palyginami visi moduliai ir įsimenamas bendras eilučių ir sutapusių eilučių skaičius. Jei programa turi daugiau modulių nei kita, jos kintamasis, rodantis eilučių kiekį, padidinamas tuose moduliuose esančių eilučių skaičiumi. Programų panašumo įverčiai gaunami sutapusių eilučių kiekį dalinant iš programos eilučių kiekio ir gautą rezultatą dauginant iš 100. Galimi programų palyginimo tobulinimo būdai:

- ✓ Programose ne visada yra svarbu, kad moduliai būtų iškviečiami tam tikra tvarka. Šiuo atveju plagijuojančiam asmeniui sukeitus modulių iškvietimo tvarką programoje, plagijato įvertinimo modulis gautų mažesnį rezultatą. Siekiant gauti tikslesnį įvertinimą, galima kiekvieną pirmos programos modulį lyginti su kiekvienu antros programos moduliu ir atrinkti labiausiai sutapusius. Deja, lyginant daug programų, šis būdas gali labai pailginti plagijavimo nustatymo modulio darbą.
- ✓ Programų modulių lyginimo etape neatsižvelgiama į sutapusių eilučių pozicijas moduliuose. Tai daroma dėl to, kad dažnai nėra svarbi programinio kodo eilučių vieta modulyje. Taigi tokia pati eilutė vieno modulio pradžioje ir kito pabaigoje fiksuoja sutapimą. Deja, tai ne visais atvejais yra teisinga. Todėl sutapusių eilučių kiekį galima būtų koreguoti pagal jų vietas moduliuose.

Sistemą numatoma adaptuoti ir **Java** kalba parašytų programų kokybės analizei. Sistema suprogramuota naudojant MS Windows šeimos (Win9x/NT/2000) operacinę sistemą. Techninė įranga priklauso nuo naudojamos OS ir turi atitikti naudojamos OS reikalavimus. Sąsaja realizuota naudojant standartinius Windows API elementus: mygtukus, meniu, lenteles ir t.t. Realizavimui pasirinkta *Borland C++ Builder* sistema. Atskiri sistemos moduliai yra realizuoti DLL failuose. Tai leidžia taupiau naudoti kompiuterio operatyvinę atmintį. DLL failai užkraunami į atmintį tada, kai programa kreipiasi į juose realizuotas funkcijas. Remiantis aukščiau pateiktomis specifikacijomis, realizacijai pasirinkta MS Visual C++ programavimo kalba. Penktajame paveikslėlyje pateikiamas sąsajos langas su programinių modulių panašumo įverčiais.



Failas Nr.1/Failas Nr.2	DaivLevi	GediGuob	JaniPaul	KestJona	KrisZiut	MaryPala	MykoJuoz	PameAnta	SaruTara
DaivLevi	X	4,41	0,74	0,00	4,41	4,41	0,00	22,79	0,74
GediGuob	15,79	X	2,63	2,63	21,05	21,05	2,63	15,79	0,00
JaniPaul	1,79	1,79	X	30,36	1,79	1,79	17,86	1,79	0,00
KestJona	0,00	2,63	44,74	X	2,63	2,63	7,89	0,00	0,00
KrisZiut	12,00	16,00	2,00	2,00	X	80,00	0,00	12,00	0,00
MaryPala	12,24	16,33	2,04	2,04	81,63	X	0,00	12,24	0,00
MykoJuoz	0,00	2,08	20,83	6,25	0,00	0,00	X	2,08	0,00
PameAnta	9,12	1,76	0,29	0,00	1,76	1,76	0,29	X	4,71
SaruTara	2,86	0,00	0,00	0,00	0,00	0,00	0,00	45,71	X
SaulPetr	0,00	2,63	15,79	7,89	0,00	0,00	97,37	2,63	0,00
TeofZuba	24,07	2,22	0,37	0,00	2,22	2,22	0,00	16,67	0,37

5 pav. Programų panašumo įvertinimai sąsajos lange

Programų palyginimo trukmė. Testuojant plagijavimo modulį (procesorius Celeron 450 MHz), buvo nustatyta, kad 20 programų (190 palyginimų), kurių kiekvienos apimtis siekia 100 programinio kodo eilučių, palyginimo trukmė neviršija 5 min. Analogiškai, 100 programų (4950 palyginimų) testui reikėtų ~1,5 val. kompiuterinio laiko.

Išvados

6. Pagal pateiktus studentų žinių analizės ir vertinimo kriterijus pasirinkta programų kokybės vertinimo kompiuterizavimo sistemos struktūra ir principai.
7. Sistema suprojektuota pagal *UML* specifikacijos reikalavimus ir realizuota aplinka, kuri apjungia du kokybės vertinimo modulius: plagijato nustatymo ir kokybės tyrimo. Sistema projektuota naudojant *evoliucinio vystymo* modelį.

8. Modulių testavimas programos kūrimo ir realizavimo etapuose patvirtino, kad sistema atitinka funkcinius, technologinius ir vartotojų poreikiams keliamus reikalavimus.
9. Sukurtą sistemą numatoma įtraukti į kompleksinę studentų žinių kontrolės sistemą TestTool, kuri jau keletą metų taikoma programavimo disciplinų moduliuose. Pats principas gali būti pritaikytas ir kitos srities praktinių darbų kokybės vertinimui, jei yra aiškiai apibrėžti darbų vertinimo ir jų palyginimo kriterijai. Sistema gali būti panaudota mokyklose, informatikos kursuose, mokant mokinius programavimo pagrindų.

Padėka. Autoriai nuoširdžiai dėkoja KTU Informatikos fakulteto magistrantams Audriui Petrikui ir Andriui Kaušiniui, kurių kvalifikuotas ir geranoriškas darbas yra dalis skelbiamų rezultatų.

Literatūros sąrašas

1. Fenton N. E. Software Metrics. A rigorous approach, 1992 Chapman & Hall, London;
2. Leinecher R. C. The Visual C++ Programmer's Reference, 1997 Ventana Press, USA;
3. Ceilidh reference, UK (http://www.cs.nott.ac.uk./~ceilidh).

Summary

Software Quality Evaluation through computer-based knowledge assesment system

Deividas Grigas, Bronius Tamulynas

Software quality evaluation system is presented. It could be used for two quality criterion: detection source program code plagiarism and racional source programming culture. The system is used in computer-based programming course studies as well as for informatics teaching.