Die opsporing van plagiaat in grafiese gebruikerskoppelvlakprogrammeringsopdragte

PS Botes

orcid.org/0000-0003-1416-8317

Verhandeling aanvaar ter gedeeltelike nakoming vir die graad *Magister Scientiae in Rekenaarwetenskap* aan die Noordwes-Universiteit

Studieleier: Dr JA Liebenberg

Mede-studieleier: Prof GR Drevin

Eksaminering Julie 2018

23591226

DANKBETUIGINGS

Hiermee wil ek my opregte waardering en dank betuig aan die volgende persone wat hierdie studie moontlik gemaak het:

- Baie dankie aan my ouers vir die onvoorwaardelike liefde, geduld, motivering en ondersteuning gedurende my studietydperk.

- My studieleier, Dr. J.A. Liebenberg vir al haar bekwame professionele leiding, wysheid, tyd en geduld. Asook vir haar bystand en opofferings gedurende hierdie studie se administratiewe stryd.

- My mede-studieleier, Prof. G. Drevin vir sy wysheid, rekenaartaalkundigheid en bystand.

- Familie en vriende vir die motivering en belangstelling in die studie en die vordering daarvan.

- Die SA Akademie vir Wetenskap en Kuns vir die finansiële ondersteuning.

- Die Noordwes-Universiteit vir die geleentheid om hierdie studie uit te voer.

- *Software analysis and forensic engineering* vir die verskaffing van 'n akademiese lisensie vir die gebruik van *CodeSuite*®.

- Statistiese Konsultasiediens, Prof. S.M. Ellis, vir haar advies aangaande die statistiese verwerking.

- Dr. I.J. Swart vir die taalkundige versorging.

- Laastens, maar allermins die minste, ons Hemelse Vader vir die daaglikse krag en genade wat Hy aan my geskenk het om hierdie studie tot die beste van my vermoë te voltooi.

OPSOMMING

Die voorkoms van en toename in plagiaat in programmeringskursusse is ’n kommerwekkende saak vir akademiese instellings. Die vinnige ontwikkeling in tegnologie en Internettoegang lei tot die bestaan van eenvoudige en slinkse maniere vir studente om programmeringsopdragte te plagieer. Die doel van hierdie studie is om die omvang van plagiaat in grafiese gebruikerskoppelvlakprogrammering te bepaal, met die oog op aanbevelings en riglyne aangaande programkodeplagiaat. Studies aangaande plagiaat in konsoleprogrammering bestaan, maar geen studie wat spesifiek fokus op plagiaat in grafiese gebruikerskoppelvlakprogrammering kon gevind word nie.

Die literatuurstudie ondersoek plagiaat en programkodeplagiaat, asook verskeie tegnieke en algoritmes wat deur geoutomatiseerde plagiaatherkenningstelsels gebruik word. Eerstens word ondersoek ingestel na plagiaat in die algemeen: die definisie, vorms, redes en metodes. Vervolgens word programkodeplagiaat gedefinieer en die verskeie metodes van verdoeseling van programkodeplagiaat word beskryf. Laastens word plagiaatherkenningstelsels bespreek, waar verskeie klassifikasies en benaderings van plagiaatherkenningstelsels beskryf word.

Die empiriese ondersoek se doel was om die omvang van plagiaat in programmeringsopdragte te ondersoek, asook om huidige programkodeplagiaat-herkenningstelsels te evalueer. Die analisering van die vraelys se resultate het aangedui dat die grafiese gebruikerskoppelvlakstudente verstaan wat plagiaat behels, asook die gevolge daarvan, maar plagiaatgevalle kom steeds voor. Daar is gevind dat die vrouestudente, die studente wat Turnitin ken, dié studente wat goeie tydsbestuur toepas, die Swart respondente en die studente wat hul eie akademiese prestasie tussen 0%-49% beoordeel, aandui dat hulle oor meer kennis beskik en beter ingelig is oor plagiaat. Teenstrydigheid aangaande die aanvaarbaarheid van plagiaat is vanuit die kwalitatiewe data verkry wat aandui dat die studente onder mekaar verskil ten opsigte van plagiaat; dit wil sê of dit reg of verkeerd is. Die top tegnieke wat deur die studente gebruik word om programkodeplagiaat te verdoesel is die verandering van die veranderlikes se name, kommentaar en die uitleg van die program (GGK). Vanuit die kwalitatiewe data het ’n nuwe verdoeselingstegniek na vore gekom, naamlik om die volgorde van die kode te verander. Die tegnieke wat die studente hoofsaaklik gebruik om kode aan medestudente te verskaf is om ’n foto van die programkode te neem en oor sosiale media te stuur of om die programmeringsopdrag op ’n geheuestokkie te laai.

Ses programkodeplagiaatherkenningstelsels, naamlik *AntiCutAndPaste Software Plagiarism detection software* (ACNP), *CodeMatch*®, *Copy/Paste Detector* (CPD), *JPlag*, *Measure of*

Software Similarity (MOSS) en *Simian* is geselekteer op grond van die vereistes dat die stelsels die programmeringstaal C# ondersteun en gratis beskikbaar is. Die stelsels se werking, vaardighede en opsporingsgehalte is geëvalueer deur tien plagiaatverdoeselingsmetodes wat uit die literatuur na vore gekom het in die programkode toe te pas. Die resultate van die evaluering van die stelsels het aangedui dat die programkodeplagiaatherkenningstelsels *CodeMatch*®, *JPlag* en *CPD* aanbeveel kan word vir die ontbloting van programkodeplagiaat in grafiese gebruikerskoppelvlakprogramme.

Trefwoorde: plagiaat; programkodeplagiaat; verdoeseling van programkodeplagiaat; grafiese gebruikerskoppelvlakprogrammering; programkodeplagiaatherkenningstelsels.

ABSTRACT

The occurrence of and increase in plagiarism in programming courses is a matter of concern for academic institutions. The rapid development in technology and easy Internet access lead to the existence of simple and cunning ways for students to plagiarise programming assignments. The aim of this study is to determine the extent of plagiarism in graphical user interface programming in order to provide recommendations and guidelines regarding source code plagiarism. Studies on plagiarism in console programming exist, but no study specifically focusing on plagiarism in graphical user interface programming could be found.

The literature study investigates plagiarism and source code plagiarism, as well as various techniques and algorithms used by automated plagiarism detection systems. Firstly, research into plagiarism in general is investigated: the definition, forms, motives and methods. Next, program source code plagiarism is defined and the various methods of concealment of plagiarism in program code are described. Finally, plagiarism detection systems are discussed, describing various classifications and approaches of plagiarism detection systems.

The aim of the empirical investigation was to investigate the extent of plagiarism in programming assignments and to evaluate current source code plagiarism detection systems. The analysis of the questionnaire results has shown that the graphical user interface students understand what plagiarism entails, as well as the consequences thereof, but plagiarism cases still occur. It was found that the female students, the students who know Turnitin, the students who apply good time management, the Black respondents and the students who judge their own academic achievement between 0% and 49% indicate that they have more knowledge

and they are better informed about plagiarism. Contradicting results regarding the acceptability of plagiarism were obtained from the qualitative data, which indicates that the students differ in their view on plagiarism, whether it is right or wrong. The top techniques used by students to hide source code plagiarism are changing the variables' names, comments and program layout (GGK). From the qualitative data, a new concealment technique has emerged, namely by changing the order of the code. The techniques the students used to provide code to fellow students are to take a photo of the code and send the program code via social media or to download the programming assignment onto a memory stick.

Six source code plagiarism detection systems, namely *AntiCutAndPaste Software Plagiarism Detection Software (ACNP), CodeMatch®, Copy / Paste Detector (CPD), JPlag, Measure of Software Similarity (MOSS) and Simian* were selected based on the requirements that the system supports the programming language C# and is available free of charge. The systems' functioning, skills and detection quality were evaluated by applying ten code plagiarism concealment methods that emerged from the literature in the program code. The results of the evaluation of the systems indicated that the source code detection systems *CodeMatch®, JPlag and CPD* could be recommended for the detection of program code plagiarism in graphical user interface programs.

Keywords: plagiarism; source code plagiarism; concealment of source code plagiarism; graphical user interface programming; source code plagiarism detection systems.

INHOUDSOPGAWE

HOOFSTUK 4

HOOFSTUK 5

LYS VAN TABELLE

LYS VAN FIGURE

LYS VAN AFKORTINGS

PPHS	Programkodeplagiaatherkenningstelsels
GGK	Grafiese gebruikerskoppelvlakke
VPO	Veranderde programmeringsopdragte
KON	Kontroles
LMI	Liggaamsmassa-indeks
ACNP	AntiCutAndPaste Software Plagiarism detection software
CPD	Copy/Paste Detector
MOSS	Measure of Software Similarity
HTML	Hiperteksmarkeertaal
JDK	Java ontwikkelingstel

AFRIKAANS - ENGELS WOORDELYS

Afrikaans	Engels
Attribuut-tellingstelsels	Attribute counting systems
Balie	Bin
Bevelaanporring	Command prompt
Dekompakteer / uitgepak	Unzip
Dokumentvingerafdruk	Document fingerprinting
Drumpelwaarde	Threshold
Enkele meting	Singular metrics
Foute/haakplekke	Bugs
Gepaarde meting	Paired metrics
Gids	Directory
Gulsige String-Teël algoritme	Greedy String Tiling algorithm
Hiperteksmarkeertaal	Hypertext markup language
Identifiseerder	Identifier
Identifiseerderpassingsalgoritme	Identifier matching algorithm
Instruksievolgorde ooreenstemmingsalgoritme	Instruction sequences match algorithm
Java ontwikkelingstel	Java development kit
Karp-Rabin stringooreenstemming-algoritme.	Karp-Rabin matching algorithm
Kenteken	Token
Konsolegebaseerde	Console based
Kommentaar/string ooreenstemmingsalgoritme	Comments/String match algorithm

Kompaklêer	Zip folder
Korporale meting	Corpal metrics
Meting	Metrics
Metriekstelsels	Metric system
Multidimensionele meting	Multi-Dimensional metrics
Opsporingsenjin	Detection engine
Parameterpassingsalgoritme	Parameterised matching
Sigblad	Spreadsheet
Skrip	Script
Stellingpassingsalgoritme	Statement matching algorithm
Stringpassing	String-matching
Struktuurmetingstelsel	Structure metric system
Vensterskepping	Winnowing
Verlenging	Extension
Verklaringooreenstemmingsalgoritme	Statement match algorithm
Verstek	Default
Verstekinstelling	Default setting
Verdelingsboom vergelyking	Parse tree comparison
Voorleggings	Submissions
Wysiger	Modifier

HOOFSTUK 1
ORIËNTERING, NAVORSINGSONTWERP EN METODOLOGIE

1.1 Probleemstelling en motivering

Met die wye beskikbaarheid van tegnologie en die Internet, is dit vir studente makliker om oplossings tot programmeringsprobleme te bekom en onder mekaar te versprei (Ohno & Murao, 2011; Arabyarmohamady *et al.*, 2012; Deokate & Hanchate, 2016). In 'n opname deur Sraka en Kaucic (2009) waarby 138 studente betrokke was, het 72.5% van die studente erken dat hulle plagiaat gepleeg het, waarna hulle verder beraam het dat ongeveer 75.5% van alle studente minstens een keer tydens hulle studies plagiaat sou pleeg.

Volgens Sraka en Kaucic (2009) se opname en ander navorsing, is plagiaat besig om toe te neem wat dus tot 'n ernstige opvoedkundige probleem lei (Vamplew & Dermoudy, 2005; Sraka & Kaucic, 2009; Liao, 2010; Arabyarmohamady *et al.*, 2012). Studente wat 'n opdrag gedeeltelik of volledig plagieer, leer dus minder as wat hulle sou, indien hulle dit self voltooi het. Die gevolg van studente wat plagieer is onvoldoende kennis oor die onderwerp. In uiterste gevalle kan die student selfs 'n kwalifikasie verwerf, maar hy/sy beskik nie oor die vereiste kennis en vaardighede nie (Vamplew & Dermoudy, 2005).

Volgens die Center for Higher Education Trust (CHET) (2013), het die aantal voorgraadse studente in Suid-Afrika wat rekenaar- en inligtingswetenskappe studeer vanaf 2009 tot 2013 met 12.5% toegeneem. Met hierdie toename in studente, is dit moeiliker om vas te stel of studente saamwerk of kopieer, veral wanneer daar gebruik gemaak word van 'n nie-geoutomatiseerde plagiaatherkenner vir programkode. Beskikbare statistieke oor studente wat wel uitgevang word as hulle plagiaat pleeg, is skraal. Culwin *et al.* (2001) rapporteer dat 'n gemiddeld van 5% van die studente jaarliks uitgevang word, wat impliseer dat 'n aansienlike aantal plagiaatgevalle tans ongemerk deurglip.

Daar is outomatiese programkodeplagiaatherkenningstelsels beskikbaar soos MOSS (Aiken, 1994), AC (Freire *et al.*, 2007), Ferret (Green *et al.*, 2012) en JPlag (Prechelt *et al.*, 2002) wat vir die algemene programmeringstale soos C, C++, C# en Java ontwerp is. Die meeste van hierdie herkenners is egter nie geskep om plagiaat in grafiese gebruikerskoppelvlak-programmeringsopdragte te ontleed nie.

Met die hoë aanvraag na Java-, C#- en .Net-ontwikkelaars in Suid-Afrika (MyBroadband, 2017) wat die ontwikkeling van grafiese gebruikerskoppelvlakvaardighede insluit, word daar verwag dat die studente oor voldoende en unieke kennis beskik. Daar bestaan studies wat

plagiaat in konsoleprogrammering met die programmeringstale C, C++ of Java (Joy & Luck, 1999; Culwin *et al.*, 2001; Kilinc *et al.*, 2015) ondersoek, maar geen studie wat spesifiek fokus op grafiese gebruikerskoppelvlakprogrammering kon gevind word nie.

Universiteite is uiters bewus van die probleem rondom plagiaat en beskik oor duidelik geformuleerde beleide aangaande plagiaatpleging. Dit wil egter voorkom dat nie alle universiteite oor 'n beleid spesifiek vir plagiaat in programmering beskik nie.

Uit die voorgaande is dit duidelik dat 'n gaping in die kennis ten opsigte van plagiaatherkenning spesifiek in grafiese gebruikerskoppelvlakprogrammering bestaan en dus wil hierdie studie ondersoek instel na plagiaat in grafiese gebruikerskoppelvlak-programmeringsopdragte om moontlike riglyne en oplossings aan te beveel.

1.2 Navorsingsdoelwitte en -doelstellings

Die hoofdoelwit van hierdie studie is om plagiaat op te spoor in programmeringsopdragte. Om hierdie doelwit te bereik, is die volgende sekondêre doelwitte gestel:

1. ondersoek die huidige literatuur aangaande plagiaat en geoutomatiseerde plagiaatherkenners;
2. ondersoek die omvang van plagiaat in programmeringsopdragte in voorgraadse modules;
3. evalueer huidige geoutomatiseerde programkodeplagiaatherkenners; en
4. stel riglyne spesifiek vir die hantering van plagiaat in programmering op.

1.3 Navorsingmetode en -ontwerp

1.3.1 Literatuurstudie

'n Literatuurstudie van primêre navorsingshulpbronne is onderneem. Daar is gebruik gemaak van die Scopus, ScienceDirect, Google Scholar, ACM digital library en IEEEXplore navorsingsdatabasisse en tydskrifartikels, en boeke en konferensiepublikasies is ook in die literatuurstudie gebruik. Die volgende sleutelwoorde is in die soektog aangewend: "*plagiarism", "automatic code testing", "automatic plagiarism detection", "source code plagiarism*" en "*source code plagiarism detection*".

Die literatuurstudie ondersoek drie sleutelkonsepte:

1. Plagiaat

Hierdie afdeling van die literatuurstudie fokus op plagiaat oor die algemeen deur verskeie definisies van plagiaat, verskeie vorms van plagiaat, redes waarom studente plagieer en hoe om plagiaat te vermy te bespreek.

2. Programkodeplagiaat

Hierdie afdeling van die literatuurstudie fokus op plagiaat, spesifiek in programkode. Die definisies van programkodeplagiaat word ondersoek, asook die metode van die verkryging van programkode onder studente en hoe daar te werk gegaan word om die plagiaat te verdoesel.

3. Geoutomatiseerde programkodeplagiaatherkenners

Hierdie afdeling van die literatuurstudie fokus op die ontstaan van programkodeplagiaat, prosesse wat gebruik word om plagiaat in programmeringsopdragte op te spoor, die klassifisering van plagiaatherkenningstelsels en die verskillende benaderings (algoritmes) tot geoutomatiseerde programkodeplagiaatherkenners.

1.3.2 Ondersoekmetodes

In hierdie studie is twee navorsingsmetodes gebruik, naamlik ’n opname en ’n eksperiment.

1.3.2.1 Opname

Vir navorsingsdoelwit 2 wat die omvang van plagiaat in grafiese gebruikerskoppelvlakmodules ondersoek, is die opnamenavorsingstrategie gebruik. Volgens Oates (2005), fokus ’n opname om dieselfde soort data by ’n groot bevolking te verkry op ’n gestandaardiseerde en sistematiese manier. Patrone in die data word deur middel van statistiek bestudeer.

1.3.2.1.1 Deelnemers

Die deelnemers van hierdie deel van die studie was 287 grafiese gebruikerskoppelvlak I-studente van ’n universiteit in Suid-Afrika. Die deelnemers bestaan uit persone van verskillende geslagte, etniese agtergronde, ouderdomme, studierigtings en akademiese jare.

1.3.2.1.2 Data-insamelingsmetode

’n Vraelys is vir die generering en insameling van data gebruik. Oates (2005) definieer ’n vraelys as ’n stel voorafgedefinieerde vrae wat in ’n voorafbepaalde volgorde weergegee word. ’n Vraelys word gebruik in die geval waar ’n navorser data by ’n groot bevolking wil insamel.

Die vraelys het uit beide oop vrae, asook geslote vrae bestaan. Die vraelys is in Google Forms opgestel en bestaan uit agt biografiese vrae, nege vrae aangaande plagiaat oor die algemeen en 16 vrae aangaande plagiaat in programmeringsopdragte.

1.3.2.1.3 Data-insameling

Voordat die vraelys bekend gemaak is, het die navorser eers klasbesoek aan die grafiese koppelvlakprogrammering I-module gebring, waar die studente volledig ingelig is oor die studie en die feit dat hul deelname vrywillig is en die data anoniem versamel word. 'n Skakel na die vraelys is daarna deur middel van die universiteit se e-leerstelsel aan die deelnemers beskikbaar gestel. Die skakel na die vraelys was vir ’n week tot die deelnemers se beskikking, waarna die ontvang van response gesluit het.

1.3.2.1.4 Data-analise

Basiese ontleding is op die data gedoen deur die gemiddelde en standaardafwykings te bereken. Frekwensies is vir die nie-Likert-skaalvrae bereken. Ses groepe is op grond van geslag, IT as ’n skoolvak, ingelig oor die Turnitin plagiaatherkenningstelsel, tydsbestuur, etniese agtergrond en gemiddelde akademiese prestasie vir programmeringsmodules uit die data geïdentifiseer. Hierdie groepe is vir praktiese en statistiese betekenisvolle verskil tussen die gemiddelde getoets deur gebruik te maak van T-toetsing en ANOVA’s.

1.3.2.1.5 Betroubaarheid

Faktoranalise is op 17 Likert-skaalvrae om ooreenkomstige veranderlikes as faktore te verklaar, toegepas. Die 266 respondente is ondersoek deur gebruik te maak van hoofkomponentfaktoranalise waarvan twee faktore genaamd Kennis_en_inligting en Verskaffing_van_programkode, asook vyf enkelstaande items verkry is.

Die betroubaarheid van hierdie twee faktore is verseker deur Cronbach se α koëffisiënte vir elke faktor te bereken. Cronbach se α koëffisiënte voorsien ’n maatstaf van die interne konsekwentheid van ’n toets of bron deur ’n skaal weer te gee van ’n getal tussen 0 en 1. Die interne konsekwentheid van ’n toets beskryf of al die items as dieselfde konsep gemeet kan word en of daar ’n verwantskap gekoppel kan word tussen die items binne die toets (Tavakol & Dennick, 2011).

1.3.2.1.6 Etiese aspekte van die navorsing

Die deelnemers aan die studie word deur die instelling waar die studie uitgevoer is, beskou as ’n kwesbare groep. ’n Volledige etiese aansoek is deur die navorser ingedien. Die navorser

het die voorgeskrewe etiese opleiding voltooi (kyk Bylaag L). Die studie het etiese goedkeuring van die Etiekkomitee van die Fakulteit Opvoedingswetenskappe *(ESREC) en* by die Noordwes-Universiteit Institusionele Navorsingsetiek Regulerende Komitee (NWU-IRERC) verkry by wyse van etiek-nommer: NWU-HS-2017-0074 (vir sertifikaat kyk Bylaag M).

1.3.2.2 Eksperiment

Volgens Oates (2005), is ’n eksperiment 'n strategie wat oorsaak- en gevolgverhoudings ondersoek om 'n oorsaaklike verband tussen 'n faktor en 'n waargenome uitkoms te bewys. In hierdie studie is ’n eksperiment uitgevoer wat ses programkode-plagiaatherkenningstelsels (PPHS) evalueer.

1.3.2.2.1 PPHS

Daar bestaan verskeie PPHS’e, maar vir hierdie studie se eksperiment was die vereistes dat die stelsels die programmeringstaal C# ondersteun en gratis beskikbaar is. Die volgende ses PPHS’e is derhalwe vir hierdie studie se eksperiment geselekteer:

1. *AntiCutAndPaste Software Plagiarism detection software* (ACNP);
2. *CodeMatch*®;
3. *Copy/Paste Detector* (CPD);
4. *JPlag*;
5. *Measure of Software Similarity* (MOSS); en
6. *Simian*.

1.3.2.2.2 Toetsdata

Die toetsdata van hierdie eksperiment was programmeringsopdragte wat oor kommentaar, *if*-stellings en *for*-stellings beskik. Twaalf C# programme is uit ’n programmeringswerksopdrag van studente vir die studie geselekteer. Die werksopdrag het behels dat studente ’n program moes skryfwat die gebruiker se liggaamsmassa-indeks (LMI) bereken deur die lengte en gewig van die gebruiker as toevoer te ontvang.

Om die PPHS te evalueer is Faidhi en Robinson (1987) se ses vlakke van hoe studente te werk gaan om plagiaat in programmeringsopdragte te verdoesel gebruik as riglyne om aanpassings tot die geselekteerde programmeringsopdragte aan te bring.

Kopieë van hierdie twaalf programmeringsopdragte is gemaak, waarna die veranderlikes, kommentaar, verklarings, programmodules, stellings, besluitlogika en grafiese gebruikerskoppelvlakke verander is.

1.3.2.2.3 Evalueringsproses

Die veranderde programmeringsopdragte is deur middel van elke PPHS met die kontrole programmeringsopdragte vergelyk. Die kontrole programmeringsopdragte is die oorpronklike programmeringsopdragte wat sonder enige veranderings die gebruiker LMI bereken. Hierdie stelsels het ŉ ooreenkomspersentasie gelewer wat as die maatstaf van plagiaatherkenning tussen die opdragte gebruik is.

Daar is in hierdie eksperiment twee toetsgevalle ondersoek:

1. vergelyking van slegs die kodelêers; en
2. vergelyking van die kodelêers saam met die ontwerplêers.

Die volledige proses wat gevolg is om die PPHS te evalueer word in Afdeling 3.1 weergegee.

1.4 Uitleg van die verhandeling

In Hoofstuk 2 word die literatuurstudie oor plagiaat en plagiaatherkenning, soos uiteengesit in 1.3.1 onderneem.

In Hoofstuk 3 word die eksperiment (die evaluering van PPHS) wat uitgevoer is, bespreek. Al ses PPHS'e tesame met die resultate wat in die uitvoering van die eksperiment verkry is, word uiteengesit. ŉ Kort refleksie oor die PPHS'e word weergegee.

Die opname en resultate van die omvang van plagiaat word in Hoofstuk 4 beskryf tesame met die bespreking en interpretasie van die resultate.

Die gevolgtrekking, aanbevelings, asook die riglyne vir die hantering van plagiaat in programmering word in Hoofstuk 5 bespreek.

1.5 Bydrae van die studie

Die navorsing gaan dus ŉ bydrae op die volgende gebiede kan lewer:

- verbetering van kennis oor die omvang van plagiaat in programmeringskursusse;
- bydraes tot kennis van bestaande geoutomatiseerde programkodeplagiaat-herkenners vir grafiese koppelvlakke; en
- riglyne vir die hantering van plagiaat in programmering vir gebruik deur dosente van programmeringsklasse.

HOOFSTUK 2
PLAGIAAT EN PLAGIAATHERKENNING

In hierdie hoofstuk word die agtergrond van plagiaat en plagiaatherkenning geskep deur die verwante literatuur aangaande plagiaat, opsporings- en verdoeselingstegnieke van plagiaat en die metodes wat deur programkodeplagiaatherkenningstelsels (PPHS) gebruik word te bestudeer.

2.1 Definiëring van plagiaat

In die literatuur kom Hannabuss (2001) se definisie van plagiaat die meeste voor wat plagiaat definieer as die ongemagtigde gebruik of nabootsing van iemand anders se idees, taal of uitdrukkings. Hierdie definisie oorvleuel met dié van die HAT (2005:868) wat plagiaat beskryf as “die oorneem van gedagtes, argumente, ens., uit ’n ander skrywer se werk en dit as jou eie te laat deurgaan”.

Flint *et al.* (2006) het ’n opname uitgevoer om akademici se persepsies van plagiaat onder studente te verkry. Die akademici wat aan die opname deelgeneem het, was van verskeie departemente van ’n enkele universiteit. Uit hierdie opname is daar gevind dat personeel hulle eie persoonlike definisies het vir plagiaat. Hierdie definisies word vanuit die personeel se persepsies en ervarings aangaande plagiaat geskep en is nie ooreenstemmend met die instelling se beleid oor plagiaat nie. Sulke teenstrydigheid oor die definisie van plagiaat onder verskeie akademici lei daartoe dat plagiaat nie volgens die beleid van die instelling hanteer word nie (Martin, 1994; Flint *et al.*, 2006). Hierdie teenstrydigheid oor die persepsie van plagiaat kan onvermydelik verwarring onder studente skep met betrekking tot wat as plagiaat beskou word, en wat nie.

Vanuit Flint *et al.* (2006) se opname is dit duidelik dat ’n definisie aangaande plagiaat van belang is. Verskeie definisies van plagiaat in die literatuur bestaan. Barnhart (1988) het dus die etimologie van die woord plagiaat (“literêre diefstal”) gaan ondersoek. Die woord is afkomstig van die vroeë Engelse woord *plagiary* (“een wat verkeerdelik ’n ander se woorde of idees neem”), wat afgelei word van die Latynse woord *plagarius* (ontvoerder), wat van *plagium* (ontvoering) en *plaga* (vangnet, net) afkomstig is. Die term plagiaat word dus gewoonlik gebruik om te verwys na die steel van woorde of idees wat nie as algemene kennis beskou word nie.

Daar bestaan ’n beleid oor plagiaat, ander vorme van akademiese oneerlikheid en wangedrag by die Noordwes-Universiteit waar hierdie studie uitgevoer word (Bylaag A), wat plagiaat definieer as “die aanbied, sonder toestemming of bronverwysing, van ’n ander persoon se teks

of ander gepubliseerde intellektuele produk deur dit voor te doen as die oorspronklike werk van die persoon wat hoop om voordeel daaruit te trek" (Noordwes-Universiteit, 2013:2).

Vir die doel van hierdie verhandeling word plagiaat beskou as:

- die inhandiging van iemand anders se werk as jou eie;
- kopiëring van woorde of idees van iemand anders sonder om aan hom/haar erkenning te gee;
- verskaffing van die verkeerde inligting van die bron wat gebruik is;
- versuim om aanhalingstekens te gebruik vir 'n direkte aanhaling;
- verandering van die woorde van 'n gekopieerde sin sonder om erkenning te gee aan die bron; en
- kopiëring van so baie woorde of idees vanuit die bron dat dit die meerderheid van die werk beslaan (Plagiarism.org, 2012).

2.2 Vorms van plagiaat

Met die definisie (kyk 2.1) en wye omvang van plagiaat het Martin (1994); Clough (2000); Marshall en Garry (2005) gevind dat plagiaat in 'n verskeidenheid van unieke vorms verdeel kan word, naamlik

1. **Woord vir woord plagiaat:** direkte kopiëring van frases of gedeeltes van 'n gepubliseerde teks sonder om aanhalings te gebruik of erkenning te verleen.
2. **Parafraseringplagiaat:** wanneer woorde of sintaksis herskryf word, maar die bron-teks kan steeds herken word.
3. **Plagiaat vanaf sekondêre bronne:** wanneer daar verwys word na 'n oorspronklike bron of aangehaal word sonder om na die oorspronklike bron te gaan kyk.
4. **Plagiaat in die vorm van die bron:** woordelikse kopiëring of herskrywing van die struktuur van die bron se argument.
5. **Plagiaat van idees:** die hergebruik van 'n oorpronklike idee van 'n bron sonder dat die oorspronklike woorde of vorm van die bron gebruik word.
6. **Outeurskapplagiaat:** die geval waar jy jou eie naam op iemand anders se werk plaas.
7. **Vertaalplagiaat:** met die hand of outomatiese verandering van die konteks vanaf een taal na 'n ander om sodoende die oorsprong van die konteks te verskuil.

Hierdie verskillende vorms van plagiaat stel 'n plagiaris in staat om **opsetlik**, **onopsetlik** of **per ongeluk** plagiaat te pleeg.

Per ongeluk: is die geval wanneer daar ŉ gebrek aan kennis oor plagiaat en die begrip van verwysings bestaan.

Onopsetlik: wanneer daar baie inligting beskikbaar is oor ŉ spesifieke onderwerp veroorsaak dit dat jou gedagtes beïnvloed word deur hierdie magdom inligting wat kan lei tot die skrywe van dieselfde idees, maar jy sien dit as jou eie.

Opsetlik: word hier verwys na bewustelike kopiëring van iemand anders se werk sonder om behoorlik erkenning te gee aan daardie persoon (Beasley, 2004).

Hierdie vorms maak ook die wyse waarop plagiaat gepleeg word, bekend (Maurer *et al*., 2006). Wilhoit (1994); Brandt (2002); en Howard (2002) vind dat studente op vier hoof-maniere plagiaat pleeg deur

1. materiaal vanaf ŉ ander bron te steel en dit as hul eie aan te bied, bv. aankoop van ŉ opdrag, kopiëring van die hele opdrag vanaf ŉ bron sonder behoorlike erkenning en indiening van ŉ ander student se werk met of sonder sy/haar kennis.
2. indiening van ŉ opdrag as hul eie, maar is geskryf deur iemand anders.
3. kopiëring van gedeeltes vanaf bronne met die korrekte bronverwysing, maar sonder aanhalingstekens wat voorgee dat dit oorgeskryf is.
4. parafrasering van materiaal van een of meer bronne sonder die toepaslike dokumentasie en verwysings.

Daar bestaan verskeie redes waarom studente hierdie maniere gebruik om plagiaat pleeg. Dit word grondig in die afdeling hierna bespreek.

2.3 Waarom studente plagieer

Bennett (2005) het ŉ literatuurstudie gedoen aangaande faktore wat studente motiveer om te plagieer. Hieruit word daar drie kategorieë geklassifiseer, naamlik beskikbaarheid van middels en geleentheid, die student se persoonlike eienskappe en die student se persoonlike omstandighede.

Middels en geleentheid: Die wydverspreidheid van die Internet- en aanlynartikels lewer ŉ groot bydrae tot die toename in plagiaat onder studente omdat dit vir hul moontlik is om verskeie materiaal te vind en af te laai vanaf verskeie bronne sonder om bydraes te maak tot die materiaal. Daar bestaan ook dienste op die Internet waar studente ŉ navraag kan indien vir ŉ opdrag wat aan persoonlike spesifikasies voldoen. Met ŉ tekort aan reëls en uitvoering daarvan hits dit die studente aan om van hierdie dienste gebruik te maak.

Persoonlike eienskappe: Die moontlike interne oortuigings van studente dat akademiese kullery immoreel en oneerlik is, ontmoedig plagiaat. Daar is ook gevind dat hoogs godsdienstige studente geneig is om minder oneerlik op te tree as ander. Björklund en Wenestam (1999) het gevind dat die grootste motivering vir studente om plagiaat te pleeg die begeerte vir beter punte is, wat dan verder kan afhang van verskillende faktore, soos patologiese vrees vir mislukking, verbetering van selfbeeld, ens.

Individuele omstandighede: Daar bestaan verskeie omstandighede wat studente se beskikbare tyd vir akademiese werk beïnvloed en dus soek die studente dan ’n vinnige uitweg om die werklading van die Universiteit te verminder. Druk en swak tydsbestuur lei daartoe dat die studente wat plagieer vermeerder.

Daar bestaan ook verskeie ander redes in die literatuur wat met die kategorieë hier bo ooreenstem oor waarom studente plagieer. Hierdie redes met verwysings word vervolgens gerieflikheidshalwe in Tabel 2-1 gelys.

Tabel 2-1 Redes vanuit die literatuur oor waarom studente plagiaat pleeg

Redes	Outeur
Swak akademiese prestasie	(Love & Simmons, 1998; Joy & Luck, 1999)
Student is nie gemotiveerd nie	(Joy & Luck, 1999)
Plagiaat en wette aangaande plagiaat is swak gedefinieer, asook die gevolge daarvan	(Stevens & Stevens, 1987; Joy & Luck, 1999; Zobel & Hamilton, 2002; Sheard *et al.*, 2003; Devlin & Gray, 2007)
Persoonlike waardes of houdings teenoor die onderwyser of klas	(Stevens & Stevens, 1987; Zobel & Hamilton, 2002; Sheard *et al.*, 2003; Devlin & Gray, 2007)
Student het ’n gebrek aan selfvertroue	(Calabrese & Cochran, 1990; Raffetto, 1985)
Swak tydsbestuur	(Love & Simmons, 1998; Zobel & Hamilton, 2002; Sheard *et al.*, 2003; Devlin & Gray, 2007)
Groepsdruk	(Zobel & Hamilton, 2002; Sheard *et al.*, 2003; Burrows *et al.*, 2004; Devlin & Gray, 2007)

Tabel 2-1 Redes vanuit die literatuur oor waarom studente plagiaat pleeg (vervolg)

Redes	Outeur
Vriendskap of die begeerte om 'n klasmaat te help	(Zobel & Hamilton, 2002; Sheard *et al.*, 2003; Devlin & Gray, 2007)
Onderwyser het 'n gebrek aan die nodige onderrigvaardighede	(Zobel & Hamilton, 2002; Sheard *et al.*, 2003; Devlin & Gray, 2007)
Inhoud is buite die student se vlak van kennis	(Zobel & Hamilton, 2002; Sheard *et al.*, 2003; Devlin & Gray, 2007)
Die relevansie van die aktiwiteit word nie deur die student verstaan nie	(Stevens & Stevens, 1987; Zobel & Hamilton, 2002; Sheard *et al.*, 2003; Devlin & Gray, 2007)
Hardeware, sagteware en toegang tot die biblioteek of onderwyser is onvoldoende	(Zobel & Hamilton, 2002; Sheard *et al.*, 2003; Devlin & Gray, 2007)
"Verbetering" van punte	(Stevens & Stevens, 1987; Davis *et al.*, 1992; Love & Simmons, 1998)
Versoeking en geleentheid	(Davis *et al.*, 1992)

2.4 Hoe om plagiaat te vermy

Een van die vyf kategorieë hoe plagiaat hanteer word, is om dit te vermy (Schoeman & Pieterse, 2004). Plagiaat kan op studentevlak of dosentevlak vermy word.

Vir studente word die reëls en regulasies aangaande plagiaat op die universiteit se webtuiste bekend gemaak waar daar ook wenke gegee word oor hoe om uit plagiaat se kloue te bly. Die Noordwes-Universiteit webtuiste se biblioteekafdeling bevat wenke en brosjures om voorgraadse studente in kennis te stel van hoe om plagiaat te voorkom.

Plagiaat kan voorkom word deur:

- jou eie idees, stem, opinie en werk te gebruik en te ontwikkel;
- iemand anders se idees met matigheid voor oë te gebruik;
- direkte aanhalings wat van belang vir jou navorsing moet wees te minimeer;
- krities, onafhanklik te dink en jou eie gevolgtrekking te maak;
- beplanning en nooit die opdragte te los tot die op die laaste nie; en
- krediet en erkenning te gee deur middel van verwysings en 'n bronnelys (NWU, 2016).

Ober *et al.* (2013) gee vyf eenvoudige reëls om plagiaat te vermy:

1. **Moenie kopieer nie**. Die namaking van woorde vanuit ander artikels of boeke is nie ’n goeie skryfwyse nie. Kort aanhalings met aanhalingstekens en verwysings is aanvaarbaar. Die oorskrywing van woorde net so sonder aanhalings is plagiaat.
2. **Skryf in jou eie woorde.** Skryf al jou idees in jou eie woorde sonder om iemand anders se woorde en skryfstyl te gebruik; dit beteken ook dat parafrasering vermy moet word.
3. **Indien jy twyfel gebruik eerder verwysings.** As jy vind dat daar net verwysing is, beteken dit dat jy nie jou eie woorde gebruik het nie en dus moet daar eerder oorweeg word om die opdrag oor te skryf.
4. **Moenie prente, figure of tabelle hergebruik vanuit jou vorige werk nie**. Dit is beter om nie jou eie figure weer te gebruik nie. Indien daar nie ’n ander uitweg is nie, verwys dan net na die oorpronklike gepubliseerde figuur. Pasop vir self-plagiaat indien jy opvolgartikels skryf.
5. **Vra toestemming.** Wanneer jy ’n figuur, tabel of enige data wil gebruik wat nog nie gepubliseer is nie of deur iemand anders geskep is, moet daar toestemming gevra word om dit te kan gebruik.

Universiteite bied spesiale kontaksessies aan om studente bloot te stel aan die nuwe skryfwyse met verwysings sodat hulle kan leer hoe om op universiteitsvlak opdragte te voltooi. Daar bestaan ook gidse wat gedurende die studies gebruik kan word vir riglyne, soos die brosjure “Moenie met jou toekoms dobbel nie, plagiaat is ’n oortreding” in Bylaag B wat bekend maak wat plagiaat is en hoe om dit tydens die voltooiing van opdragte te verhoed. Soos in Bylaag B gesien kan word, is hierdie riglyne gefokus op geskrewe teksopdragte of navorsing en nie spesifiek op programmeringsopdragte nie.

Die bogenoemde is algemene maniere hoe studente kan verseker dat hulle nie plagiaat pleeg nie en volgens die universiteit se reëls opdragte voltooi. Verder hang dit dan ook van die dosente af om te verseker dat dit nie vir studente maklik is om plagiaat te pleeg nie. Die volgende kan volgens Wilhoit (1994); Zobel en Hamilton (2002) deur dosente in gedagte gehou word om plagiaat onder studente te ontmoedig.

Definieer en bespreek plagiaat deeglik. Dosente van elke studierigting wat opdragte aan studente gee, moet vooraf vir hulle ’n gedrukte verklaring gee wat die definisie en beginsels van plagiaat verskaf saam met voorbeelde en strawwe wat studente in die gesig sal staar indien hulle wel plagiaat pleeg. Elke dissipline se plagiaatverklaring word op die situasies toegepas wat na vore kom op die spesifieke gebied. Hierdie bespreking van plagiaat moet

meer as net een keer per semester plaasvind om sodoende die belang daarvan te beklemtoon.

Bespreek hipotetiese gevalle met studente. Praat met die studente oor die verskillende probleme wat hulle in die uitvoering van die opdrag sal teëkom en hoe om dit op ’n etiese manier op te los. Vertel studente van gevalle waar vorige studente uitgevang is vir plagiaat en die gevolge daarvan.

Die ontwerp van die opdrag. Opdragte moet so ontwerp word dat die studente individualiteit kan vertoon deur ’n unieke oplossing te verskaf. Hierdie opdragte lei tot selfplagiaatopsporing deurdat elke student se oplossing uniek moet wees.

Verifieerbare inhandiging. Waar moontlik moet opdragte so geskep word dat dit elektronies ingehandig word. Hierdie inhandiging bied die voordeel om die opdrag deur beskikbare outomatiese prosesseerders te sit. Indien die opdrag uit programme of invulvrae bestaan wat aanlyn gedoen is, moet studente ook kan bewys lewer dat hulle die opdrag self voltooi het deur ’n rugsteunlêer daarvan te hou.

Skep nuwe opdragte. Opdragte moet soveel moontlik van een jaar tot ’n volgende verskil om te voorkom dat die opdragte net so deur die volgende generasie studente hergebruik kan word. Daar bestaan verskeie webtuistes waar ou opdragte vir die hoogste aanbod verkoop word.

Beperk groepswerk. Groot departemente wat oorlaai is, neig daartoe om opdragte in groepsformaat uit te reik om die oorhoofse aantal opdragte te verminder. Dit help studente om die nodige vaardighede te leer om as ’n span saam te werk, maar swak studente is geneig om weg te bly en lewer dus nie ’n bydrae nie of gebruik dan ander se werk en beweer dat dit hul eie is.

Met al die bogenoemde agtergrond oor plagiaat in die algemeen word plagiaat in programkode volgende bespreek.

2.5 Plagiaat in programkode

Plagiaat in programmeringsopdragte verskil aansienlik van ander vorms van plagiaat. Die kanse dat die programkode in programmeringsopdragte dieselfde is, is hoër want die sintaksis wat gebruik word, is meer beperk as dié van natuurlike taalteks. Dit is ook makliker om programkode te kopieer en die algemene voorkoms van die programkode te verander wat dit

moeiliker maak om deur persoonlike observasies of basiese outomatiese algoritmes op te spoor (Arabyarmohamady *et al.*, 2012; Pieterse, 2014).

Die verskil tussen programkodeplagiaat en dié van natuurlike taalteksplagiaat is dat die metodes wat gebruik word om plagiaat op te spoor verskil (Lancaster & Culwin, 2005). Alhoewel dit makliker is om programkodeplagiaat op te spoor vergeleke met dié van natuurlike taalteksplagiaat, word daar vir programkodeplagiaat ŉ deskundige benodig om te beoordeel of die programkode wat verander is wel geplagieer is. Dit is ook moeilik om ŉ nie-deskundige van programmering te oortuig dat dit wel geplagieer is (Clough, 2003; Chuda *et al.*, 2012; Pieterse, 2014).

Vanuit die bostaande is dit duidelik dat programkodeplagiaat uniek is en anders gedefinieer word as ander vorms van plagiaat en dus word verskeie programkodeplagiaatdefinisies wat vanuit die literatuur verkry is vervolgens bepreek.

2.5.1 Programkodeplagiaat: ŉ definisie

Die verskille tussen universiteitsbeleid, opdragvereistes en persoonlike akademiese voorkeure kan lei tot ŉ verskeidenheid van aannames onder akademici en studente oor wat as programkodeplagiaat beskou word (Cosma & Joy, 2012).

Definisies van programkodeplagiaat wat in die literatuur bestaan, is geskep deur akademici wat geoutomatiseerde programmeringplagiaatherkenners ontwikkel het en probeer het om in hul publikasies plagiaat te definieer.

Parker en Hamblen (1989) definieer programmeringsplagiaat as ŉ program wat geskep word vanaf ŉ ander program deur klein transformasies toe te pas.

Joy en Luck (1999) definieer plagiaat as die onwetende kopiëring van dokumente of programme.

Volgens Jones (2001a), is ŉ program wat geplagieer is ŉ presiese kopie van die oorpronklike of ŉ variasie wat verkry word deur die toepassing van verskeie tekstransformasies.

ŉ Voorgestelde nuwe definisie is verkry uit Cosma en Joy (2008) se aanlynvraelys wat uitgestuur is aan ongeveer 120 akademiese instellings in die VK waarvan slegs 59 response verkry is wat programkodeplagiaat in ŉ akademiese konteks plaas. Hierdie definisie maak ook bekend hoe studente in programmeringsopdragte plagieer. Programkodeplagiaat kom voor in programmeringsopdragte wanneer ŉ student programkode ***hergebruik*** wat deur iemand anders geskryf is en bewustelik of onbewustelik versuim om ***voldoende erkenning*** te gee en

dit dus inhandig as sy/haar eie werk. Dit behels die ***verkryging*** van programkode, met of sonder die toestemming van die oorspronklike skrywer.

Die hoofterm (hergebruik, voldoende erkenning en verkryging) wat in die definisie van Cosma en Joy (2008) voorkom, word vervolgens bepreek. Hierdie bespreking maak ook bekend hoe studente te werk gaan om in programmeringopdragte te plagieer.

2.5.1.1 Hergebruik van programkode

Hergebruik sluit die volgende in (Cosma & Joy, 2008):

- reproduseer / kopiëring van programkode sonder enige veranderings;
- reproduseer / kopiëring van programkode met ’n minimale tot matige aanpassing. Minimale of matige aanpassing is wanneer die opdrag steeds uit gedeeltes bestaan wat deur iemand anders geskryf is;
- die algehele of gedeeltelike omskakeling van iemand anders se programkode na ’n ander programmeringstaal word beskou as plagiaat afhangende van hoe soortgelyk die tale is en die moeite wat die student moes doen om die omskakeling te doen. Die omskakeling word nie beskou as plagiaat wanneer die student idees en inspirasie verkry vanaf ander programme in ander programmeringstaal nie; en
- die outomatiese generering van programkode deur gebruik te maak van kodegenereringsagteware.

Hergebruik van programkode kom baie voor, veral in objekgeoriënteerde programmering en dus moet daar ook op die uitkyk wees vir selfplagiaat. Die veranderings wat aan programkode aangebring word om plagiaat te verdoesel word bespreek in 2.5.2.

2.5.1.2 Verkryging van programkode

Daar bestaan verskeie metodes hoe programkode verkry kan word. Die onderstaande lys is algemene metodes wat gebruik word om programkode te verkry. Verkryging van programkode met of sonder die toestemming van die oorspronklike skrywer sluit in:

- betaal iemand anders om die program of ’n gedeelte daarvan te skryf;
- steel van iemand anders se programkode;
- samewerking met een of meer medestudente om ’n programmeringsopdrag te voltooi, waarin daar spesifiek aangedui word dat dit nie groepswerk is nie. Dit lei daartoe dat almal wat saam aan die opdrag gewerk het dieselfde opdrag inhandig; en

- uitruil van gedeeltes van die programmeringsopdrag tussen studente van verskillende groepe wat dieselfde opdrag ontvang het (Cosma & Joy, 2008).

2.5.1.3 Onvoldoende erkenning van outeurskap van programkode

Onvoldoende erkenning van eienaarskap van programkode sluit in (Cosma & Joy, 2008):

- versuim om te verwys na die bron en die outeur van die programkode, binne die program of in die toepaslike dokumentasie;
- verskaffing van valse verwysings (verwysing wat opgemaak is deur die student of verwysings wat nie ooreenstem nie); en
- verandering van die program se afvoer om voor te gee dat die program werk (*hardcoding*).

Met hierdie drie hoofmetodes hoe studente te werk gaan om oor die algemeen te plagieer is dit ook van belang om ondersoek in te stel na hoe studente te werk gaan om plagiaat in programmeringsopdragte te verdoesel.

2.5.2 Verdoeseling van plagiaat in programmeringsopdragte

Die definisie van Parker en Hamblen (1989) oor programmeringplagiaat (kyk 2.5.1) maak dit duidelik dat ’n program wat geplagieer is, bestaan uit klein transformasies in die programkode. Joy en Luck (1999) het twee algemene transformasiestrategieë geïdentifiseer wat in programkode toegepas kan word om te probeer om programmeringplagiaat te verdoesel. Die twee transformasiestrategieë is die volgende:

- **Leksikale veranderings:** Hierdie veranderings in programkode kan uitgevoer word deur enige geskikte teksskrywer te gebruik en benodig geen kennis van die programmeringstaal nie. Hierdie veranderings kan die volgende insluit: herbewoording, verandering van formatering, byvoeging of weglating van kommentaar en verandering van veranderlikes se name.

- **Strukturele veranderings:** Hierdie veranderings in programkode benodig die gebruiker se kennis van programmering om sodoende die struktuur op ’n manier te verander dat die program steeds uitvoer. Hierdie veranderings is afhanklik van die programmeringstaal. Dit kan die vervanging van ekwivalente iterasiestrukture of die verandering van operatore insluit.

Studente gebruik eenvoudige strategieë om plagiaat te verdoesel, deur leksikale veranderings in programkode toe te pas (Gitchell & Tran, 1999; Luquini & Omar, 2011). Volgens Whale (1990) en Jones (2001a), is die verandering wat studente gebruik om programkodeplagiaat te verdoesel soos volg:

- verandering van die veranderlikes se benamings;
- verandering van die kommentaar;
- verandering van die formatering;
- herformulering en strukturering van ’n blokkode;
- byvoeging of weglating van oorbodige elemente;
- herskrywing van seleksiestelling (*if-* / *case*-stellings);
- herskrywing van die hele program.
- herskrywing van iterasiestellings (*while* / *for*);
- verandering van datatipes;
- verander orde van stellings;
- verandering aan operatore binne uitdrukkings; en
- toevoeging van onnodige stellings of veranderlikes.

Faidhi en Robinson (1987) gaan verder deur hierdie veranderings wat studente gebruik om programkodeplagiaat te verdoesel te verdeel in ses vlakke:

- vlak 1: verandering in kommentaar;
- vlak 2: verandering van vlak 1, asook in veranderlikes;
- vlak 3: verandering van vlak 2, asook in verklarings;
- vlak 4: verandering van vlak 3, asook in programmodules;
- vlak 5: verandering van vlak 4, asook in stellings; en
- vlak 6: verandering van vlak 5, asook in die besluitlogika.

Daar word geen programmeringkennis benodig in die eerste vlak nie, maar goeie programmeringkennis en vaardighede word vir die laaste vlak benodig. Sraka en Kaucic (2009) het in hulle studie gevind dat daar ’n vlak 0, waar geen verandering gemaak is aan die gekopieerde program nie, bygevoeg kan word tot Faidhi en Robinson (1987) se verdelingsvlakke tot programkodeplagiaatverdoeseling.

Cosma en Joy (2006) het ’n volledige lys van moontlike programkode, programdokumentasie en grafiese gebruikerskoppelvlakplagiaatverdoeselingstegnieke opgestel vanuit die tegnieke wat verkry is deur Faidhi en Robinson (1987); Whale (1990); Wise (1996); Joy en Luck (1999);

Prechelt *et al.* (2002) wat alreeds bespreek is, asook persoonlike ervaring wat verkry is deur die studente waar te neem. Hierdie lys word weergegee in Tabelle 2-2 en 2-3.

Tabel 2-2 Lys van moontlike programkodeveranderings om plagiaat te verdoesel (Cosma & Joy, 2006).

Verandering van die programkode:	
Metodeveranderings	Verander die metode se liggaam.
	Herorganiseer funksies.
	Verander die metode se naam.
	Herorganiseer funksies.
	Herorganiseer programkode binne 'n funksie.
	Verandering van die waarde wat die funksie terugstuur.
	Vervang die roep van 'n metode deur die metodeliggaam.
	Verander of verwydering van wysigers van metodes (bv. *public, private*, ens.)
Veranderlike (identifiseerder) veranderings	Verander die tipe van die veranderlikes.
	Verander die name van die veranderlikes.
	Verander die plek waar die veranderlike verklaar word.
	Verander die waardes wat toegeken word aan die veranderlikes.
	Die verspreiding of die saamvoeging van veranderlikes se verklarings.
	Verandering of die verwydering van die wysiger in die veranderlike se verklaring.
	Verander die verklaring van 'n globale veranderlike na 'n plaaslike asook die omgekeerde.
Iterasie- en seleksie-verklaringveranderings	Voeg meer uitvoeringspaaie by.
	Verander stellings na ekwivalente stellings.
	Verander die volgorde van die seleksiestelling.
	Verander voorwaardes na ekwivalente voorwaardes.
	Skakel 'n seleksiestelling om na iets soortgelyks, soos bv. 'n *if*-stelling na 'n *switch*-stelling.

Tabel 2-3 Veranderings in programkode, dokumentasie en in grafiese gebruikerskoppelvlak (GGK) (Cosma & Joy, 2006)

Wiskundige uitdrukkings-veranderings	Kombineer verskeie wiskundige uitdrukkings.
	Verander die volgorde van die operand, soos bv. y < x word x > y.
	Verander die wiskundige uitdrukkings, maar verkry steeds dieselfde afvoer.
Datastruktuur-veranderings	Verander soek- en sorteeralgoritmes.
	Verander datastrukture, soos byvoorbeeld die gebruik van ’n karakterskikking in plaas van ’n stringskikking.
	Die omskakeling van programkode wat gebruik maak van vektore na programkode wat skikkings gebruik.
Klasveranderings	Verander die naam van die klas.
	Verruil metodes tussen klasse.
Funksionaliteits-veranderings	Sluit foute doelbewus in.
	Herstel foute.
	Voeg funksies en programkode by wat nooit geroep word nie.
Programkode-kommentaar-veranderings	Verander die kommentaar se posisie.
	Verspreiding of die saamvoeging van kommentaar.
	Herbewoording of verandering van die kommentaar.
Veranderings in die programkode se dokumentasie:	
Gebruikershandleiding.	
Programspesifikasies.	
Toetsingdokumentasie.	
Tegniese handleidings.	
Dokumentasie wat die programkode vergesel.	
Grafiese gebruikerskoppelvlak (GGK) veranderinge:	
Verander die voorkoms van die afvoer.	
Verander die kleure of skrif van die koppelvlak.	
Verander die grafiese gebruikerskoppelvlak se teks.	
Verander die orde van die toevoer vanaf die gebruiker.	
Versoek ekstra of oortollige toevoer vanaf die gebruiker.	
Verander die uitleg van die grafiese gebruikerskoppelvlak.	
Verander die uitleg van hoe die toevoer ontvang word vanaf die gebruiker.	

Uit die voorafgaande is dit duidelik wat as programmeringsplagiaat beskou word en dat die definisie van Parker en Hamblen (1989) oor programmeringplagiaat voldoende is. Die gevolge van enige vorm van plagiaat word volgende bespreek.

2.6 Die gevolge van plagiaat

Studente sien plagiaat dikwels as ’n geringe oortreding omdat slegs ’n klein aantal studente wat plagieer, opgespoor en vasgevat word (Martin, 1994). Verskeie studies wys daarop dat akademiese personeel en studente se houding teenoor plagiaat en oneerlikheid verskil (Anderson & Obenshain, 1994; Higbee & Thomas, 2000). Dit is dus belangrik dat akademiese instellings ’n beleid vir alle plagiaat opstel met die uiteensetting van gevolge en hoe dit hanteer gaan word. Hierdie beleid se reëls en regulasies moet gedurende die registrasieproses, asook deur die webtuiste van die universiteit aan studente bekend gemaak word. Die optrede rakende plagiaat aan die Noordwes-Universiteit waar hierdie studie uitgevoer word, word weergegee in Bylaag A.

Schoeman en Pieterse (2004) klassifiseer die verskillende maniere hoe plagiaat hanteer kan word in vyf kategorieë, naamlik straf, vermy, voorkom, delegeer en opvoeding. Die straf van plagiaat hang af van die instelling, die wyse waarop plagiaat hanteer word, asook die toename in plagiaatgevalle. Stanford Universiteit het agtergekom dat daar vanaf 1998 ’n 126% toename in intra-institusionele plagiaat was, waarvan die algemene straf skorsing was en 40 uur gemeenskapsdiens. Die Universiteit van Yale se strawwe wissel van teregwysings tot skorsing, terwyl die Universiteit van California probeer om plagiaatgevalle direk tussen die dosent en student op te los deur strawwe wat algemener toegepas word, soos waarskuwingsnotas, gemeenskapsdiens, indiening van ’n verskoningsbrief of addisionele assesseringsaktiwiteite (Maurer *et al.*, 2006).

Akademiese oneerlikheid kan hanteer word op dosentevlak of op instellingsvlak. Die dosente kan dit hanteer deur geskrewe of mondelinge waarskuwings, verandering van punte of deur bykomende opdragte aan die skuldiges te gee. Op instellingsvlak kan dit hanteer word deur middel van ’n ondersoek deur ’n gevestigde komitee waar die beskuldigde deel is van die hele proses wat kan lei tot ’n moontlike verhoor. Indien ’n student skuldig bevind word, kan dit lei tot akademiese integriteitsopleiding, opskorting van module, skorsing, herroeping van graad of sertifikaat en selfs die moontlikheid van geregtelike vervolging (Vamplew & Dermoudy, 2005; Maurer *et al.*, 2006; Liaqat & Ahmad, 2011).

Die institusionele beleid van Manchester Metropolitaanse Universiteit omskryf die strawwe vir plagiaat op voorgraadse vlak soos volg (Manchester Metropolitan University, 2014):

Oortreding gedurende eerste fase (eerste jaar):

- ’n Eerste oortreding met nie meer as 20% plagiaat nie verkry ’n skriftelike waarskuwing en maksimum van 40% vir die opdrag.
- ’n Eerste oortreding en meer as 20% plagiaat verkry ’n skriftelike waarskuwing en ’n punt van 0 vir die opdrag.
- ’n Tweede oortreding in enige eenheid binne dieselfde program verkry ’n verdere waarskuwing en ’n punt van 0 vir die tweede opdrag.
- ’n Derde oortreding in enige eenheid binne dieselfde program beteken dat die eenheid gedruip word.

Oortreding gedurende tweede en derde fases:

- Daar bestaan geen rekord van vorige plagiaatoortreding nie en die plagiaat verteenwoordig nie meer as 20% van die opdrag nie kan ’n maksimum van slegs 40% verkry word vir die opdrag.
- Daar bestaan geen rekord van vorige plagiaatoortreding nie en die plagiaat verteenwoordig meer as 20% van die opdrag verkry ’n punt van 0 vir die opdrag.
- Die student is alreeds gedurende hierdie fase gestraf vir plagiaat en druip dus die fase.

2.7 Geoutomatiseerde plagiaatherkenners

Navorsing aangaande geoutomatiseerde plagiaatherkenners konsentreer op die identifisering van plagiaat en nabootsing van teks. Sedert die 1970s het die rigting en gewildheid van geoutomatiseerde plagiaatherkenners verander. Die empiriese navorsing vir geoutomatiseerde plagiaatherkenners is afkomstig vanaf die programmeringgemeenskap waar rekenaarwetenskaplikes hulpmiddels ontwikkel het om ongewone ooreenkomste tussen programmeringopdragte te identifiseer. Die vroegste geoutomatiseerde plagiaatherkenner het gebruik gemaak van Halstead (1977) se wetenskaplike metingsattributesagteware om die vlak van ooreenkoms tussen programpare te bepaal. Die vroegste stelsel wat gevind word in die literatuur is dié van Ottenstein (1976) wat slegs die basiese Halstead (1977) metingsattribute (aantal unieke operateur η_1, die aantal unieke operant η_2, die totale voorkoms van operateur N_1, die totale voorkoms van operant N_2) gebruik het op Fortran-programme. Die programme word slegs vir plagiaat oorweeg wanneer al vier die metingsattribute saamval. Latere stelsels, soos dié van Donaldson *et al.* (1981); Grier (1981); Berghel en Sallach (1984) en Faidhi en Robinson (1987) het groter metingsattribute bekend gemaak. Werk word nog binne hierdie industrie verrig omdat daar ’n groot belangstelling is in die identifisering van ooreenkomste binne groot sagtewareprogramme. In die akademie het die belangstelling geskuif na die identifisering van plagiaat in natuurlike taalteks met die oog op identifisering

van woordelikse knip en plak *(cut-and-paste)* vanaf webgebaseerdebronne asook konteks van die bron wat geparafraseer is.

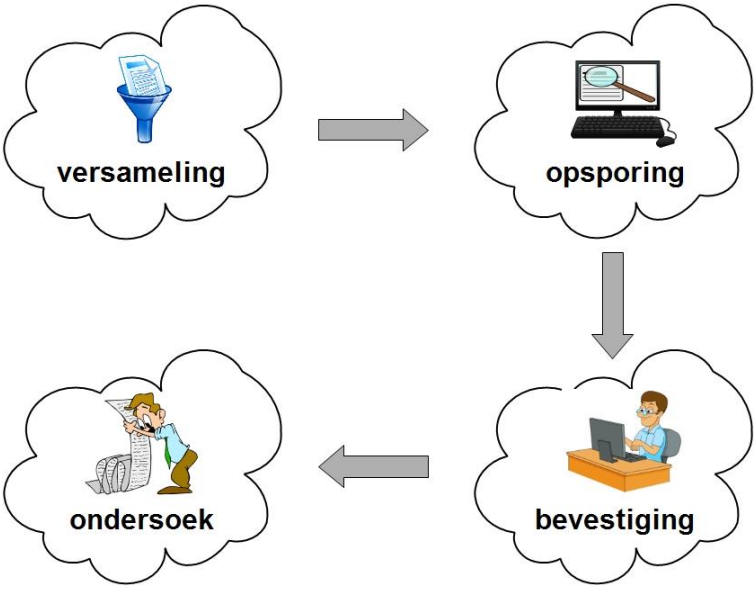
Whale (1990) het voorgestel dat die taak van 'n geoutomatiseerde plagiaatherkenner kan vereenvoudig word deur 'n kenmerkende eienskap soos spelfoute of geparafraseerde kommentaar te vind, maar hierdie toepassing is moeilik om in te bou in 'n geoutomatiseerde plagiaatherkenningstelsel. Om meer komplekse vorms van plagiaat te identifiseer buiten die knip en plak of die eenvoudige herskryf, is moeiliker veral in natuurlike taal, dus maak geoutomatiseerde plagiaatherkenners gebruik van 'n kwantifiseerbare diskrimineerder wat die ooreenkomste van teks meet. Een van die talle uitdagings van geoutomatiseerde plagiaatherkenning is die selektering van die geskikte diskrimineerder wat plagiaat weerspieël en nie soortgelyk is as gevolg van die tema of inhoud nie. Die fokus van geoutomatiseerde plagiaatherkenners is op leksikale en strukturele ooreenkomste in beide programkode en natuurlike taal, maar die doeltreffendheid van die ooreenkomste daal wanneer die herskryfgraad of vorm van plagiaat meer kompleks word (Joy & Luck, 1999; Clough, 2003).

Geoutomatiseerde plagiaatherkenningstelsels is nie eenvormig nie en word verdeel in hermetiese stelsels (aflyn) of webgebaseerde stelsels (aanlyn) waarna dit verder in algemene gebruik, natuurlike taalteks en programkodegeoriënteerde teks verdeel kan word. Die webgebaseerde stelsels probeer ooreenkomste vind vanaf die Internet. Hierdie stelsels is soortgelyk aan 'n gewone Internetsoekenjin. Daar bestaan slegs 'n minimum aantal webgebaseerde plagiaatherkenningstelsels omdat hierdie soort stelsel 'n geweldige aantal berekeningskrag benodig. Die webgebaseerde plagiaatherkenningstelsel moet verseker dat die toeganklike aanlyndokumente wat beskikbaar is onbepaald is om so-doende 'n hoë akkurate herkenning van plagiaat te bied. Sommige van die bestaande webgebaseerde plagiaatherkenningstelsels, soos Turnitin®, maak ook gebruik van dokumente wat intern versamel is. Hermetiese stelsels soek vir gevalle van plagiaat binne 'n versameling van plaaslike dokumente. Hierdie stelsels onderhou hulle eie databasis of maak gebruik van databasisse wat deur iemand anders opgestel en onderhou word. Databasisse bevat byvoorbeeld werk wat deur studente ingegee is en ander hulpbronne oor die kursus en moet gereeld bygewerk word om so te verseker dat die stelsels alle gevalle van plagiaat kan ontbloot (Lancaster & Culwin, 2004; Mozgovoy, 2006; Mozgovoy *et al.*, 2010).

Volgens Clough (2003) en Parker en Hamblen (1989), is die doel van 'n geoutomatiseerde plagiaatherkenningstelsel om die handgedrewe herkenning by te staan deur die hoeveelheid tyd wat spandeer word om die teks te vergelyk te verminder, 'n groot aantal veelvoudige tekste met mekaar te vergelyk en elektroniese bronne te verkry wat moontlik gebruik kan word vir vergelyking van plagiaatherkenning. Die stelsels moet die aantal vals positiewe (verkeerdelik

aangedui as plagiaat) en vals negatiewe (verkeerdelik aangedui as geen plagiaat nie) herkenning minimeer en die aantal ware positiewe (korrek aangedui as plagiaat) en ware negatiewe (korrek aangedui as geen plagiaat nie) herkenning maksimeer. Die uiteindelike doel van geoutomatiseerde plagiaatherkenningstelsels is om plagiaat te verminder.

Om plagiaat op te spoor moet die opdragte deur ŉ proses gaan. Culwin en Lancaster (2001) het ŉ vier-fase plagiaatmodel ontwerp, soos in Figuur 2-1, om te illustreer hoe ŉ outomatiese benadering tot plagiaatherkenning geïmplementeer word. Die geoutomatiseerde plagiaatherkenningstelsel bestaan uit die eerste twee fases van hierdie plagiaatmodel waarna die laaste twee fases met die hand seker maak dat plagiaat korrek herken is deur die stelsel.



Figuur 2-1 Die vier-fase plagiaatherkenningproses (Culwin & Lancaster, 2001)

In die versamelingfase word daar van die studente verwag om hul opdrag in te dien na die stelsel, gewoonlik deur middel van ŉ webenjin wat as koppelvlak dien tussen die student en die stelsel. Volgende, in die opsporingfase, word al die opdragte wat versamel is deur ŉ opsporingsenjin geplaas waarna dit ŉ verslag gee van watter studente se opdragte soortgelyk aan mekaar is. Die bevestigingfase se doel is om seker te maak dat die ooreenkoms wat berig is in die opsporingfase wel plagiaat verteenwoordig deur gebruik te maak van handbevestiging. Stelsels kan verkeerdelik verslagdoening gee oor die ooreenkomste wat verkry is. Daar bestaan ooreenkomste waar twee persone op die regte wyse dieselfde bron aangehaal het. Enige verdere ooreenkomste wat bestaan, word aangestuur na die ondersoekfase waar verdere ondersoek ingestel word wat kan lei tot penalisasie (Culwin & Lancaster, 2001).

2.8 Klassifisering van plagiaatherkenningstelsels

Plagiaatherkenningstelsels besit elkeen unieke eienskappe wat gebruik kan word om hulle in verskillende klasse te verdeel. Tabel 2-4 hier onder bevat ŉ uiteensetting van die verskillende klasse waarvolgens Lancaster (2003) plagiaatherkenningstelsels verdeel het. ŉ Bespreking van die klassifisering van PPHS volg.

Tabel 2-4 Die klassifisering van plagiaatherkenningstelsels volgens Lancaster (2003)

Klassifiseer deur	Omskrywing van klassifikasie
Inhandigingstipe	Die klassifisering van indieningstipe kan gesien word as een van die belangrike maniere om te onderskei tussen plagiaatherkenningstelsels. Die indieningstipe word hoofsaaklik verdeel tussen **programkode** en **natuurlike taalteks**. Programkodeplagiaatherkenners vind ooreenkomste tussen studente se programme wat ingehandig word en natuurlike taalteksplagiaatherkenners vind ooreenkomste tussen studente se natuurlike taalteksopdragte wat ingehandig word. Die verskillende tipes van indiening wat na vore kan kom, is onbeperk. Ander voorbeelde sluit in diagramme of musiek waarvan af plagiaat gepleeg kan word.
Eienskappe	Algemener klassifisering word hier gebruik. Dit kan behulpsaam wees om te weet of die plagiaatherkenningstelsels webgebaseerd of plaaslik is (kyk 2.8), asook watter beskikbaar is vir akademiese gebruik. As die stelsels beskikbaar is vir almal staan dit bekend as **publiek**. Sommige stelsels kan bekom word deur ŉ spesiale reëling te tref en ander is slegs in **privaat** beskikbaar tot die plaaslike instelling. Die stelsels wat nie meer gebruik word nie of vervang is, kan geklassifiseer word as die **gewese**. PPHS'e kan verder geklassifiseer word volgens die programmeringstaal waarin hulle plagiaat kan ontbloot, soos byvoorbeeld Java, C, Prolog en C#. Lancaster en Culwin (2004) het ŉ tabel opgestel wat PPHS'e klassifiseer volgens programmeringstaal.

Tabel 2-4 Die klassifisering van plagiaatherkenningstelsels volgens Lancaster (2003) (vervolg)

Klassifiseer deur	Omskrywing van klassifikasie
Die aantal indienings wat verwerk moet word deur die verskeie metings	’n Moontlike metode vir die klassifisering van verskillende maatstawwe is om die aantal voorleggings wat nodig is om ooreenkomste te verkry tussen die indienings in ag te neem. ’n **Enkele meting** word toegepas op ’n opdrag en genereer ’n eenvoudige getal wat ’n eienskap van hierdie opdrag verteenwoordig. Voorbeelde van enkel meting vir programkode sluit in: die gemiddelde aantal karakters per lyn; die aantal lyne wat kommentaar bevat; die gemiddelde aantal lyne per subprogram. **Gepaarde meting** word toegepas op twee opdragte en verskaf die gelykstaande ooreenkomste tussen die twee opdragte. Die ooreenkomste kan verkry word deur ’n enkele meting op albei opdragte toe te pas en die resultate met mekaar te vergelyk. Voorbeelde van gepaarde maatstawwe vir programkode sluit in: die aantal sleutelwoorde wat in beide programkodes voorkom; die aantal funksies wat in beide programme voorkom met dieselfde aantal lyne. **Multidimensionele meting** kan gebruik word om plagiaat in opdragte van gegroepeerde groepe van meer as twee studente op te spoor. Alhoewel daar geen voorbeelde van hierdie meting in die literatuur is nie moet daar maatstawwe wees wat toegepas kan word op drie of meer opdragte (n-dimensionele maatstawwe). **Korporale meting** bestaan uit multidimensionele maatstawwe wat gebruik word om een groep (korpus) se opdragte wat in-gedien is te vergelyk met ’n ander groep (korpus). ’n Voorbeeld van korporale meting vir programkodeopdragte sal die verhouding wees van al die opdragte wat die sleutelwoord “*while*” gebruik.

Tabel 2-4 Die klassifisering van plagiaatherkenningstelsels volgens Lancaster (2003) (vervolg)

Klassifiseer deur	Omskrywing van klassifikasie
Die operasionele kompleksiteit van die metings wat gebruik word	Verdere klassifikasie kan uitgevoer word op die kompleksiteit van die bewerking wat nodig is om plagiaat te herken. **Oppervlakkige meting** meet die ooreenkomste wat verkry kan word deur net na die opdragte te kyk. Geen kennis van die struktuur van die programmeringstaal of taalkundige eienskappe van die natuurlike taal is nodig nie. ’n Voorbeeld hiervan is om telling te hou van die gereserveerde term “*while*” in programkodeopdragte. **Strukturele meting** benodig kennis van die struktuur van die opdrag om ooreenkomste tussen opdragte te maak. ’n Voorbeeld hiervan is om te kyk na die langste opeenvolgende identiese woorde wat verkry is uit twee opdragte.

Die verskillende klassifikasies soos in Tabel 2-4 aangedui, is van toepassing op programkode- en natuurlike taalteksplagiaatherkenningstelsels. Huidige literatuur aangaande PPHS’e dui ’n verder klassifisering van die stelsels aan. Die stelsels kan in twee groepe verdeel word, naamlik **attribuut-tellingstelsels** en **struktuurmetingstelsels**, maar hierdie klassifisering word nie konstant toegepas nie. Die klassifisering is beperk. Daar bestaan herkenningstelsels wat nie in een van hierdie twee klassifikasies val nie. Die twee klassifikasies is uiteengesit voordat plagiaatherkenningstelsels vir natuurlike taalteks ontwikkel is en is dus nie van toepassing op daardie stelsels nie.

Verco en Wise (1996a; 1996b) het die PPHS’e verdeel volgens algemene klassifikasies, attribuut-tellingstelsels en struktuurmetingstelsel. Die **attribuut-tellingstelsels** word voorgestel deur stelsels wat een of ander eienskap van ’n individuele stelsel meet. Dit staan bekend as die telling van ’n attribuut, soos byvoorbeeld die aantal operatore wat in ’n program voorkom. Die **struktuurmetingstelsel** word voorgestel deur ’n stelsel wat soek na enige ooreenkomste in die strukturele aspekte van die programkode. Hierdie stelsel kan gesien word as ’n stelsel wat gebruik maak van die gepaarde of enkele maatstawwe, soos verduidelik in Tabel 2-4. Culwin *et al.* (2001) verskaf ook ongeveer dieselfde definisies vir hierdie twee klasse.

Verco en Wise (1996a; 1996b) en Jone (2001a; 2001b) het gevind dat hulle eie YAP3-herkenningstelsel gebruik maak van die struktuurmeting. Dit tel nie attribute nie, daarom word die stelsel geklassifiseer as 'n struktuurmetingstelsel. Hierdie stelsel kan nie onderskei word van ander stelsels wat geklassifiseer word as 'n struktuurmetingstelsel nie. Dalk is hierdie klassifikasie slegs handig vanuit 'n kronologiese oogpunt. Jones (2001a; 2001b) sukkel ook om te onderskei tussen hierdie twee klassifikasies deurdat sy attribuut-tellingstelsel op dieselfde beginsels werk as dié van YAP3. Hieruit is dit duidelik dat daar ook verwarring onder ontwikkelaars van hierdie stelsels bestaan.

Daar bestaan 'n benadering tot PPHS'e wat nie in een van die twee klasse van Verco en Wise (1996a; 1996b) val nie. 'n Voorbeeld hiervan is die Saxon-herkenningstelsel wat ooreenkomste verkry gebaseer op die saampersbaarheid van die programkode (Saxon, 2000). Hierdie stelsel gebruik nie strukturele metrieke of attribuut-telling nie, daarom kan dit nie onder een van die twee klassifikasies van Verco en Wise (1996a; 1996b) verdeel word nie. Die stelsel maak gebruik van gepaarde metings, aangesien daar berekeninge op twee voorleggings toegepas word.

'n Moontlike oplossing vir hierdie probleem is om die klassifikasies konstant te hou en nuwe klasse vir daardie stelsels wat nie deur die klasse geklassifiseer kan word nie te ontwikkel. Die stelsels wat gebruik maak van beide attribuut-telling en strukturele metriektelling word volgens Lancaster (2003) na verwys as **hibriede metriekstelsels.** Om die klassifikasies konstant te hou, word stelsels wat gebruik maak van attribuut-telling-maatstawwe, al is daar kode vervang deur spesifieke kentekens (*tokenization*), geklassifiseer as 'n attribuut-tellingstelsel. Op die selfde wyse word stelsels wat gebruik maak van struktuurmeting, al is die kode nie vervang deur kentekens nie (*untokenized*), geklassifiseer as 'n struktuurmetingstelsel.

Lancaster (2003) het verder gevind dat hierdie moontlike oplossing tot klassifisering van PPHS'e steeds nie voldoende is nie. Die klassifikasies is onvas, verskaf nie 'n maklike manier om tussen hulle te onderskei nie en word nie gesien as die beste manier om nuwe stelsels te klassifiseer nie, aangesien meer inligting benodig word as wat verkry kan word vanuit hierdie drie klasse. Dit maak dus meer sin om die stelsels te klassifiseer volgens die maatstawwe wat uiteen gesit is in Tabel 2-4 hier bo.

2.9 Benaderings tot geoutomatiseerde programkodeplagiaatherkenning-stelsels

In 2.8 waar geoutomatiseerde plagiaatherkenners bespreek word, is daar uiteen-gesit dat die geoutomatiseerde plagiaatherkenningstelsels verdeel kan word in hermetiese stelsels (aflyn) of webgebaseerde stelsels (aanlyn). Vir die doel van hierdie navorsing word daar gefokus op hermetiese stelsels. Hierdie afdeling bespreek PPHS volgens die kategorieë, vingerafdrukgebaseerde benadering en konteksvergelykings-tegnieke wat deur Mozgovoy (2006) geïdentifiseer is.

2.9.1 Vingerafdrukgebaseerde benadering

Die idee agter die vingerafdrukgebaseerde benadering is om ŉ vingerafdruk te skep vir al die dokumente in die versameling. Elke vingerafdruk kan bestaan uit verskeie numeriese attribute wat die struktuur van die dokument uitbeeld. Voorbeelde van attribute wat gebruik kan word vir vingerafdrukke is die gemiddelde aantal woorde per lyn en die aantal unieke woorde. Indien twee vingerafdrukke na aan mekaar is (volgens ŉ gegewe kriteria wat voorsien word as ŉ wiskundige afstandsfunksie) word die twee dokumente as dieselfde beskou (Mozgovoy, 2006).

Die vingerafdrukgebaseerde benadering word gebruik in attribuut-tellingstelsels wat grootliks bestaan het uit plagiaatherkenningstelsels van die verlede (Grier, 1981; Faidhi & Robinson, 1987). Een van die eerste projekte wat gebruik gemaak het van die vingerafdruk-benadering was Ottenstein (1976) wat ooreenkomste verkry het, soos bespreek in 2.8. Oor die jare is verskeie ander maatstawwe getoets en meer gevorderde stelsels ontwerp, maar met die tyd het die vingerafdrukgebaseerde benadering verswak deurdat enige klein teksverandering die dokument se vingerafdruk beïnvloed het en veroorsaak het dat plagiaatherkenning ontduik word. Nuwer stelsels volg dus nie meer hierdie benadering nie en is dus eerder gebaseer op konteksvergelyking (Verco & Wise, 1996a).

Die *Accuse*-stelsel (Grier, 1981) is ŉ voorbeeld van die vingerafdrukgebaseerde benadering wat verskeie parameters insluit, naamlik:

- die aantal unieke operateurs;
- die aantal unieke operante;
- die totale aantal operateurs;
- die totale aantal operante.
- die aantal lyne wat bestaan uit kode;

- die aantal veranderlikes wat verklaar is en gebruik word; en
- die totale aantal beheerde stellings.

Die korrelasieskema bereken ’n inkrement vir elke attribuutpaar:

A_1 = attribuut$_i$ getal in die eerste lêer

A_2 = attribuut$_i$ getal in die tweede lêer

Inkrement$_i$ = belangrike attribuut$_i$ – (A_1 – A_2)

Hierdie inkremente lewer dan die finale ooreenkomsverhouding tussen die twee lêers op. Vingerafdrukke in meer moderne stelsels bestaan uit waardes wat verkry is deur ’n wiskundige funksie (bv. treffunksie) toe te pas op geselekteerde onderafdelings van ’n versameling van lêers (Hoad & Zobel, 2003). Daar bestaan projekte wat die vingerafdruk-benadering kombineer met konteksvergelykingstegnieke, die bekende MOSS (“*Measure of Software Similarity”*) stelsel maak gebruik van so ’n benadering (Schleimer *et al.*, 2003). Ander vingerafdrukbenaderings sluit in die gebruik van algemene teksverkrygingstegnieke, soos “*latent semantic analysis*” (LSA) en vektorspasiemodel (Nakov, 2000; Stein & Zu Eissen, 2006).

Die MOSS-stelsel is gebaseer op ’n stringpassingsalgoritme wat die programme opdeel in k-gram, waar elke k-gram ’n aangrensende substring van lengte k is. Elke k-deel word deur die treffunksie geplaas waarna die stelsel ’n gedeelte van hierdie waardes kies as die vingerafdrukke van die program. Ooreenkomste word bepaal deur die aantal vingerafdrukke wat twee programme gemeen het; hoe meer hulle gemeen het, hoe meer is hulle dieselfde. Vir elke programkodefragment wat opgespoor is as dieselfde word die resultate as ’n opsomming gegee deur die aantal tekens en lyne wat ooreenstem, asook die persentasie van hoeveel van die programkode tussen die twee programme wat ondersoek word, oorvleuel (Schleimer *et al.*, 2003; Aiken, 2005).

2.9.2 Konteksvergelykingstegnieke

Indien die vingerafdrukvergelyking nie die plagiaat opspoor nie, kan die konteks van die dokumente vergelyk word deur verskeie tegnieke toe te pas waarvan elkeen sy eie voordele en nadele het. Daar word in die literatuur na die konteksvergelykingstegnieke as struktuurmetingstelsels verwys en hulle word deur Mozgovoy (2006) verdeel volgens hul toepassingsalgoritmes as stringpassingsalgoritme, parameterpassingsalgoritme en verdelingsboomvergelykingsalgoritme. Hierdie drie algoritmes van konteksvergelyking word in hierdie afdeling bespreek.

2.9.2.1 Stringpassingsalgoritme

Die hooffunksie wat die ooreenkomste bereken, verskil van een stelsel na ŉ ander. Met stringooreenkomste word die dokumente hanteer as ŉ stel stringe en hulle word daarvolgens vergelyk. Hierdie benaderings hou nie die hiërargiese struktuur van die programkode in gedagte nie; dit word beskou as rou data. Stringpassing-gebaseerde stelsels gebruik meer kompleks ooreenkomsalgoritmes as dié van attribuut-tellingstelsels. Die meerderheid van die stringpassingsalgoritmes gaan te werk deur die programkode om te skakel in merktekens waarna gesofistikeerde soekalgoritmes gebruik word om gemeenskaplike substringe vanuit hierdie merktekens te verkry (Cosma & Joy, 2012).

Die program wat deur Donaldson *et al.* (1981) geskep is, gaan deur die programkodelêers en stoor inligting aangaande sekere tipes stellings waarna ŉ enkel kodekarakter toegeken word aan ŉ unieke struktuur vanuit hierdie stellingtipes. Elke opdrag word dus as ŉ string karakters voorgestel en dié word dan met mekaar vergelyk om vas te stel of die opdragte identies of dieselfde is.

Na Donaldson *et al.* (1981) het stringgebaseerde plagiaatherkenningstelsels, soos YAP, gebruik gemaak van ŉ eenvoudige lyn-vir-lynvergelyking tussen twee lêers deur die Levenshtein (1966) afstandmodel. Deur die jare is meer gevorderde stringpassingmetodes in plagiaatherkenningstelsels ontwikkel en geïmplementeer. Die "*Running-Karp-Rabin Greedy-String-Tiling*" algoritme (RKR-GST) is van die gewilde lêervergelykingsmetodes wat in verskeie stelsels, insluitende deur YAP3 ("*Yet Another Plague"*) (Wise, 1996), JPlag (Prechelt *et al.*, 2002) en Plaggie (Ahtiainen *et al.*, 2006) geïmplementeer word.

Die **YAP3**-stelsel skakel programkodeopdragte wat in die stelsel ingelaai is om in ŉ string merktekens waarna dit met mekaar vergelyk word deur gebruik te maak van die RKR-GST-algoritme om sodoende ooreenkomste te verkry uit die programkodesegmente (Wise, 1996). Die stelsel doen vooraf verwerking aan die programkodeopdragte voordat dit omgeskakel word na merktekens toe. Die voorafverwerking sluit in die verwydering van kommentaar, verander hoofletters na kleinletters, herorganiseer die funksies na hulle roependе vorm toe, en verwyder alle merktekens wat nie deel is van die woordeskat van die doeltaal nie (m.a.w. verwyder al die woorde wat nie gereserveer word deur die taal nie). Die *Running-Karp-Rabin Greedy-String-Tiling*-algoritme kan soos volg uiteengesit word. Die algoritme begin deur die ooreenkomste van die aanvanklike soektog lengte en grootte te analiseer. Hierdie ooreenkomste word verkry deur die *Karp-Rabin* prosedure toe te pas (Karp & Rabin, 1987). Hierna word die ooreenkomste vanaf die langste geanaliseer en as die huidige ooreenkoms nie oorvleuel met die bestaande teëls nie, word dit bygevoeg as ŉ nuwe teël. Na al die

ooreenkomste probeer is, word ’n nuwe soektog geskep met ’n kleiner ooreenkomslengte. Die algoritme is klaar wanneer die minimale ooreenkomslengte bereik is (Mozgovoy, 2006) .

Die **JPlag**-stelsel maak gebruik van dieselfde vergelykingalgoritme (RKR-GST) as die YAP3-stelsel, maar die bewerkingstyd word geoptimeer. Die stelsel het ’n algemene syfer wat toegeken word aan die minimale ooreenkomslengte vir die algoritme. As hierdie syfer verander word, veroorsaak dit dat die resultate varieer van goed na sleg en dus benodig die gebruiker vooraf kennis van die algoritme. JPlag word geïmplementeer as ’n webdiens en bestaan uit ’n eenvoudige gebruikerskoppelvlak (Prechelt *et al.*, 2002).

Plaggie is ’n soortgelyke stelsel as JPlag, maar maak nie gebruik van ’n optimeringsalgoritme nie. Die stelsel verander die lêers na merktekens toe en maak gebruik van die RKR-GST-algoritme om ooreenkomste tussen lêers te vind. Die idee agter die stelsel is om as ’n losstaande hulpmiddel te dien eerder as ’n webgebaseerde hulpmiddel en om die akademici die vermoë te bied om sommige programkode uit die vergelyking te sluit (Ahtiainen *et al.*, 2006).

2.9.2.2 Parameterpassingsalgoritme

Die parameterpassingsalgoritme is naastenby dieselfde as die gewone kentekenalgoritme (*Tokenization*). In die parameterpassingsalgoritme vind daar net meer gevorderde substitusie plaas. Eerstens vervang die gewone kentekenalgoritme verskeie elemente van die programkode met ’n enkel kenteken, soos byvoorbeeld, enige identifiseerder kan vervang word deur die kenteken `<IDD>` en elke numeriese waarde deur die kenteken `<WAARDE>`. As die programkode bestaan uit `c = e + 70;` word dit vervang met `<IDD> = <IDD> + <WAARDE>`; Dus as die veranderlikes se name verander is, maak dit nie ’n verskil nie, aangesien die programkode verander is na dieselfde gekentekende volgorde. In die parameterpassingsalgoritme word die lêers eers omgeskakel na merktekens waarna identiese afdelings van programkode, asook sistematiese substitusie van die identifiseerder gevind word (Mozgovoy, 2006).

Die **Dup**-hulpmiddel is gebaseer op die parameterpassingsalgoritme en is aanvanklik ontwikkel om duplikate in programkode op te spoor. Dup kan identiese en parameter-afdelings van programkode identifiseer. Die hulpmiddel gebruik ’n leksikale ontleder om die programkode te skandeer waarna elke lyn programkode getransformeer word in parameters deur die identifiseerder en konstantes te verander na dieselfde simbool P met ’n lys wat rekord hou van die veranderings, soos byvoorbeeld, die lyn `y = fun(x)+3*x` word getransformeer na `P = P(P)+P*P` met ’n lys wat bestaan uit x,fun,y,3. Die leksikale ontleder skep ’n heelgetal

wat elke transformasie van die programkode voorstel. Die eendersheid word bepaal deur die persentasie parameterpassing wat gevind is tussen twee lêers. Die Dup-hulpmiddel se afvoer vertoon die persentasie-ooreenkomste tussen twee lêers, 'n profiel wat die parameterpassingsprogramkodelyne verskaf van die twee lêers en die ligging van die ooreenkomste van elke lyn (Baker, 1995a). Daar bestaan verskeie ander parameterpassingsalgoritmes in die literatuur (Amir *et al.*, 1994; Baker, 1995b; Idury & Schäffer, 1996; Fredriksson & Mozgovoy, 2006).

2.9.2.3 Verdelingsboomvergelykende algoritme

Die verdelingsboomvergelykende algoritme implementeer struktuurvergelykingstegnieke. Die idee om 'n verdelingsboom van programkode te gebruik is eerste deur die **Sim**-hulpmiddel geïmplementeer (Gitchell & Tran, 1999). Die Sim-hulpmiddel stel elke lêer voor as 'n verdelingsboom en maak gebruik van stringvertoë van die verdelingsbome om die lêers te vergelyk vir ooreenkomste. Die vergelykingsproses begin deur elke programkodelêer te verander na 'n string kentekens waarna elke kenteken vervang word deur 'n vasgestelde stel kentekens. Na die kentekening word die kentekenstrome van die lêers opgedeel in afdelings waarna die afdelings belyn word. Hierdie tegniek maak dit moontlik om geskommelde programkodefragmente op te spoor.

Sim maak gebruik van gewone stringpassingsalgoritme om die lêers wat voorgestel word as 'n verdelingsboom te vergelyk. Soos met die meeste stringpassingsalgoritmes soek Sim vir die maksimum opeenvolgende merktekens en gee die graad van ooreenkomste wat gevind is tussen die twee lêers as afvoer. Sim is oorspronklik ontwikkel om ooreenkomste tussen C-programkodes op te spoor, maar kan maklik verander word sodat dit toegepas kan word op ander programmeringstale. Die hulpmiddel kan soortgelyke lêers wat algemene veranderings, soos name, herorganiseer van stellings en funksies, en die byvoeging of verwydering van kommentaar opspoor (Gitchell & Tran, 1999).

Die **Brass**-stelsel is 'n meer onlangse toepassing van die verdelingsboomvergelykende algoritme. Die Brass-stelsel stel elke program as 'n struktuurgrafiek voor wat bestaan uit die verdelingsbome van elke lêer. Die wortel van die boom spesifiseer die kop van die lêer en die kindnodes spesifiseer die stellings en beheerstrukture wat in die lêer gevind word. Die boom word dan omgeskakel na 'n binêre boom en 'n simbooltabel (data-woordeboek) word geskep, wat die inligting aangaande die veranderlikes en datastrukture wat in die lêer voorkom bevat. Die Brass-stelsel se vergelykingsproses bestaan uit drie dele. Die eerste twee algoritmes vergelyk die boomstrukture van die lêers waarna die derde algoritme die datawoordeboekvoorstellings van die lêers vergelyk. Soortgelyke lêers word dus opgespoor

in die eerste vergelykingsalgoritme waarna die tweede en derde algoritmes toegepas word op hierdie lêers. Aangesien die boomvergelyking gebruik maak van ’n meer komplekse algoritme as die stringvergelykingsalgoritme, word daar aangeneem dat dit stadiger is, daarom het die Brass-stelsel ’n filtertegniek op die lêers geïmplementeer om die vergelykingsproses te versnel (Belkhouche *et al.*, 2004).

Volgens (Mozgovoy, 2006), is navorsing skraal op die gebied van verdelingsboomalgoritmes en dus is dit nog nie bekend of hierdie algoritme ’n verbetering is op die stringpassingsalgoritme om ooreenkomste tussen programkodes te verkry nie.

Die inligting wat verkry was vanuit die literatuurstudie word benut in Hoofstuk 3 en 4. Hoofstuk 3 vervolg wat beskikbare programkodeplagiaatherkenningstelsels evalueer.

HOOFSTUK 3
EVALUERING VAN PROGRAMKODEPLAGIAATHERKENNINGSTELSELS

In hierdie hoofstuk word ses programkodeplagiaatherkenningstelsels (PPHS) geëvalueer en bespreek. Die stelsels word geëvalueer deur twaalf C# programmeringsopdragte te gebruik wat verander word deur die aanwending van Faidhi en Robinson (1987) se ses vlakke van programkodeplagiaatverdoeseling as 'n riglyn.

3.1 Inleiding

Faidhi en Robinson (1987) het ses vlakke geïdentifiseer van hoe studente te werk gaan om programkodeplagiaat te verdoesel:

- vlak 1: verandering in kommentaar;
- vlak 2: verandering van vlak 1, asook in veranderlikes;
- vlak 3: verandering van vlak 2, asook in verklarings;
- vlak 4: verandering van vlak 3, asook in programmodules;
- vlak 5: verandering van vlak 4, asook in stellings; en
- vlak 6: verandering van vlak 5, asook in die besluitlogika.

Hierdie ses vlakke word as riglyne vir die studie gebruik om die programkodeplagiaatherkenningstelsels (PPHS) te evalueer. Die volgende tipes verandering gaan in hierdie studie gebruik word om die PPHS'e te evalueer:

- tipe verandering 1: verandering van kommentaar;
- tipe verandering 2: verandering van veranderlikes;
- tipe verandering 3: verandering van verklarings;
- tipe verandering 4: verandering van programmodules;
- tipe verandering 5: verandering van stellings; en
- tipe verandering 6: verandering van besluitlogika.

Aangesien die studie fokus op grafiese gebruikerskoppelvlakke (GGK), gaan addisionele veranderings wat nie in die vlakke van Faidhi en Robinson (1987) voorkom nie, bygevoeg word om die verandering wat aan die GGK gebring word om plagiaat te verdoesel aan te spreek:

- tipe verandering 7: verandering van die GGK se uitleg;

- tipe verandering 8: verandering van die GGK se kontroles se name;
- tipe verandering 9: verandering van die GGK se teks; en
- tipe verandering 10: verandering van die GGK se uitleg en die name van die kontroles.

Die prosedure wat in hierdie studie gevolg is, is soos volg:

1. Twaalf programmeringsopdragte is geselekteer wat gedien het as kontroles vir die studie en het die volgende bevat:
    a. Kommentaar;
    b. *If*-steling*;* en
    c. *For*-stelling*;*

2. Die programmeringsopdragte is gekopieer waarna daar veranderinge aangebring is volgens die tien tipes verandering (kyk Tabel 3-1 en Tabel 3-2 vir die veranderinge wat aangebring is). Altesame 120 veranderde programmeringsopdragte is dus geskep.

3. Die programkodeplagiaatherkenningstelsels (kyk Tabel 3-3) wat geselekteer is, moes aan die volgende vereistes voldoen:
    a. ondersteun die programmeringstaal C#; en
    b. is gratis beskikbaar.

4. Die veranderde programmeringsopdragte (VPO) is een vir een vergelyk met die ooreenstemmende kontroles (KON) deur die geselekteerde plagiaatherkenningstelsel. Die vergelyking tussen VPO en KON is op twee maniere uitgevoer:
    a. Die kodelêer (Form1.cs) van VPO word vergelyk met dié van KON. Wanneer die grafiese gebruikerskoppelvlak vergelyk word (veranderings 7 tot 10), word die ontwerplêer (Form1.Designer.cs) beskou as die kodelêer. Dit wil sê: met tipes verandering 1 tot 6 word al die Form1.cs lêers van VPO met dié van KON vergelyk en met tipes verandering 7 tot 10 word die Form1.Designer.cs lêers van VPO met dié van KON vergelyk.
    b. Die kodelêer en die ontwerplêer van VPO word vergelyk met dié van KON. Dit wil sê: in tipes verandering 1 tot 10 word al die Form1.cs en From1.Designer.cs lêers van VPO met dié van KON vergelyk.

5. Die afvoer wat in punt 4 deur die PPHS'e verkry is, is gebruik om 'n gemiddelde ooreenkomspersentasie vir die tipe verandering te bereken en word vervolgens

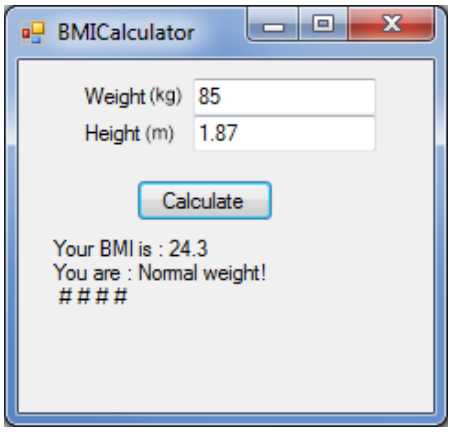
bespreek. Die afvoer wat vir elke programmeringsopdrag deur die PPHS verkry is, word in Bylaag P weergegee.

3.2 Die programmeringsopdrag

Die twaalf C# programmeringsopdragte wat vir hierdie studie gebruik word se doel is om die gebruikers se lengte (cm) en gewig (kg) as toevoer vanaf die gebruiker te verkry. Die toevoer word dan gebruik om die gebruiker se liggaamsmassa-indeks (LMI) te bereken en die volgende afvoer te lewer:

- **ondergewig** as die LMI kleiner as of gelyk aan 18.5 is;
- **normaal** as die LMI groter as 18.5 is en kleiner as 25 is;
- **oorgewig** as die LMI groter as of gelyk aan 25 is en kleiner as 30 is; of
- **vetsugtig** as die LMI groter as of gelyk aan 30 is.

Saam met hierdie afvoer, word daar ook hutstekens (#) vertoon, gebaseer op die LMI. Die aantal hutstekens wat uitgedruk word, word bereken deur die LMI te deel deur vyf. 'n Voorbeeld van die program se grafiese gebruikerskoppelvlak wat die toevoer ontvang en afvoer vertoon, word in Figuur 3-1 weergegee.

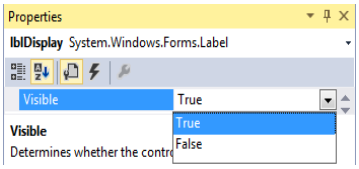


Figuur 3-1 'n Voorbeeld van 'n basiese grafiese gebruikerskoppelvlak van programmeringsopdrag

3.3 Die tipes verandering

Soos bespreek in Afdeling 3.1, word daar tien tipes verandering aan die programmeringsopdragte vir hierdie studie gemaak. Tabel 3-1 verskaf 'n beskrywing en 'n voorbeeld van ses tipes verandering wat aan die programmeringsopdragte se programlêer gemaak word. Tabel 3-2 verskaf 'n beskrywing en 'n voorbeeld van vier tipes verandering wat aan die GGK van die programmeringsopdragte aangebring word.

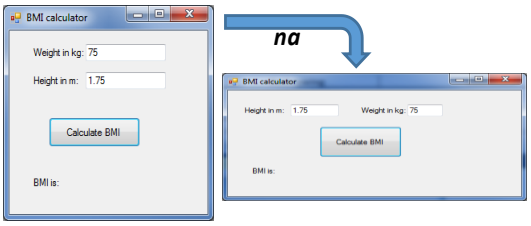
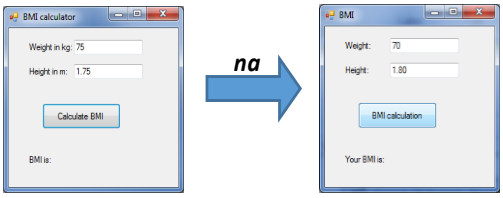
Tabel 3-1 Veranderings wat aan die programmeringsopdragte gemaak is

Tipe	Verandering	Beskrywing	Voorbeeld
1	Kommentaar	Verander deur vertaling uit Afrikaans in Engels of vanaf Engels na Afrikaans of deur die sintaks van die kommentaar te verander.	`//Roep die GetBMI metode` na `//Call the GetBMI method`
	Inkeping	Spasiëring van die die programmeringskode te verander deur die tabelsleutel (*Tab*) te gebruik.	`{ Display(BMITest); }` na `{ Display(BMITest); }`
2	Veranderlikes	Verander die benaming van die veranderlikes.	`int weight;` na `int w;`
3	Verklarende stellings	Skuif programmeringskodeblokke rond.	`{ Blokkode 1 } { Blokkode 2 } { Blokkode 3 }` na `{ Blokkode 3 } { Blokkode 1 } { Blokkode 2 }`
		Verander die volgorde van die parameters wat ontvang word deur metodes.	`KryGetalle(out Height, out Weight);` na `KryGetalle(out Weight, out Height);`
		Verander veranderlikes se volgorde per programkodelyn.	`double Height, Weight, BMI;` na `double Weight, BMI, Height;`
		Verskuil die etiketkontrole deur middel van 'n stelling in die eienskapblad.	
		Kode om die etiketkontrole weer te vertoon word bygevoeg in die programmeringskode.	`lblDisplay.Show();`
4	Programmodules	Die metode wat die hutsteken (#) afvoer geneereer volgens die LMI word geskuif en binne 'n ander metode geplaas.	`private string Sterre(double sBMI) { string stt = ""; int nom = (int)sBMI/5; for (int l = 1; l <= nom; l++) stt = stt + " #"; return stt; }` na `private void btnCalc_Click(object sender, EventArgs e) { int weight; double fBMI, height; string stt = ""; GetWeightHeight(out weight, out height); fBMI = CalculateBMI(weight, height); int nom = (int)fBMI / 5; for (int l = 1; l <= nom; l++) stt = stt + " #"; DisplayBMI(fBMI,stt); }`

Tabel 3-1 Veranderings wat aan die programmeringsopdragte gemaak is (vervolg)

Tipe	Verandering	Beskrywing	Voorbeeld
5	Stellings	Die *if* stelling se berekeningslogika word verander.	`if (fBMI < 18.5) bmiState = "Underwight"; else if (fBMI < 25) bmiState = "in Normal Range"; else if (fBMI < 30) bmiState = "Overweight"; else bmiState = "Obese";` **na** `if (fBMI > 30) bmiState = "Obese"; else if (fBMI > 25) bmiState = "Overweight"; else if (fBMI > 18.5) bmiState = "in Normal Range"; else bmiState = "Underwight";`
		Die *for* stelling word omgeskakel na 'n *while* stelling.	`for (int l = 1; l <= nom; l++) stt = stt + " #";` **na** `int i = 1; while(i <= nom) { stt = stt + " #"; i++; }`
6	Besluitlogika	Aanpassing van die LMI se formule.	`double LMI = weight / Math.Pow(height, 2);` **na** `double LMI = weight /(height * height);`
		Die stringafvoer van die hutstekens (#) word omgeskakel.	`hash = hash + " #"` `hash += " #"` **na**
		Die wyse hoe 'n desimale veranderlike omgeskakel word na 'n heelgetal.	`int aantal = Convert.ToInt32(bmi);` `int aantal = (int)bmi;` **na**

Tabel 3-2 Die GGK se veranderings wat aan die programmeringsopdragte gemaak is

Verandering	Tipe	Beskrywing	Voorbeeld
Grafiese koppelvlak	7	Skuif die kontroles rond op die vorm en verstel ook die groottes van die kontroles.	
	8	Herbenaam die kontroles van die vorm.	label 1 → lblkg btnCalc → btnBMI Textbox1 → txtkg
	9	Herskryf van die teks in die gegewe kontroles.	
	10	Skuif die kontroles rond en herbenaam die kontroles	Sien voorbeeld van vlak 7 en 8

3.4 Die programkodeplagiaatherkenningstelsels

Daar kom verskeie PPHS'e in die literatuur voor. Hierdie plagiaatherkenningstelsels is ondersoek om te verseker dat die stelsels voldoen aan die vereistes genaamd ondersteuning van die programmeringstaal C# en is gratis beskikbaar. Die stelsels wat ondersoek is, word in Tabel 3-3 met die ooreenkomstige vereistes waaraan die stelsel voldoen, gelys.

Tabel 3-3 Die programkodeplagiaatherkenningstelsels wat ondersoek is

Plagiaatherkenningstelsel	C#	Gratis beskikbaar
AC	✕	✓
AntiCutAndPaste	✓	✓
CodeMatch®	✓	✓
CPD	✓	✓
Gplag	✕	✕
JPlag	✓	✓
Marble	✓	✕
MOSS	✓	✓
Plaggie	✕	✓
Plague	✕	✓
SID	✕	✓
SIM	✕	✓
SSID	✕	✓
Simian	✓	✓
YAP3	✕	✓

Die programkodeplagiaatherkenningstelsels wat volgens Tabel 3-3 aan die twee vereistes voldoen en in hierdie studie gebruik gaan word, is:

1. *AntiCutAndPaste Software Plagiarism detection software* (ACNP);
2. *CodeMatch*®;
3. *Copy/Paste Detector* (CPD);
4. *JPlag*;
5. *Measure of Software Similarity* (MOSS); en
6. *Simian*.

Elkeen van hierdie PPHS se resultate en ŉ refleksie oor die resultate word in hierdie hoofstuk bespreek. Verdere tegniese aspekte word as bylae bygevoeg, soos hoe

- die PPHS verkry en geïnstalleer word;
- die PPHS uiteengesit is (verstellings, GGK, ens.); en
- die afvoer of verslae weergegee.

3.5 *AntiCutAndPaste* (ACNP) plagiaatherkenner

Die ACNP is ontwerp om teksfragmente in gekopieerde programmeringsopdragte of in gewone teks te soek (ACNP, 2003). ACNP ondersteun C++, Visual Basic, Delphi, Java en C# programmeringstale. Verdere tegniese aspekte, soos die installering en uitleg van ACNP word in Bylaag C bespreek.

3.5.1 Die resultate wat verkry is deur ACNP

Soos genoem in Bylaag C, het ACNP verskillende opsies wat gestel kan word voordat die toetsing vir plagiaat uitgevoer word. Vir hierdie studie word die volgende opstellings vir die toetsing vir plagiaat deur die ACNP PPHS saamgestel (Figuur 3-2):

Opstelling 1:

- *number of lines/words that must match* = drie.

Opstelling 2:

- *number of lines/words that must match* = drie; en
- *compress white spaces*-opsie is geselekteer.

Opstelling 3:

- *number of lines/words that must match* = drie;
- *compress white spaces*-opsie is geselekteer;
- *ignore comments*-opsie is geselekteer; en
- *ignore strings*-opsie is geselekteer.

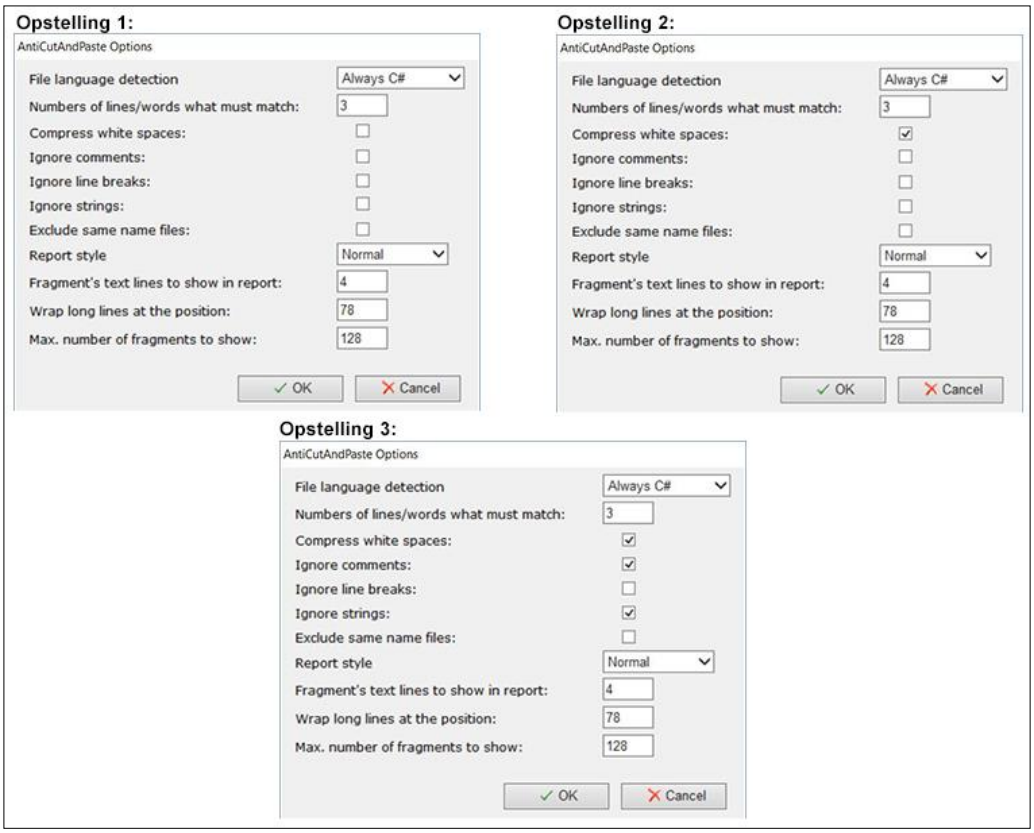


Figure 3-2 Die drie opstellings wat vir plagiaattoetsing deur ANCP gebruik is

Twee toetsgevalle word ondersoek (soos genoem in 3.1):

- **toetsgeval 1**: die vergelyking van die kodelêers; en
- **toetsgeval 2**: die vergelyking van die kodelêers saam met die ontwerplêers.

Die ooreenkomspersentasies wat deur elke tipe verandering vir elke opstelling in die gegewe toetsgeval verkry is, word in Tabel 3-4 op die volgende bladsy weergegee.

Soos verwag kan word, is daar ’n 100% ooreenkoms vir tipe verandering 1 vir toetsgevalle 1 en 2 met die opstelling 3 wat die *ignore comments*-opsie insluit. Die verandering van veranderlikes se name in tipe verandering 2 beïnvloed die opsporing van plagiaat meer as wat daar verwag is in toetsgeval 1 deurdat die ooreenkoms minder as 30% is, maar toetsgeval 2 se ooreenkoms is bo 70%. ’n Moontlike verklaring kan wees dat ANCP die programkode karakter vir karakter vergelyk en omdat daar verskillende aantalle veranderlikes in die programmeringsopdrag wat reg deur die programkode gebruik word voorkom, veroorsaak dit dat minimale programkodelyne ooreenstem. Vir toetsgeval 2 is die KON en die VPO se ontwerplêers 100% dieselfde wat dus die totale opsporingpersentasie laat toeneem.

Tabel 3-4 Die resultate wat deur ACNP verkry is

Ooreenkomspersentasies van toetsgeval 1				
		Opstellings		
Tipe verandering		1	2	3
1	Kommentaar en indentering	23.61%	34.12%	100.00%
2	Veranderlikes	25.18%	15.30%	17.07%
3	Verklarings	52.42%	38.39%	31.48%
4	Programmodules	78.89%	46.81%	35.83%
5	Stellings	81.15%	51.18%	43.57%
6	Besluitlogika	86.69%	60.68%	47.46%
7	Skuif kontroles	84.88%	53.92%	32.65%
8	Herbenaming van kontroles in GGK	33.25%	21.32%	9.73%
9	Verander teks in GGK	96.17%	59.48%	47.23%
10	Skuif en herbenaam kontroles in GGK	27.56%	17.70%	9.73%
Ooreenkomspersentasies van toetsgeval 2				
1	Kommentaar en indentering	77.13%	85.21%	100.00%
2	Veranderlikes	81.92%	76.09%	76.15%
3	Verklarings	95.50%	77.06%	83.52%
4	Programmodules	96.21%	87.43%	83.47%
5	Stellings	84.75%	87.53%	84.75%
6	Besluitlogika	96.29%	89.77%	85.89%
7	Skuif kontroles	94.84%	70.64%	61.23%
8	Herbenaming van kontroles in GGK	56.39%	45.77%	41.82%
9	Verander teks in GGK	95.11%	70.96%	70.35%
10	Skuif en herbenaam kontroles in GGK	53.86%	45.91%	38.52%

Daar sou verwag word dat opstelling 3 wat die kommentaar, spasies, onnodige stringe en breuke verwyder 'n effektiewer opsie sou wees van die drie opstellings, maar opstelling 1 wat geen addisionele opsie byvoeg nie het die hoogste ooreenkomspersentasie vir tipes verandering 2 tot 10.

As daar aanvaar word dat ’n 60% ooreenkomspersentasie as plagiaat beskou word, kan studente toetsgeval 1 van ACNP boikot deur

- veranderlikes se name te verander;
- die kontroles se benaming te verander van die GGK; en
- die kontroles rond te skuif, asook die benaming te verander van die GGK.

As daar aanvaar word dat ’n 60% ooreenkomspersentasie as plagiaat beskou word, kan studente toetsgeval 2 van ACNP boikot deur

- die kontroles se benaming te verander van die GGK; en
- die kontroles rond te skuif, asook die benaming te verander van die GGK.

Die lae ooreenkomspersentasie wat vir toetsgevalle 1 en 2 vir tipe verandering 10 verkry is, kan moontlik wees omdat meer as een verandering aan die ontwerplêer gemaak is. Verdere ondersoek moet dus ingestel word na die invloed van meer as een verandering.

3.5.2 Refleksie oor die ACNP PPHS

Die verkryging en installering van die ACNP PPHS is maklik en eenvoudig. Die evalueringsweergawe bemoeilik wel die opsporingsproses omdat die resultate wat verkry is nie gestoor of gekopieer kan word nie en skermafdrukke van die afvoer wat verkry is, geneem moes word. Dit het ook die eksperiment onder tydsdruk geplaas omdat die evalueringsweergawe binne 15 dae verval.

Die afvoer wat die ACNP PPHS verskaf, gee stuk vir stuk die karakterooreenkomste wat verkry is tussen die twee lêers wat geïnterpreteer kan word deur ’n deskundige, maar daar bestaan nie ’n opsommende afdeling wat die finale en belangrike resultate weergee nie. Omdat die afvoer slegs die karakterooreenkomste gee wat tussen die twee lêers gevind is, moes al die ooreenkomskarakters bymekaar getel word waarna dit vergelyk is met die kontrole se aantal karakters. Hierdie optelling- en vergelykingsproses was tydrowend en skep ’n omgewing vir menslike foute.

Die grafiese gebruikerskoppelvlak waarmee die gebruiker die toevoer tot die program se uitleg verskaf, is basies en verstaanbaar, alhoewel, die eerste keer wat die program gebruik is om te toets of die program wel die C# taal herken, is geen afvoer verkry nie en daar is dus gedink die program werk nie. Nadat die verslagstyl by meer opsies verander is en die toets weer uitgevoer is, het daar afvoer vertoon dat die een lêer ’n kopie van die ander een is omdat die lêers wat gebruik is presies dieselfde was. Daar kan verdere eksperimente uitgevoer word

deur die *aantal lyne/woorde wat moet ooreenstem-opsie* te gaan verander en daardie resultate te bestudeer.

Die navorser se gevolgtrekking is dat die ACNP PPHS meer op die lyn-vir-lynooreenkomste, soos met teksplagiaat fokus, omdat daar dubbele waardes in die resultate voorkom en dat die resultate as karakterooreenkomste weergegee word, maar die ACNP-sagtewaremaatskappy verskaf selfs 'n *AntiPlagiarist*-program spesifiek vir teksplagiaat.

3.6 *CodeMatch®*

Die *CodeMatch*® PPHS is deel van die *CodeSuite*® sagtewarepakket. *CodeMatch*® vergelyk duisende programkodelêers in verskeie vouers en subafdelings van die gidse om so-doende te bepaal watter lêers die meeste met mekaar korreleer. *CodeMatch*® maak gebruik van unieke algoritmes om direkte kopiëring of veranderde programkode te identifiseer. Die algoritmes wat gebruik kan word, is soos volg: Verklaring-, Kommentaar/String-, Instruksievolgorde- en Identifiseerderooreenstemming. Die programmeringstale wat deur *CodeMatch*® ondersteun word, word in Tabel 3-5 gelys (SAFE Corporation, 2015a; SAFE Corporation, 2015b).

Tabel 3-5 Programmeringstale wat deur *CodeMatch*® ondersteun word (SAFE Corporation, 2015a)

ABAP	ASM-6502	ASM-65C02	ASM-65816	ASM-M68k
BASIC	C	C++	C#	COBOL
Delphi	DRI ASM	Flash ActionScript	Fortran	FoxPro
Go	Java	JavaScript	LISP	Lotus Script
MASM	MATLAB	MPE/iX	Pascal	Perl
PHP	PL/M	PowerBuilder	PowerHouse	Prolig
Python	RealBasic	Ruby	Scala	SQL
TAL	TCL	Verilog	VHDL	Visual Basic

3.6.1 Die resultate wat deur *CodeMatch*® verkry is

Soos genoem in Bylaag D, beskik *CodeMatch*® oor vier verskillende ooreenkoms-algoritmes. Elke algoritme word in hierdie eksperiment eers afsonderlik gebruik vir die opsporing van plagiaat; daarna word al vier algoritmes gelyktydig gebruik. Die resultate wat in hierdie eksperiment verkry is, word in Tabel 3-6 uiteengesit.

Tabel 3-6 Resultate wat met *CodeMatch*® deur die verskillende algoritmes verkry is

Ooreenkomspersentasie van toetsgeval 1					
	Algoritmes				
Tipe verandering	Verklaring algoritme	Kommentaar/ String algoritme	Identifiseerder algoritme	Instruksie volgorde algoritme	Vier algoritmes gelyktydig
1	100.00%	31.75%	100.00%	100.00%	88.42%
2	43.25%	100.00%	87.50%	30.58%	71.00%
3	92.00%	100.00%	100.00%	32.58%	86.00%
4	90.83%	100.00%	99.25%	57.92%	89.08%
5	94.75%	100.00%	100.00%	60.92%	90.58%
6	95.50%	100.00%	99.58%	94.50%	97.50%
7	81.00%	99.75%	99.08%	100.00%	95.25%
8	24.00%	86.42%	93.00%	100.00%	81.58%
9	94.08%	92.33%	100.00%	94.92%	95.25%
10	23.42%	84.67%	91.58%	100.00%	80.83%
Ooreenkomspersentasie van toetsgeval 2					
1	100.00%	65.88%	100.00%	100.00%	94.21%
2	60.25%	97.67%	92.63%	65.29%	85.50%
3	95.58%	100.00%	100.00%	61.71%	93.00%
4	95.42%	100.00%	99.63%	78.96%	94.54%
5	97.38%	100.00%	100.00%	80.46%	95.29%
6	97.75%	100.00%	99.79%	97.25%	98.75%
7	90.50%	99.88%	99.54%	100.00%	97.63%
8	55.58%	93.21%	93.46%	82.21%	90.79%
9	97.04%	96.17%	100.00%	97.46%	97.63%
10	55.29%	92.33%	92.58%	82.17%	90.42%

Die verandering van veranderlikes se name in tipe verandering 2, asook die verandering van die kontroles se benaming in tipes verandering 8 en 10 van toetsgevalle 1 en 2 het, soos verwag, daartoe gelei dat die **verklaringooreenkomsalgoritme** nie die programkodelyne as dieselfde beskou nie.

Soos verwag, het die **kommentaar-/string- ooreenkomsalgoritme** opgespoor dat daar met tipe verandering 1 vir toetsgevalle 1 en 2 veranderings aan die kommentaar aangebring is. Die tipes verandering 2 tot 6 het 100% ooreenkoms verkry vir toetsgevalle 1 en 2, maar die tipes verandering 7 tot 10 wat veranderings maak in die GGK se ooreenkomspersentasie is hoog, maar nie 100% nie. 'n Moontlike verklaring kan wees dat kommentaar gegenereer word soos wat daar aan die kontroles van die GGK verander word.

Daar kan duidelik gesien word dat die **instruksieooreenkomsalgoritme** beïnvloed word deur die verandering in verklarings in tipe verandering 3 en die verandering in programmodules in tipe verandering 4 van albei toetsgevalle, omdat die programkode in beide hierdie twee veranderings rondgeskuif word.

Daar sou verwag word dat die ooreenkomspersentasie van tipe verandering 2 met die **instruksievolgordealgoritme** vir albei toetsgevalle nie so negatief beïnvloed word nie, aangesien die algoritme, soos beskryf in Bylaag D nie beïnvloed word deur die verandering van veranderlike name nie.

Met die **instruksievolgordealgoritme** verkry tipes verandering 8 en 10 'n ooreenkoms van 100% in toetsgeval 1, terwyl die ooreenkoms in toetsgeval 2 minder is. 'n Moontlike verklaring kan wees dat die nuwe name van die kontroles in die kodelêer dit laat voorkom asof daar nuwe instruksies geskep is. Daar sou verwag word dat die rondskuif en die herbenaming van die kontroles in tipe verandering 10 'n groter verskil sal oplewer vergeleke met net die herbenaming van die kontroles in tipe verandering 8 in toetsgeval 1.

Die gelyktydige gebruik van al **vier die bogenoemde algoritmes** is 'n meer effektiewe oplossing om plagiaat op te spoor. Soos uit die resultate gesien kan word, is daar vir toetsgeval 1 nie 'n ooreenkomspersentasie minder as 70% nie en nie 'n ooreenkomspersentasie minder as 85% vir toetsgeval 2 nie.

Studente kan probeer om *CodeMatch*® te boikot deur

- veranderlikes se name te verander;
- die kontroles van die GGK se benaming te verander; of
- die kontroles rond te skuif, asook die benaming te verander van die kontroles op die GGK.

Een van die vier algoritmes sal egter wel die plagiaat ontbloot.

Vir die ooreenkomspersentasie van elke opdrag van elke tipe verandering, kyk Bylaag P.

3.6.2 Refleksie oor *CodeMatch*®

Die verkryging van die universiteitslisensie vir *CodeSuite*®, die pakket wat *CodeMatch*® bevat, is volgens die voorgeskrewe voorskrifte uitgevoer en goedkeuring is ontvang tesame met die gepaste bemagtigingssleutel vir *CodeSuite*®. Hierdie sleutel aktiveer *CodeSuite*® slegs op een spesifieke rekenaar. Indien dit nie op 'n draagbare rekenaar geaktiveer is nie, soos in die navorser se geval, kan dit frustrerend wees.

Die uitleg van *CodeMatch*® is gebruikersvriendelik. Die grafiese koppelvlak is op so 'n manier uiteengesit dat die gebruiker moet begin deur die lêers, dan die programmeringstaal en lêertipe en daarna die spesifieke algoritme te selekteer en dan word die ooreenkomste uitgevoer. *CodeMatch*® vertoon 'n boodskap aan die gebruiker indien die programmeringstaal nie geselekteer is nie of as daar nie minstens een algoritme geselekteer is nie.

Die gebruiker moet die verskillende algoritmes in ag neem en verstaan wanneer toetsing vir plagiaat uitgevoer word. Verskeie kombinasies van die algoritmes gee verskillende resultate en kan verwarring veroorsaak as die afvoerverslag nie bygevoeg word nie.

Omdat *CodeMatch*® die resultate as 'n databasislêer stoor en die gebruiker die lêer met *CodeSuite*® moet oopmaak en spesifiseer as watter lêer die afvoer gestoor moet word, kan die stoor van die afvoer 'n langdradig proses wees, veral as hierdie proses, soos in hierdie eksperiment oor die 600 keer herhaal moet word.

3.7 Copy/Paste Detector (CPD)

CPD is 'n hulpmiddel om duplikate in programkode te vind en is 'n module van PMD. PMD is 'n kragtige statiese analitiese hulpmiddel wat 'n omvattende stel reëls bevat wat op verskeie maniere opgestel kan word. PMD ondersoek Java programkode en kyk of daar potensiële probleme, soos moontlike foute/haakplekke, dooie kode, suboptimale kode of oorgekompliseerde uitdrukkings voorkom. PMD is gebaseer op 'n stel reëls wat ooreenstemmend is met goeie kodering (Rutar *et al.*, 2004).

Die CPD-model spoor programkodeduplikate in programme op. CPD was eers gebaseer op 'n variasie van Michael Wise se *Greedy String Tiling* algoritme, waarna dit heeltemal deur Brian Ewins met die gebruik van die *Burrows-Wheeler* transformasie herskryf is. Laastens is dit weer herskryf deur Steve Hawkins wat die *Karp-Rabin* stringooreenstemmingalgoritme gebruik het (PMD Contributors, 2011). CPD is 'n kentekengebaseerde opsporingstelsel wat slegs die duplikaatlyne van die lêers as afvoer terugstuur (Copeland, 2003). Verdere tegniese aspekte aangaande CPD word in Bylaag E weergegee.

CPD ondersteun verskeie programmeringstale wat in Tabel 3-7 weergegee word. Indien die programmeringstaal nie in die tabel voorkom nie, laat CPD die gebruiker toe om 'n nuwe programmeringstaal by te voeg deur die stappe op die volgende webblad te volg: http://pmd.sourceforge.net/pmd-4.3.0/cpd-parser-howto.html.

Tabel 3-7 Die vasgestelde programmeringstale wat in CPD gebruik kan word

Apex	C#	C++	Fortran	Go	Groovy
Java	JavaScript	JSP	MATLAB	Objective-C	Perl
PHP	PL/SQL	Python	Ruby	Scala	Swift

3.7.1 Die resultate wat deur CPD verkry is

Vir die toetsing van CPD is die *report duplicate chunks larger than* (drumpelwaarde)opsie verstel na 3, 6, 9 en 12 om die opsporing van plagiaat te ondersoek. Die ooreenkomspersentasies wat deur hierdie opsies verkry is, word in Tabel 3-8 weergegee.

Tabel 3-8 Die resultate wat verkry is deur CPD met die verskillende duplikaatgroottes

Ooreenkomspersentasie van toetsgeval 1					
		Drumpelwaardes			
	Tipe verandering	3	6	9	12
1	Kommentaar en indentering	99.68%	99.68%	99.68%	99.68%
2	Veranderlikes	109.66%	63.00%	47.71%	39.28%
3	Verklarings	107.58%	93.33%	89.72%	83.46%
4	Programmodules	95.34%	92.28%	90.71%	89.15%
5	Stellings	96.62%	84.35%	80.80%	79.64%
6	Besluitlogika	92.48%	88.55%	88.09%	88.09%
7	Skuif kontroles	100.00%	100.00%	99.93%	99.93%
8	Herbenaming van kontroles in GGK	97.59%	82.29%	71.56%	71.56%
9	Verander teks in GGK	92.71%	90.06%	90.06%	90.06%
10	Skuif en herbenaam kontroles in GGK	89.86%	78.63%	63.07%	63.07%
Ooreenkoms persentasie van toetsgeval 2					

1	**Kommentaar en indentering**	95.44%	95.44%	95.44%	95.44%
2	**Veranderlikes**	99.35%	81.35%	74.75%	71.29%
3	**Verklarings**	104.72%	98.17%	95.87%	92.02%
4	**Programmodules**	96.68%	95.19%	94.44%	93.60%
5	**Stellings**	98.07%	92.10%	88.93%	88.17%
6	**Besluitlogika**	96.83%	96.74%	96.49%	96.49%
7	**Skuif kontroles**	99.72%	99.11%	99.11%	99.11%
8	**Herbenaming van kontroles in GGK**	96.31%	92.34%	78.71%	77.61%
9	**Verander teks in GGK**	91.04%	91.04%	91.04%	90.67%
10	**Skuif en herbenaam kontroles in GGK**	90.53%	88.26%	73.63%	72.31%

In die Tabel 3-8 hier bo is dit duidelik dat die duplikaatgrootte 3 (drumpelwaarde) by tipes verandering 2 en 3 in toetsgeval 1 'n ongeldige ooreenkomspersentasie verskaf van bo 100%, dus gaan hierdie duplikaatgrootte 3 verwerp word. Daar bestaan 'n groot verskil tussen die ooreenkomspersentasie wat met die duplikaatgrootte 3 verkry is teenoor dié van 6 vir die tipe verandering 2 in toetsgeval 1.

Tipe verandering 1 van kommentaar en indentering vir toetsgeval 1 bly konstant in al vier duplikaatgroottes. Uit die resultate van elke programmeringsopdrag in Bylaag P word daar gevind dat net programmeringsopdragte twee en drie nie 'n 100% ooreenkoms verkry het nie: dit is moontlik dat CNP nie kommentaar in ag neem nie.

Soos verwag, het die tipe verandering 2 'n negatiewe invloed op die opsporing van plagiaat in toetsgeval 1, maar die byvoeging van die ontwerplêer in toetsgeval 2 druk die ooreenkomspersentasie weer in 'n positiewe rigting.

Die ooreenkomspersentasie neem ook af namate die grootte van die duplikaatgrootte (drumpelwaarde) toeneem omrede die programkodes wat met mekaar moet ooreenstem groter is.

Tipe verandering 7 wat net die kontroles op die GGK rondskuif, ontduik nie die opsporing van plagiaat nie, aangesien daar in toetsgeval 1 slegs een programmeringsopdrag is wat **nié** 'n 100% ooreenkoms verkry het vir die duplikaatgroottes 9 en 12 nie (kyk Bylaag P). Daar sou verwag word, omdat tipe verandering 7 nie so 'n groot invloed het nie, dat die ooreenkomspersentasies vir tipes verandering 8 en 10 dieselfde sal wees, maar die

ooreenkomspersentasie neem met 3.41% tot 21.5% af vir tipe verandering 8 en vir tipe verandering 10 neem die ooreenkomspersentasie met 9.19% tot 26.8% af.

Indien daar aangeneem word dat ’n 60% ooreenkomswaarde as plagiaat beskou word, kan studente CPD in toetsgeval 1 boikot deur die veranderlikes se name te verander (tipe verandering 2) indien die duplikaatgrootte 9 of 12 is.

3.7.2 Refleksie oor CPD

Die grafiese gebruikerskoppelvlak maak dit vir die gebruiker makliker om die toevoer na CPD te gee. Van die opsies wat nie vir die gespesifiseerde programmeringstaal beskikbaar is nie word onaktief en kan nie deur die gebruiker geselekteer word nie. Die programmeringsopdragte se programkodelêers van albei, die kontrole en veranderde vlak, moet in een vouer voorkom wat dan aan CPD gegee word.

Die resultate wat deur CPD verkry word, is soms herhalend, soos gesien kan word met die duplikaatgrootte 3 in Tabel 3-8 wat ook dan die waardes van meer as 100% veroorsaak/verklaar. Die navorser moes dus self deur al die resultate gaan wat verkry is en die herhalende ooreenkomste wat aangedui is aftrek van die oorspronklike aantal ooreenkomste. Dit was ’n tydsame proses en van die resultate was so veelvuldig dat dit eerder uitgedruk is en per hand nagegaan is vir herhalings. Soos die drumpelwaarde groter geword het, het die herhaling van ooreenkomste verminder, maar daar het steeds herhalende ooreenkomste in die resultate voorgekom. Hierdie herhalings van ooreenkomste benadeel CPD met betrekking tot outomatiese plagiaatherkening omdat daar steeds per hand deur die resultate gegaan moet word, wat kan veroorsaak dat menslike foute kan voorkom.

3.8 JPlag

JPlag, vir die opsporing van sagtewareplagiaat, is deur Guido Malpohl aan die Universiteit van Karlsruhe ontwikkel waar dit in 1996 begin het as ’n student se navorsingsprojek. ’n Paar maande later het hierdie projek gegroei tot die eerste aanlynstelsel. In 2005 het Emeric Kwemou en Moritz Kroll JPlag verander in ’n webdiens. JPlag maak gebruik van ’n strukturele gebaseerde benadering deur die programme in kentekenstringe wat die struktuur van die program verteenwoordig te verander. Om twee kentekenstringe te vergelyk, maak JPlag gebruik van die *grypstringtelling*algoritme wat deur Michael Wise voorgestel is, maar met verskillende optimaliserings (JPlag, 2013).

JPlag vind ooreenkomste tussen ’n stel programkodelêers en ondersteun Java, C, C++, C# en Scheme programmeringstale, asook natuurlike taalteks.

3.8.1 Die resultate wat deur JPlag verkry is

JPlag het ’n opsie, soos uitgewys in Figuur 1 in Bylaag F wat die sensitiwiteit van die ooreenkomste met behulp van die kentekengrootte kan stel. Hoe kleiner hierdie getal is, hoe sensitiewer is die ooreenkomste. Die ooreenkomssensitiwiteitwaardes (kenteken) 3, 6, 9 en 12 word gedurende hierdie toetsing van JPlag gebruik en die resultate wat verkry is, word in Tabel 3-9 weergegee.

Tabel 3-9 Die resultate van JPlag met vier ooreenkomssensitiwiteitwaardes (kenteken)

Ooreenkomspersentasie van toetsgeval 1					
		Kentekengrootte			
	Tipe verandering	3	6	9	12
1	Kommentaar en indentering	100.00%	100.00%	100.00%	100.00%
2	Veranderlikes	100.00%	100.00%	100.00%	100.00%
3	Verklarings	92.10%	81.04%	71.78%	60.26%
4	Programmodules	93.94%	82.22%	71.78%	72.09%
5	Stellings	92.90%	91.57%	89.82%	88.91%
6	Besluitlogika	98.57%	98.57%	96.51%	90.49%
7	Skuif kontroles	100.00%	100.00%	100.00%	100.00%
8	Herbenaming van kontroles in GGK	100.00%	100.00%	100.00%	100.00%
9	Verander teks in GGK	99.46%	98.13%	97.68%	96.73%
10	Skuif en herbenaam kontroles in GGK	100.00%	100.00%	100.00%	100.00%
Ooreenkomspersentasie van toetsgeval 2					
1	Kommentaar en indentering	100.00%	100.00%	100.00%	100.00%
2	Veranderlikes	100.00%	100.00%	100.00%	100.00%
3	Verklarings	96.05%	90.52%	85.89%	80.13%
4	Programmodules	96.97%	91.11%	87.27%	86.05%
5	Stellings	96.45%	95.78%	94.91%	94.46%
6	Besluitlogika	99.28%	99.29%	98.25%	95.24%
7	Skuif kontroles	100.00%	100.00%	100.00%	100.00%
8	Herbenaming van kontroles in GGK	100.00%	100.00%	100.00%	100.00%

9	**Verander teks in GGK**	99.73%	99.07%	98.84%	98.37%
10	**Skuif en herbenaam kontroles in GGK**	100.00%	100.00%	100.00%	100.00%

Tipes verandering 1, 2, 7, 8 en 10 in toetsgevalle 1 en 2 het vir al vier die kentekens 'n ooreenkomspersentasie van 100% verkry. Die verandering van die GGK boikot nie hierdie stelsels nie, maar slegs die verandering van die teks op die GGK veroorsaak dat daar nie meer 'n 100% ooreenkoms is nie.

In toetsgeval 1 verkry tipe verandering 3 wat verklarings verander 'n laer ooreenkomspersentasie en dit verminder soos die kentekengrootte toeneem. Dit kan moontlik wees dat omdat slegs klein gedeeltes van die programkode eerstens met mekaar vergelyk word, die verskil kleiner is as wanneer die kentekengrootte toeneem en groter gedeeltes in ag geneem word wat dan geskuif of verander is.

3.8.2 Refleksie oor JPlag

JPlag is 'n eenvoudige Java-program wat in die bevelaanporring hardloop as al die nodige opsies vir JPlag om uit te voer verskaf word. Die opsommende eindresultaat word aan die gebruiker in die bevelaanporring gegee waarna dit verder in diepte bestudeer kan word in die HTML-lêer wat gestoor is. 'n Probleem wat kan voorkom, is die oorskrywing van die resultate wanneer die gebruiker vergeet het om die gids waarnatoe die resultaat geskryf moet word te verander. Dit kan ook gebeur wanneer daar nie 'n gespesifiseerde gids is vir die resultate nie en JPlag gebruik die verstek wat ook na elke ooreenkomsvergelyking oorgeskryf word.

Die uitleg van die resultate in die HTML-lêer vir verdere bestudering van die ooreenkomste is gebruikersvriendelik. Die kleurpassing van die ooreenkomste maak dit makliker vir die gebruiker om te sien watter programmeringskode deur JPlag as dieselfde aangedui word. Die uitleg van die resultate in die HTML-lêer vergemaklik die lees van die ooreenkomste tussen die opdragte.

3.9 MOSS

MOSS (Measure Of Software Similarity) vir die meet van sagteware-ooreenkomste is in 1994 deur Alex Aiken aan die Stanford Universiteit ontwikkel. MOSS vergelyk die strukturele ooreenkomste in programme deur gebruik te maak van die dokumentvingerafdrukalgoritme genaamd vensterskepping (Moss, 2014).

Die dokumentvingerafdruktegniek verdeel 'n dokument in aangrensende sub-stringe, genaamd k-gramme, waar die gebruiker die k spesifiseer. Elke k-gram word gehuts waarna 'n

substel van die gehutste k-gramme geselekteer word as die dokument se vingerafdruk (Burrows *et al.*, 2007).

MOSS is ŉ outomatiese stelsel wat ooreenkomste in programme identifiseer en is as ŉ Internetdiens beskikbaar. MOSS kan tans die programmeringstale in Tabel 3-10 analiseer.

Tabel 3-10 Die programmeringstale wat MOSS ondersteun

a8086 assembly	Ada	C	C#	C++
Haskell	FORTRAN	Java	JavaScript	Lisp
MATLAB	MIPS assembly	ML	Modula2	Pascal
Perl	Python	Scheme	Spice	TCL
Verilog	VHDL	Visual Basic	HCL2	

3.9.1 Die resultate wat deur MOSS verkry is

MOSS beskik oor drie opsies (M, N, C) wat verstel kan word, soos bespreek in Bylaag G. Die enigste opsie wat verstel kan word wat dalk ŉ verskil gaan maak, is die opsie M wat die maksimum aantal kere stel wat ŉ gedeelte kan voorkom voordat dit geïgnoreer word. Die opsie is verander na 3, 6, 9 en 12, maar geen verandering aan die ooreenkomspersentasie word aangedui nie, soos in Tabel 3-11 op die volgende bladsy gesien kan word.

Tipes verandering 1 en 2 in toetsgeval 1 beskik oor dieselfde ooreenkomspersentasie. Daar sou verwag word dat die tipe verandering 2 ŉ groter negatiewe effek op die ooreenkomspersentasie tot gevolg sal hê. ŉ Moontlike verklaring kan wees dat dieselfde aantal veranderings wat in tipe verandering 1 aangebring is ook in tipe verandering 2 gemaak is.

Tipes verandering 1, 2, 4, 5 en 6 se ooreenkomspersentasies neem af vanaf toetsgeval 1 na toetsgeval 2 wanneer die ontwerplêer bygevoeg word. Daar sou verwag word dat die ooreenkomspersentasie eerder sal toeneem, aangesien daar geen veranderings in die ontwerplêer gemaak is nie. ŉ Moontlike verklaring kan wees dat die vingerafdruk wat gebruik word om die lêers met mekaar te vergelyk, verskil (groter is) vir toetsgeval 2 en dus ŉ laer ooreenkomspersentasie lewer.

Tabel 3-11 Die resultate wat deur MOSS met die verskillende M-opsie stellings verkry is

Ooreenkomspersentasie van toetsgeval 1					
		M-opsies			
	Tipe verandering	3	6	9	12
1	Kommentaar en indentering	97.17%	97.17%	97.17%	97.17%
2	Veranderlikes	97.17%	97.17%	97.17%	97.17%
3	Verklarings	53.58%	53.58%	53.58%	53.58%
4	Programmodules	67.42%	67.42%	67.42%	67.42%
5	Stellings	77.75%	77.75%	77.75%	77.75%
6	Besluitlogika	82.29%	82.29%	82.29%	82.29%
7	Skuif kontroles	60.00%	60.00%	60.00%	60.00%
8	Herbenaming van kontroles in GGK	60.00%	60.00%	60.00%	60.00%
9	Verander teks in GGK	55.96%	55.96%	55.96%	55.96%
10	Skuif en herbenaam kontroles in GGK	60.00%	60.00%	60.00%	60.00%
Ooreenkomspersentasie van toetsgeval 2					
1	Kommentaar en indentering	78.58%	78.58%	78.58%	78.58%
2	Veranderlikes	78.46%	78.46%	78.46%	78.46%
3	Verklarings	56.83%	56.83%	56.83%	56.83%
4	Programmodules	63.73%	63.73%	63.73%	63.73%
5	Stellings	68.88%	68.88%	68.88%	68.88%
6	Besluitlogika	71.15%	71.15%	71.15%	71.15%
7	Skuif kontroles	78.58%	78.58%	78.58%	78.58%
8	Herbenaming van kontroles in GGK	78.58%	78.58%	78.58%	78.58%
9	Verander teks in GGK	76.63%	76.63%	76.63%	76.63%
10	Skuif en herbenaam kontroles in GGK	78.58%	78.58%	78.58%	78.58%

Die tipes verandering 7, 8, 9 en 10 wat veranderings maak aan die GGK verkry laer ooreenkomspersentasies in toetsgeval 1 as dié van die veranderings van kode. Veranderingtipes 7 en 8 verkry dieselfde ooreenkomspersentasies in toetsgevalle 1 en 2. Die gekombineerde GGK veranderings in tipe verandering 10 lewer dieselfde ooreenkomspersentasie as wanneer net een van die twee GGK veranderings gemaak is.

As daar aanvaar word dat ’n 60% ooreenkoms as plagiaat beskou word, kan studente MOSS boikot in toetsgeval 1 deur

- veranderings te maak in verklarende stellings deur programkode rond te skuif, die veranderlikes se volgorde te verander, onnodige programkode by te voeg; en
- die teks te verander van die GGK.

As daar aanvaar word dat ’n 60% ooreenkoms as plagiaat beskou word, kan studente MOSS boikot in toetsgeval 2 deur

- veranderings te maak in verklarende stellings deur programkode rond te skuif, die veranderlikes se volgorde te verander en onnodige programkode by te voeg.

3.9.2 Refleksie oor MOSS

MOSS is ingewikkelder omdat dit ’n skrip is wat aan ’n web-bediener kommunikeer. Die grafiese koppelvlak wat deur ’n gemeenskapslid ontwikkel is, vergemaklik dit vir persone wat nie kenners van skrip of die rekenaarveld is nie.

Die verkryging van die gebruiker-id is heel eenvoudig, maar omdat die hele skrip outomaties deur MOSS teruggestuur word, is daar eers na die derde e-pos mooi deur die skrip gegaan en is daar gesien dat die gebruiker-id binne-in die skrip is en dat dit nie afsonderlik per e-pos aan die gebruiker gestuur word nie.

Die GGK van MOSS verduidelik elke opsie volledig. Aanvanklik is die basislêer wat opgelaai kan word, beskou as die kontrolelêer omdat daar nie ’n verduideliking gegee is by die kies bronlêerknoppie nie. Nadat daar weer nagelees is oor wat die doel van die basislêer is, is daar tot die besef gekom dat die basislêer gesien word as ’n memorandum of dit wat vooraf al gegee is.

Omdat die GGK van MOSS die gegewe inligting aan die MOSS-bediener stuur, is daar ’n tydperk wat die program hang voordat die resultaatskakel verskyn. Gedurende hierdie tyd is dit aan te bevele dat die program aan die gebruiker vertoon dat hy/sy besig is om die data te verwerk. ’n Mens weet dus nie altyd of die program wel die inligting oorstuur na MOSS nie en ook nie altyd wanneer afvoer suksesvol van MOSS ontvang is nie in gevalle waar jy nie die skakel dophou om te sien of die kode verander het nie. In sommige gevalle wanneer die inligting na MOSS oorgestuur word, is ’n foutboodskap ontvang of die program reageer glad nie en moes toegemaak word en weer oopgemaak word.

Die resultate wat verkry word in HTML-formaat word tydelik op MOSS se bediener gestoor. Die toetsing is uitgevoer en die resultate geboekmerk om later te bestudeer, maar toe daar op die boekmerk geklik is, was daar nie meer toegang tot die resultate nie en moes al die toetsing herhaal word.

3.10 Simian

Simian identifiseer duplikate in die programmeringstale wat in Tabel 3-12 gelys is, asook in natuurlike taalteks. Simian gebruik ŉ lyngebaseerde stringvergelyking om die ooreenkomste tussen lêers te bereken (Simian, 2003).

Simian is ŉ Java-program wat op enige bedryfstelsel (Windows, MacOS, Linux) kan hardloop indien Java2 of hoër daarop voorkom.

Tabel 3-12 Die programmeringstale wat Simian ondersteun

ASP	C	C#	COBOL	C++
Java	JSP	Lisp	Groovy	JavaScript
Objective-C	Ruby	SQL	HTML	XML
Visual Basic	Swift			

Vir hierdie eksperiment was die evalueringslisensie voldoende en word Simian dus net afgelaai. Die lêer word uitgepak en Simian kan in die bevelaanporring uitgevoer word deur die toepassinglêer of Java-lêer se bevele uit te voer.

3.10.1 Die resultate wat deur Simian verkry is

Soos bespreek in Bylaag H, het Simian die opsie om te spesifiseer wat die aantal lyne moet wees wat met mekaar moet ooreenstem (drumpelwaarde). Vir die toetsing van Simian word 3, 6, 9 en 12 as die drumpelwaardes gebruik en die resultate wat verkry is, weergegee in Tabel 3-13 op die volgende bladsy.

Tabel 3-13 Simian se resultate met die verskillende drumpelwaardes

Ooreenkomspersentasie van toetsgeval 1					
		Drumpelwaardes			
	Tipe verandering	3	6	9	12
1	Kommentaar en indentering	100.00%	100.00%	100.00%	100.00%
2	Veranderlikes	10.13%	0.00%	0.00%	0.00%
3	Verklarings	58.38%	31.74%	28.19%	8.90%
4	Programmodules	67.99%	58.26%	48.17%	22.52%
5	Stellings	64.98%	60.10%	45.02%	23.29%
6	Besluitlogika	86.57%	77.93%	55.57%	44.18%
7	Skuif kontroles	78.67%	52.63%	44.36%	43.22%
8	Herbenaming van kontroles in GGK	20.67%	10.32%	0.00%	0.00%
9	Verander teks in GGK	81.89%	62.62%	49.08%	33.75%
10	Skuif en herbenaam kontroles in GGK	17.08%	10.32%	0.00%	0.00%
Ooreenkomspersentasie van toetsgeval 2					
1	Kommentaar en indentering	100.00%	100.00%	100.00%	100.00%
2	Veranderlikes	70.93%	67.68%	67.68%	67.68%
3	Verklarings	86.80%	77.67%	76.94%	70.27%
4	Programmodules	90.01%	86.95%	83.88%	75.63%
5	Stellings	88.47%	86.91%	82.04%	74.78%
6	Besluitlogika	95.67%	92.88%	85.81%	82.41%
7	Skuif kontroles	85.58%	67.94%	62.32%	61.54%
8	Herbenaming van kontroles in GGK	41.53%	29.59%	21.47%%	17.89%
9	Verander teks in GGK	87.67%	74.52%	65.45%	54.91%
10	Skuif en herbenaam kontroles in GGK	39.21%	29.59%	21.47%	17.89%

Simian ignoreer kommentaar soos daar gesien kan word met tipe verandering 1 se resultate wat ’n 100% ooreenkoms vir toetsgevalle 1 en 2 verkry het.

Tipe verandering 2 verkry in toetsgeval 1 ’n baie lae ooreenkomspersentasie van 10.13% met die drumpelwaarde 3 en die ooreenkomspersentasie daal na 0% toe wanneer hierdie

drumpelwaarde verdubbel word. ’n Moontlike verklaring kan wees dat Simian karakter vir karakter die ooreenkomste toets en met die verandering van die veranderlikes skuif dit die programkode wat veroorsaak dat niks meer ooreenstem met mekaar met die drumpelwaarde van 6 nie. Dit kan ook dan die verklaring wees vir die afname in die ooreenkomspersentasie van tipe verandering 8.

Indien ooreenkomste laer as 60% nie beskou word as plagiaat nie word Simian geflous in toetsgeval 1 deur

- vier uit die tien veranderings met die drumpelwaarde 3;
- ses uit die tien veranderings met die drumpelwaarde 6;
- nege uit die tien veranderings met die drumpelwaarde 9; en
- nege uit die tien veranderings met die drumpelwaarde 12.

In Toetsgeval 2 flous slegs twee van die veranderings Simian met alle drumpelwaardes.

Daar sou verwag word dat die ooreenkomspersentasie hoër sal wees in toetsgeval 2 vir die tipes verandering 8 en 10 met die drumpelwaardes 9 en 12 omdat die kodelêer bygevoeg word wat ooreenstemmend met mekaar is en dus ’n groter ooreenkoms moet lewer.

3.10.2 Refleksie oor Simian

Simian beskik oor verskeie opsies, maar nie al hierdie opsies is beskikbaar vir al die programmeringstale nie en die gebruiker moet dus seker maak watter opsies wel gebruik kan word.

Al die programmeringsopdragte wat met mekaar vergelyk moet word, moet in een vouer geplaas word omdat Simian een gids vir die toetsing ontvang en gebruik. Die eerste keer wat Simian laat loop is, is die gids gespesifiseer maar geen ooreenkomste is gekry nie. Die programmeringsopdragte is verander sodat presies dieselfde opdragte in die vouer voorkom maar steeds is geen ooreenkomste verkry nie. Nadat daar nagelees is oor die installering van Simian is daar bevind dat mens saam met die taal moet spesifiseer watter verlenging in die gids voorkom. In die geval van hierdie eksperiment moet .cs aan die einde van die gespesifiseerde gids in die bevelaanporring bygevoeg word.

Die afvoer wat deur Simian gegee word, is verwarrend. Die afvoer gee slegs die lynnommers waar ooreenkomste gevind is en die gebruiker moet self in die programmeringsopdragte gaan soek waar en wat dit is. Die aantal duplikate wat ooreenstem met die gegewe lyne en die totale duplikate wat gevind is, tel nie op tot die selfde aantal nie, dus word van die duplikate

wat verkry is se beskrywing van waar dit gevind is nie weergegee nie. Dit het veroorsaak dat ŉ mens nie geweet het watter aantal duplikate om te gebruik as resultaat nie want Simian het bv. 70 duplikate gevind, maar slegs 59 duplikate se ligging word in die programmeringsopdragte gegee. Nadat ondersoek met die hand ingestel is, is daar gevind dat die aantal duplikate wat Simian heel onder aan die verslag weergee die korrekte aantal duplikate is en dat almal se ligging net nie gegee word nie, maar wel die aantal blokkodes wat ooreenstem.

3.11 Gevolgtrekking

ŉ Opsomming van die ses PPHS'e wat in hierdie studie ondersoek is, word in Tabel 3-14 weergegee. Die algoritme wat deur die PPHS gebruik word, programmeringstale wat die stelsel ondersteun, die soorte toevoerkoppelvlakke, die verslaggewingformaat en tipes lisensie wat die verskillende PPHS'e ondersteun, word kortliks bespreek.

Tabel 3-14 Opsomming van die ses programkodeplagiaatherkenningstelsels

	Algoritme(s)	Programmeringstale	Toevoerkoppelvlak	Verslag/afvoerformaat	Lisensie
AntiCutAndPaste **(ACNP)**		C++, Visual Basic, Delphi, Java, C#.	Grafiese gebruikerskoppelvlak.	Die evalueringpakket laat die gebruiker nie toe om die resultate te stoor nie. Die afvoer word in die grafiese gebruikerskoppelvlak vertoon as een van vyf style (Samesteller, kort, normaal, omslagtig of volledig).	Evalueringspakket - gratis. Onbeperkte weergawe - vanaf $9.95 per lesensie afhangende van die bedryfstelsel.
CodeMatch®	Verklaring-, kommentaar/string-, instruksievolgorde-, identifiseerder-ooreenstemmingalgoritme.	ABAP, ASM-6502, ASM-65C02, ASM-65816, ASM-M68k, BASIC, C, C++, C#, COBOL, Delphi, DRI ASM, Flash ActionScript, Fortran, FoxPro, Go, Java, JavaScript, LISP, Lotus Script, MASM, MATLAB, MPE/iX, Pascal, Perl, PHP, PL/M, PowerBuilder, PowerHouse, Prolig, Python, RealBasic, Ruby, Scala, SQL, TAL, TCL, Verilog, VHDL, Visual Basic.	Grafiese gebruikerskoppelvlak.	HTML-verslag, CLOC-sigblad, verspreidingsigblad, soektogsigblad, opsommende sigblad, PID-notering en XML.	Drie-, ses-maande- of jaarlikse lisensie kan verky word. 'n Projeklisensie - teen $10 per lisensie. Vyftig proeflisensies vir evaluering - gratis. Universiteitprogramlisensie - gratis.
Copy/Paste Detector **(CPD)**	Karp-Rabin stringooreenstemming-algoritme.	Apex, C#, C++, Fortran, Go, Groovy, Java, JavaScript, JSP, MATLAB, Objective-C, Perl, PHP, PL/SQL, Python, Ruby, Scala, Swift.	Bevelaanporring of Grafiese gebruikerskoppelvak.	Teksformaat, XLM-formaat of komma-geskeide-waarde-(.csv) formaat.	Volledige program is gratis verkrygbaar.

Tabel 3-14 Opsomming van die ses programkodeplagiaatherkenningstelsels (vervolg)

	Algoritme(s)	Programmeringstale	Toevoerkoppelvlak	Verslag/afvoerformaat	Lisensie
JPlag	Gulsige String-Teël algoritme.	Java, C, C++, C#, Scheme, natuurlike taalteks.	Bevelaanporring.	HTML-formaat.	Gratis aanlyn.
***Measure of Software Similarity* (MOSS)**	Dokumentvingerafdruk-algoritme genaamd Winnowing.	a8086 assembly, Ada, C, C#, C++, Haskell, FORTRAN, Java, JavaScript, Lisp, MATLAB, MIPS assembly, ML, Modula2, Pascal, Perl, Python, Scheme, Spice, TCL Verilog, VHDL, Visual Basic, HCL2 .	Algemene Lisp-indieningseenheid, Java-weergawe (CMD), PHP-weergawe (CMD), grafiese gebruikerskoppelvlak vir Windows-weergawe.	HTML-formaat.	MOSS skrip is vrylik beskikbaar, daar moet slegs 'n gebruiker-id verkry word deur 'n e-pos te stuur aan MOSS.
Simian	Lyngebaseerde stringvergelyking.	Java, C#, C, C++, Objective-C, JavaScript, COBOL, Ruby, SQL, Lisp, JSP, ASP, HTML, XML, Visual Basic, Swift, Groovy, natuurlike taalteks.	Bevelaanporring.	Teks, xml, emacs, visual studio, yaml.	Nie-kommersiële projekte en evalueringslisensie - gratis. Persoonlike / SOHO lisensie - teen $99. Projek / bou van bedienaarlisensie - teen $499. Ondernemings- en ander lisensies.

HOOFSTUK 4
DIE OMVANG VAN PLAGIAAT

In hierdie hoofstuk word die navorsingsdoelwit om die omvang van plagiaat in programmeringsopdragte te ondersoek bespreek. Die navorsingsmetode, data-analise en die resultate wat deur die studie verkry is, word hier bespreek.

4.1 Navorsingsmetode

4.1.1 Deelnemers

Die deelnemers aan hierdie studie was 287 grafiese gebruikerskoppelvlak I-studente van ŉ universiteit in Suid-Afrika. ŉ Totaal van 266 bruikbare response is verkry, ŉ algehele responskoers van 92.7%. Die deelnemers sluit 249 BSc, 16 BCom en 1 BIng voorgraadse studente in. Uit die 249 BSc-deelnemers is 132 ingeskryf vir BSc in Inligtingstegnologie (IT). Verdere biografiese data van die deelnemers word in Tabel 4-1 weergegee.

Tabel 4-1 Profiel van die respondente (N = 266)

Kriteria	Kategorie	Aantal studente
Geslag	Manlik	183 (68.8%)
	Vroulik	83 (31.2%)
Etniese agtergrond	Wit	186 (69.9%)
	Swart	53 (20.0%)
	Verkies om nie te sê nie	17 (6.4%)
	Bruin	7 (2.6%)
	Indiër	3 (1.1%)
IT as 'n vak op skool gehad	Ja	74 (27.8%)
	Nee	192 (72.2%)
Huidige akademiese jaar	1ste jaar	238 (89.1%)
	2de jaar	25 (9.4%)
	3de jaar	4 (1.5%)
Tydsbestuur	Matig	150 (56.4%)
	Goed	96 (36.1%)
	Sleg	20 (7.5%)
Gemiddelde akademiese prestasie vir programmeringsmodules	75-100%	100 (37.6%)
	50-74%	149 (56.0%)
	0-49%	17 (6.4%)
Het al gehoor van die Turnitin plagiaatherkenningstelsel	Ja	149 (56.0%)
	Nee	117 (44.0%)

4.1.2 Data-insameling en instrument

Beskikbare vraelyste van Chuda *et al.* (2012), Sraka en Kaucic (2009) en Power (2009) is ondersoek waarna nuwe en aangepaste vrae vir die vraelys ontwikkel is om sodoende die omvang van plagiaat te ondersoek. Die vraelys is in Google Forms opgestel en bestaan uit agt biografiese vrae, nege vrae aangaande plagiaat oor die algemeen, twee oop vrae en 16 vrae aangaande plagiaat in programmeringsopdragte. Die vraelys is deur drie deskundiges bestudeer van wie twee op die gebied van IT werksaam is en een op die gebied van statistiek om geldigheid te verseker.

Die hoofonderwerpe waaroor die vraelys handel, is:

- kennis oor plagiaat (wat dit behels, wat word as plagiaat beskou);
- beleid ten opsigte van plagiaat;
- verskeie metodes van programmeringsplagiaat; en
- verskeie metodes van verdoeseling van programmeringsplagiaat.

Tabel 4-3 verskaf ŉ uiteensetting van die vrae wat in die vraelys gevra is, asook ŉ aanduiding van die moontlike antwoordkeuses wat gegee is. Die volledige vraelys word in Bylaag I weergegee.

Voordat die vraelys bekend gemaak is, het die navorser eers klasbesoek gebring aan die grafiese koppelvlakprogrammering I-klas, waar die studente volledig ingelig is oor die studie en die feit dat hul deelname vrywillig is en die data anoniem versamel word. 'n Skakel na die vraelys is daarna deur middel van die universiteit se e-leerstelsel aan die deelnemers beskikbaar gestel. Die skakel na die vraelys was vir ŉ week tot die deelnemers se beskikking, waarna die ontvang van response gesluit het.

Faktoranalise is op die 17 Likert-skaalvrae toegepas om ooreenstemmende veranderlikes as faktore te verklaar. Die 266 response is ondersoek deur gebruik te maak van hoofkomponentfaktoranalise waarvan twee faktore genaamd Kennis_en_inligting en Verskaffing_van_programkode verkry is, asook ŉ verdere vyf enkel items. Die Cronbach α koëffisiënt is vir albei faktore bereken (Tabel 4-2) waar daar gevind is dat dit betroubaar is met $\alpha \geq 0.60$.

Tabel 4-2 Betroubaarheidskoëffisiënte van die faktore

Faktor	Cronbach alpha (α)
Kennis_en_inligting	0.868
Verskaffing_van_programkode	0.688

**Kyk Bylaag J vir die items van elke faktor*

Tabel 4-3 Opsomming van die vraelys se vrae

No.	Vrae	Antwoord tipe
1.	Na my mening het ek genoeg tyd tot my beskikking om my programmeringsopdragte te voltooi.	Likert-skaal
2.	Hoe sal jy jou tydsbestuur beskryf?	Meervoudige keuse
3.	Ek verstaan wat plagiaat behels.	Likert-skaal
4.	Plagiaat is om iemand anders se werk as jou eie voor te hou.	Likert-skaal
5.	Plagiaat kan gesien word as 'n misdaad.	Likert-skaal
6.	Ek het al gehoor van die Turnitin plagiaatherkenningstelsel.	Ja / Nee
7.	Die Turnitin plagiaatherkenningstelsel is van belang om onetiese gedrag te ontbloot.	Likert-skaal
8.	Ek mag kopieer solank ek net erkenning gee.	Likert-skaal
9.	Dosente lig ons in oor plagiaat.	Likert-skaal
10.	Dosente lig ons in oor die gevolge van plagiaat.	Likert-skaal
11.	Die programmeringsmodules se studiegidse bevat 'n verduideliking van plagiaat spesifiek in programmering.	Likert-skaal
12.	Die universiteit beskik oor 'n geskikte plagiaatbeleid.	Likert-skaal
13.	Die universiteit beskik oor 'n geskikte plagiaatbeleid ten opsigte van programmering.	Likert-skaal
14.	Kopiëring van programmering is anders as dié van tekskopiëring.	Likert-skaal
15.	Solank ek die kodering van 'n gekopieerde programmeringsopdrag verstaan, kan ek dit as my eie indien.	Likert-skaal
16.	Ek het 'n programmeringsopdrag gekopieer en as my eie ingehandig.	Meervoudige keuse

17.	Volgens jou, wat is die algemeenste metode om plagiaat te pleeg in programmering?	Meervoudige keuse
18.	Volgens jou, wat is die algemeenste manier om kode te verander om nie uitgevang te word nie?	Meervoudige keuse
19.	In jou opinie, watter ander metodes kan gebruik word om nie uitgevang te word nie?	Oop vraag
20.	As ek 'n gekopieerde program as my eie sou inhandig, sal ek uitgevang word.	Likert-skaal
21.	Ek sal 'n programmeringstoets of -eksamen kopieer en inhandig.	Meervoudige keuse
22.	Ek het vriende/kennisse wie se programme ek kan kopieer.	Likert-skaal
23.	Ek sal vir iemand anders my programmeringsopdragte gee as hulle daarvoor vrae.	Likert-skaal
24.	Ek het al my kode vir 'n medestudent gegee.	Meervoudige keuse
25.	Solank as wat my medestudent die kodering verstaan, kan ek dit maar vir hom/haar gee.	Likert-skaal
26.	Nog enige opmerkings aangaande kopiëring of plagiaat.	Oop vraag
27.	Watter metode(s) het jy gebruik om plagiaat te pleeg?	Meervoudige keuses
28.	Op watter manier(e) het jy te werk gegaan om die kodering te verander?	Meervoudige keuses
29.	Op watter manier(e) het jy te werk gegaan om die kodering aan jou medestudent te verskaf?	Meervoudige keuses

4.1.3 Data-analise

Basiese analise is gedoen deur die gemiddelde en standaardafwykings van die twee faktore en enkel items te bereken. Frekwensies is bereken vir die nie-Likertskaalvrae aangaande die verskeie metodes en ervarings van plagiaat en verdoeseling van plagiaat. Ses groepe is geïdentifiseer wat gebaseer is op geslag, IT as ’n skoolvak, ingeligtheid oor die Turnitin plagiaatherkenningstelsel, tydsbestuur, etniese agtergrond en gemiddelde akademiese prestasie vir programmeringsmodules. Hierdie groepe is vir praktiese en statistiese betekenisvolle verskille tussen die gemiddelde getoets deur gebruik te maak van T-toetsing vir die eerste drie en ’n ANOVA-toets is vir die laaste drie groepe gebruik. Die etniese agtergrond van dié wat verkies om nie te sê nie, Bruin en Indiër is saam gegroepeer as "Ander” om ’n gepaste groepgrootte te verkry.

4.2 Resultate en bespreking

4.2.1 Algehele resultate van die groep

Tabel 4-4 toon die gemiddelde waardes en standaardafwykings van die faktore en items. Die hoogste gemiddelde is die faktor Kennis_en_inligting wat aandui dat die groep verstaan wat plagiaat behels, dat plagiaat as ’n misdaad beskou kan word, die groep is ingelig oor plagiaat deur hul dosente en studiegidse en laastens voel die groep ook dat die universiteit oor ’n geskikte plagiaatbeleid beskik. Die item wat die meeste bydra tot die faktor Kennis_en_inligting is die stelling dat plagiaat is om iemand anders se werk as jou eie voor te hou. Dit dui aan dat die groep presies weet waaroor dit gaan.

Die gemiddelde van die faktor Kennis_en_inligting dui aan dat die studente plagiaat, asook die gevolge daarvan verstaan. Dit word verder uitgelig deurdat die studente voel dat hulle uitgevang sal word indien hulle ’n gekopieerde program indien.

Verdere analise dui aan dat die geslag, etniese agtergrond en die kennis van die Turnitin plagiaatherkenningstelsel ten opsigte van ingeligtheid omtrent plagiaat verskil.

Die laagste gemiddelde van 2.323 is vir die faktor: *Solank ek die kodering van ’n gekopieerde programmeringsopdrag verstaan, kan ek dit as my eie indien*. Hierdie gemiddelde dui daarop dat die groep voel dit is verkeerd, maar daar is steeds ’n onsekerheid. Die standaardafwyking toon dat daar gevalle bestaan waar studente voel dit is tog reg, solank hulle die kodering verstaan. Verdere analise dui daarop dat die response op IT as ’n skoolvak en geslag verskil ten opsigte van hierdie faktor.

Die faktor Verskaffing_van_programkode toon ’n lae gemiddelde wat aandui dat die studente nie vriende/kennisse het wie se programmeringsopdragte hulle kan kopieer nie, dat hulle nie vir iemand anders hulle programmeringsopdragte sal gee nie, maar die item *Solank as wat my medestudent die kodering verstaan, kan ek dit maar vir hom/haar* gee, het ’n hoë gemiddelde wat aandui dat die studente niks verkeerds daarmee sien om die programmeringsopdrag aan ’n medestudent te verskaf nie; hulle moet net die kodering verstaan. Verdere analise dui ook aan dat die etniese agtergrond verskil ten opsigte van die faktor.

Tabel 4-4 Basiese analise van twee faktore en die vyf items

Faktor	Gemiddeldde*	Standaard-afwyking
Kennis_en_Inligting	4.280	0.581
As ek 'n gekopieerde program as my eie sou inhandig, sal ek uitgevang word.	4.050	0.997
Na my mening het ek genoeg tyd tot my beskikking om my programmeringsopdragte te voltooi.	3.703	1.049
Ek mag kopieer solank ek net erkenning gee.	3.391	1.177
Kopiëring van programmering is anders as dié van tekskopiëring.	2.939	1.325
Verskaffing_van_programkode	2.610	0.868
Solank ek die kodering van 'n gekopieerde programmeringsopdrag verstaan, kan ek dit as my eie indien.	2.323	1.113

** Response is gebaseer op die Likert-skaal vanaf 1 tot 5*

In die kwalitatiewe data het die vraag na enige ander opmerkings aangaande plagiaat opgelewer dat twee studente weergee dat plagiaat onwettig is en dat dit ŉ strafbare oortreding is wat daarop dui dat hulle sterk oor die saak voel. In teenstelling daarmee voel ŉ ander student weer dat plagiaat te streng hanteer word:

"Plagiaat vind plaas, maar ek voel dit word baie streng gehanteer. Ek voel eerder studente moet net nul kry in plaas van 'n verhoor".

ŉ Tema wat na vore gekom het uit die kwalitatiewe data is die Aanvaarbaarheid van plagiaat: Sommige studente beskou plagiaat as aanvaarbaar, maar na die mening van sommige studente is plagiaat verkeerd. Die resultate word in Tabel 4-5 weergegee.

Tabel 4-5 Aanvaarbaarheid van plagiaat: Kwalitatiewe data

Tema	Aantal
Aanvaarbaarheid van plagiaat	
- Plagiaat is aanvaarbaar.	6
- Plagiaat is verkeerd.	15
• *"Plagiaat is verkeerd, daar is geen twyfel daaraan nie."* • *"Dit is iets wat ek nooit sal oorweeg nie want dit stel my loopbaan in gevaar."* • *"It should be used, but making sure that the work is understood."*	

4.2.2 Plagiaat en plagiaatverdoeseling

In Tabelle 4-6 hieronder word die frekwensies van nie-Likertskaalvrae weergegee. Twee studente het meer as tien keer programmeringsopdragte gekopieer en ingedien as hul eie. Bietjie meer as die helfte van die studente het nog nooit ’n gekopieerde opdrag ingedien nie.

Tabel 4-6 Frekwensies aangaande plagiaat, metodes en maniere

Kriteria	Kategorie	Aantal studente
Ek het 'n programmeringsopdrag gekopieer en as my eie ingehandig.	Nooit nie	148 (55.6%)
	Een of twee keer	101 (38.0%)
	Drie tot vyf keer	14 (5.3%)
	Ses tot nege keer	1 (0.4%)
	Meer as tien keer	2 (0.8%)
Die algemeenste metode om plagiaat te pleeg in programmering.	Verander die kode	93 (35.0%)
	Kopieer kode vanaf 'n foto	71 (26.7%)
	Kopieer kode vanaf die Internet	45 (16.9%)
	Kopieer kode van 'n klasmaat se opdrag in 'n nuwe projek	34 (12.8%)
	Kopieer die hele program	19 (7.1%)
	Ander	4 (1.5%)
Watter metode(s) het jy gebruik om plagiaat te pleeg?	Kopieer kode vanaf 'n foto wat ek geneem het	60 (50.8%)
	Kopieer kode vanaf die Internet	44 (37.3%)
	Kopieer kode van 'n klasmaat se opdrag in 'n nuwe projek	30 (25.4%)
	Kopieer die hele program	19 (16.1%)
	Ander	10 (8.5%)

Tabel 4-6 Frekwensies aangaande plagiaat, metodes en maniere (vervolg)

Kriteria	Kategorie	Aantal studente
Watter manier(e) het jy te werk gegaan om die kodering te verskaf aan jou medestudent?	Deur die kode af te neem en te stuur oor sosiale media	102 (64.6%)
	Deur die kode op 'n geheuestokkie te laai	63 (39.9%)
	Ander	19 (12.0%)
	Kode aan te stuur deur middel van e-pos	18 (11.4%)
	Deur die kode op 'n cloud te laai (bv. Google drive of Dropbox)	10 (6.3%)
Ek het al my kode vir 'n medestudent gegee.	Een of twee keer	118 (44.4%)
	Nooit nie	108 (40.6%)
	Drie tot vyf keer	23 (8.6%)
	Ses tot nege keer	10 (3.8%)
	Meer as tien keer	7 (2.6%)
Op watter manier(e) het jy te werk gegaan om die kodering te verander?	Verander veranderlikes se name	69 (58.5%)
	Verandering of byvoeging van kommentaar	55 (46.6%)
	Verander die uitleg van die program (GGK)	26 (22.0%)
	Verander net die program se naam	25 (21.2%)
	Geen veranderings nie	17 (14.4%)
	Verander die spasiëring van die kode	14 (11.9%)
	Ander	3 (2.5%)
Die algemeenste manier om kode te verander om nie uitgevang te word nie.	Verander net die program se naam	160 (60.2%)
	Verander die uitleg van die program (GGK)	65 (24.4%)
	Verandering of byvoeging van kommentaar	22 (8.3%)
	Verander veranderlikes se name	10 (3.8%)
	Verander die spasiëring van die kode	9 (3.4%)

Tabel 4-6 Frekwensies aangaande plagiaat, metodes en maniere (vervolg)

Kriteria	Kategorie	Aantal studente
Ek sal 'n programmeringstoets of -eksamen kopieer en inhandig.	Sal dit nie eers oorweeg nie	230 (86.5%)
	Sal dit nie doen nie; ek is te bang ek word uitgevang	28 (10.5%)
	Sal dit doen al is ek bang ek word uitgevang	6 (2.3%)
	Ander	2 (0.8%)
	Sal dit doen as ek die kans kry	0 (0.0%)

Dit is opvallend dat die studente dink die algemeenste metode om plagiaat in programmering te pleeg is om net die kode te verander, maar die metode wat hulle die meeste self gebruik is om die kode te kopieer vanaf ’n foto wat hulle van die kode geneem het. Die kopiëring van die hele program is deur 19 studente aangedui as die algemene metode om plagiaat in programmering te pleeg en 19 studente het hierdie metode gebruik om plagiaat te pleeg.

Vier studente het aangedui dat hulle dink daar bestaan ander metodes wat die algemeenste metode is om in programmering te plagieer. Slegs tien studente het van ander metodes gebruik gemaak as wat deur die vraelys verskaf word, wat dus aandui dat die metodes wat in die literatuur gevind is relatief voldoende was.

Met die snelle ontwikkeling van tegnologie en die resultate wat verkry is vir die metodes wat gebruik word om plagiaat te pleeg, is dit te wagte dat die studente programkode met mekaar deel deur foto’s van die programkode oor sosiale media te stuur. Moontlike redes hoekom hierdie metode so gewild is, is omdat dit vinnig en eenvoudig is en die student hoef geen verdere moeite te doen nie. Indien die programkode werk en die regte afvoer lewer, word dit net afgeneem en aan ’n vriend gestuur waarna die vriend enige iets met daardie inligting kan doen. Nog ’n rede kan wees dat die student wat die foto verskaf, voel dat die medestudent steeds self die kode moet interpreteer en oortik en dus word dit nie as plagiaat beskou nie. Dit is ook maklik vir die studente om ontslae te raak van enige bewyse wat teen hulle gebruik kan word.

Alhoewel tegnologie ontwikkel, beteken dit nie dat dit altyd ten volle benut word nie. Die verspreiding van kode deur dit op ’n *cloud* te laai is slegs deur tien studente gebruik. Dit kan moontlik wees dat die studente nie kennis dra daarvan nie, of dit kan moontlik wees omdat die meerderheid studente eerstejaars is en nie IT op skool gehad het nie.

Daar is 118 studente wat een of twee keer hulle programkode aan ’n medestudent gegee het wat meer is as dié wat erken het dat hulle ’n programmeringsopdrag gekopieer het en ingehandig het as hul eie. Daar is dus ook sewe studente wat meer as tien keer kode aan ’n medestudent gegee het. Hierdie resultate toon dat studente wat die programmeringsopdrag verkry nie noodwendig die kopie net so inhandig nie, omdat die groep bang is hulle word uitgevang. Dit kan ook wees dat die studente wat die opdragte as kopieë ingee nie eerlik was toe hulle die vraelys ingevul het nie.

Meer as die helfte van die studente vermoed dat die algemene manier om plagiaat te verdoesel is om die naam van die program te gaan verander, maar die manier wat die studente aangedui het hulle die meeste gebruik, is om die veranderlikes se name te verander. Die programmeringsopdrag se voorkoms verskil effens van die oorspronklike wat misleidend kan wees indien dit ondersoek word. Die veranderlikes se name kan in ’n japtrap verander word deur middel van die soek- en vervangmetode. Volgens Faidhi en Robinson (1987), benodig die student geen programmeringskennis om hierdie soort programkodeplagiaatverdoeseling toe te pas nie. Dit geld ook vir die verandering of byvoeging van kommentaar wat die tweede meeste voorkom as die manier wat die studente gebruik om programkodeplagiaat te verdoesel.

Die verandering van die uitleg van die program (GGK) kom in die top drie maniere van verandering voor. Hierdie manier word die derde meeste deur die studente gebruik om plagiaat te verdoesel en word deur die studente beskou as die tweede algemeenste manier hoe programmeringsopdragte verander kan word om plagiaat te verdoesel. Verandering van die spasiëring van die kode word die minste deur die studente gebruik en is nie so ’n algemene manier om die kode te verander nie. Dit kan moontlik wees omdat dit tydrowend is en meer programmeringskennis benodig om te verseker dat die program steeds sal uitvoer.

Dit is skrikwekkend om te sien dat 17 studente geen veranderings aan hul gekopieerde programkode gemaak het nie. Dit bevestig dat plagiaat ongemerk deurglip en beklemtoon die belangrikheid van plagiaatopsporing in programmeringsopdragte. Drie studente het aangetoon dat hulle ander maniere gebruik het om hul kodering te verander wat aandui dat die manier wat vanuit die literatuur verkry is relatief voldoende was.

Dit wil voorkom dat 86.5% van die studente dit nie eers sal oorweeg om ’n programmeringstoets of -eksamen te kopieer nie; hulle sal dit nie eers doen al bestaan daar ’n geleentheid nie. Ses studente het aangedui hulle sal ’n programmeringstoets of -eksamen kopieer, maar is te bang hulle word uitgevang. Hieruit kan daar afgelei word

dat die studente programmeringstoetse en -eksamens in ’n ernstiger lig beskou as programmeringsopdragte.

Verdere ondersoek van die kwantitatiewe en kwalitatiewe data het getoon dat die metodes om plagiaat te verdoesel wat in die vraelys gelys is relatief voldoende was (kyk Tabel 4-7) en herhaal word in die kwalitatiewe data wanneer studente gevra is om hul eie opinie te gee oor watter ander metodes gebruik kan word om nie uitgevang te word nie. Daar het twee nuwe metodes van verdoeseling te voorskyn gekom wat in Tabel 4-8 beskryf word.

Tabel 4-7 Herhalende temas wat aangaande metodes van verdoeseling verkry is

Tema	Aantal
Verandering of byvoeging van kommentaar.	23
Verander veranderlikes se name.	17
Verander die spasiëring van die kode.	2
Verander net die program se naam.	7
Verander die uitleg van die program (GGK)	7
Die gelyste metodes is voldoende.	3
• *“Verander die bewoording van die kommentaar.”* • *“Byvoeg van kommentaar.”* • *“Verander die taal van die veranderlikes.”* • *“Verander die veranderlikes.”* • *“Spasiëring van die kode kan verander word.”* • *“Change the name of the program”* • *“Ander uitleg en struktuur van die program.”* • *“Die bogenoemde is die algemeenste metodes*.”	

Tabel 4-8 Nuwe temas wat aangaande metodes van verdoeseling verkry is

Tema	Aantal
Volgorde van die kode te verander.	15
Verandering aan metodes	3
• *“Verander die volgorde van statements en verander die metode van omskakeling bv. int.Parse() in plaas van Convert.toInt(). Byvoeg van kommentaar.”* • *“Verander sover as moontlik die volgorde van die programkode.”* • *“Verander metodes.”*	

Een metode wat vanuit die kwalitatiewe data uitgestaan het, was: “*om ’n demi te vra vir die antwoorde*.” ’n Demi is ’n senior student wat reeds die module voltooi het en wat die dosent gedurende praktiese periodes assisteer. Daar word soms memorandums van die praktiese opdrag aan die demi’s verskaf om die studente gedurende die prakties te

help. Die praktiese periodes is laat in die middae, en studente is moeg en lui en daarom gebeur dit dat demi's studente toelaat om foto's van die memorandum te neem sodat die studente vinniger kan klaar kry sodat die demi kan loop en dis minder moeite as om vir die studente te probeer verduidelik. Hierdie manier is 'n ander benadering wat ondersoek en in ag geneem moet word.

4.2.2.1 Geslag

Verskille wat tussen die geslagte voorkom, is deur middel van 'n T-toets ontleed. Tabel 4-9 toon die faktore en items wat praktiese betekenisvolle verskille wys. 'n Klein praktiese betekenisvolle verskil (d = 0.34) ten opsigte van Kennis_en_inligting is gevind en volgens die gemiddelde voel die vroue dat hulle meer ingelig is en oor meer kennis beskik ten opsigte van plagiaat. Dit kan moontlik wees omdat vroue skynbaar beter luister en aandag gee as die mans. Die vrouestudente stem ook meer saam dat alhoewel die student die kodering verstaan, dit steeds nie korrek is om dit dan net so in te dien as hulle eie nie. Dit lewer 'n medium praktiese betekenisvolle verskil en 'n statistiese betekenisvolle verskil (d = 0.459; p < 0.001).

Tabel 4-9 Verskille gebaseer op geslag

Faktore en items	Gemiddelde		Effekgrootte (d)	p
	Mans	Vroue		
Kennis_en_inligting	4.214	4.426	0.340	<0.050
Solank ek die kodering van 'n gekopieerde programmeringsopdrag verstaan, kan ek dit as my eie indien.	2.491	1.951	0.459	<0.001

4.2.2.2 IT as skoolvak

IT kan op skool as 'n vak in Grade 10 tot 12 by sommige hoërskole in Suid-Afrika gekies word. Met die T-toetse is daar bevind dat daar 'n middelmatige praktiese betekenisvolle verskil (d = 0.622), asook 'n statistiese betekenisvolle verskil (p < 0.001) voorkom tussen die studente wat IT as 'n skoolvak gehad het en diegene wat nie die vak gehad het nie met betrekking tot die tyd wat beskikbaar is om 'n programmeringsopdrag te voltooi (kyk Tabel 4-10). Die studente wat IT as 'n skoolvak gehad het, stem saam dat hulle genoeg tyd het om programmeringsopdragte te voltooi, terwyl die studente wat nie IT as skoolvak gehad het nie neutraler voel. Dit kan wees dat die studente wat IT as 'n skoolvak gehad het meer vertroud is met programmering en dus hulle tyd beter kan bestuur, terwyl die studente sonder IT as skoolvak, programmering nog onder die knie moet kry en dus moet opsoek ens. wat tydrowend kan wees. Die studente wat IT as

skoolvak geneem het, voel dis nie heeltemal so verkeerd om die programkode in te dien nie solank jy dit verstaan. Die gemiddelde en die effekgrootte dui daarop dat daar'n klein praktiese verskil voorkom tussen diegene wat IT as skoolvak gehad het en diegene wat dit nie gehad het nie. Dit kan wees dat die studente wat IT as skoolvak gehad het, geleer is dat indien jy die programkode verstaan, dit voldoende is.

Tabel 4-10 Verskille gebaseer op die studente wat IT as 'n vak op skool geneem het teenoor dié wat nie die vak geneem het nie

Faktore en items	Gemiddelde		Effekgrootte (d)	p
	IT op skool	Nie IT op skool nie		
Solank ek die kodering van 'n gekopieerde programmeringsopdrag verstaan, kan ek dit as my eie indien.	2.622	2.208	0.358	<0.050
Na my mening het ek genoeg tyd tot my beskikking om my programmeringsopdragte te voltooi.	4.176	3.521	0.622*	<0.001

** middelmatige praktiese betekenisvolle verskil (d ≥ 0.5)*

In die kwalitatiewe data is daar ook bevind dat studente voel dat plagiaat minder sal wees as daar meer in ag geneem word dat nie almal IT op skool gehad het nie en ook nie die soort denkwyse van programmeerders het nie. Die response wat verkry is in verband met hierdie tema is:

- *"If IT was taught from the foundation like all students never had it. Plagiarism could be greatly reduced. But instead it is taken that all students have Java and Delphi and they therefore know how to program when it is not the case."*

- *"Beter verduidelikings sal lei tot minder plagiaat vir ander student."*

- *"Baie keer het mens self nie IT op skool gehad nie, en moet tog vriende vra om te help veral as mens die aand voor elf uur moet submit en die werk nie ordentlik verduidelik is nie, want daar word aangeneem almal verstaan programmering, maar dat meer as die helfte nie eers IT op skool gehad nie."*

- *"Die werk is egter van so aard dat sommige studente net nooit dit reg kry nie omdat hulle denkwyse nie die van programmeerder is nie. Dus is daar soms geen ander uitweg om 'n desperate persoon te help met kode nie. Al is dit net vir die blote feit dat hulle kode het wat gebruik kan word vir leerdoeleindes in 'n eksamen. Tog as ek moet kies tussen twee swakhede. Die eerste swakheid: om nie 'n persoon te help nie en bestempel te word as onhulpvaardig. Of iemand te*

help deur van my kode vir hulle te gee. Kies ek die tweede opsie, dis bloot menslik om iemand te help."

4.2.2.3 Turnitin plagiaatherkenningstelsel

Turnitin is 'n internetgebaseerde plagiaatherkenningstelsel wat in 1997 deur iParadigms geskep is. Sommige dosente aan universiteite gebruik hierdie stelsel om te toets of die inhoud van die opdragte wat studente indien uniek is en dus nie geplagieer is nie. Dit is 'n teksgebaseerde plagiaatherkenningstelsel.

'n T-toets is uitgevoer om te bepaal of daar 'n verskil voorkom tussen die studente wat weet van Turnitin en dié wat nie van hierdie stelsel weet nie. Tabel 4-11 toon dat daar 'n klein praktiese betekenisvolle verskil voorkom ten opsigte van die faktor Kennis_en_inligting. Soos verwag kan word, is die studente wat Turnitin ken beter ingelig oor plagiaat vergeleke met dié wat nie Turnitin ken nie. Dit kan moontlik verklaar word deur die feit dat die studente reeds blootgestel is aan 'n plagiaatopsporingsproses wat dus hul kennis verbreed het.

Tabel 4-11 Verskille gebaseer op die studente wat Turnitin plagiaatherkenningstelsel ken teenoor dié wat nie Turnitin ken nie

Faktore en items	Gemiddelde		Effekgrootte (d)	p
	Ken Turnitin	Ken nie Turnitin nie		
Kennis_en_inligting	4.380	4.154	0.374	<0.050

4.2.2.4 Algehele tydsbestuur

Studente is gevra om hulle tydsbestuur te beskryf (goed, matig of sleg). 'n ANOVA-toets is gedoen wat die resultate soos uitgebeeld in Tabel 4-12 opgelewer het en wat 'n praktiese betekenisvolle verskil aandui tussen goeie en slegte tydsbestuur met betrekking tot Kennis_en_inligting aangaande plagiaat. Die studente wat goeie tydsbestuur het, voel beter ingelig en dat hulle plagiaat verstaan.

Tabel 4-12 Verskille gebaseer op tydsbestuur

Faktore en items	Gemiddelde		Effekgrootte (d)	p
	Goeie tyds-bestuur	Slegte tyds-bestuur		
Kennis_en_inligting	4.362	4.135	0.395	0.244

4.2.2.5 Etniese agtergrond

Verskille tussen die etniese agtergronde van studente is met behulp van 'n ANOVA-toets geanaliseer. Die resultate van die ANOVA in Tabel 4-13 toon 'n praktiese betekenisvolle verskil tussen Swart en Wit met betrekking tot die verskaffing van programkode en voldoende tyd om programmeringsopdragte te voltooi, asook 'n praktiese betekenisvolle verskil tussen Swart en ander ten opsigte van Kennis_en_inligting van plagiaat en voldoende tyd om programmeringsopdragte te voltooi.

Daar bestaan 'n middelmatige praktiese betekenisvolle verskil en statistiese betekenisvolle verskil ($d = 0.621$; $p < 0.001$) tussen die Swart respondente en die Wit respondente ten opsigte van beskikbare tyd om programmeringsopdragte te voltooi. Die Wit respondente voel neutraler oor of dit van pas is om programkode te verskaf vergeleke met die Swart respondente wat meer neig na dis verkeerd of hulle sal dit nie doen nie.

'n Middelmatige praktiese betekenisvolle verskil ($d = 0.536$) is verkry met verwysing na Kennis_en_inligting van plagiaat tussen Swart en ander, met die Swart respondente wat voel dat hulle oor meer kennis en inligting aangaande plagiaat as die ander beskik.

Tabel 4-13 Verskille gebaseer op etniese agtergrond

Faktore en items	Gemiddelde		Effekgrootte (d)	p
	Swart	Wit		
Verskaffing_van_programkode	2.346	2.706	0.406	<0.050
Na my mening het ek genoeg tyd tot my beskikking om my programmeringsopdragte te voltooi.	3.226	3.860	0.621*	<0.001
Faktore en items	**Gemiddelde**		**Effekgrootte (d)**	**p**
	Swart	**Ander**		
Kennis_en_inligting	4.419	4.123	0.536*	0.079
Na my mening het ek genoeg tyd tot my beskikking om my programmeringsopdragte te voltooi.	3.226	3.556	0.313	0.362

** middelmatige praktiese betekenisvolle verskil ($d \geq 0.5$)*

14.2.2.6 Akademiese prestasie

In die vraelys is die studente gevra om hulle gemiddelde akademiese prestasie vir programmeringsmodules te gradeer tussen 0%-49%, 50%-74% en 75%-100%. 'n ANOVA-toets is uitgevoer waarvan die resultate in Tabel 4-14 uitgebeeld word. Daar is

praktiese betekenisvolle verskille tussen die groep studente met 0%-49% en 50%-74%, asook die 0%-49% groep en die 75%-100% groep en verder nog die 50%-74% groep en die 75%-100% groep wat betref die item dat studente genoeg tyd het om ŉ programmeringsopdrag te voltooi. Soos verwag, is daar ŉ groot praktiese betekenisvolle verskil (d = 0.866) tussen 0%-49% (druipelinge) en 75%-100% (onderskeidings) waar dié wat onderskeidings behaal sterk voel dat hulle genoeg tyd het om programmeringsopdragte te voltooi.

Die 0%-49% studente voel beter ingelig en beskik oor meer kennis oor plagiaat as die 50%-74% studente.

Tabel 4-14 Verskille gebaseer op die gemiddelde akademies prestasie vir programmeringsmodules

Faktor en items	**Gemiddelde**		**Effekgrootte (d)**	**p**
	0%-49%	**50%-74%**		
Kennis_en_inligting	4.457	4.223	0.398	0.255
Na my mening het ek genoeg tyd tot my beskikking om my programmeringsopdragte te voltooi.	3.059	3.544	0.423	0.148
Faktor en items	**0%-49%**	**75%-100%**	**Effekgrootte (d)**	**p**
Na my mening het ek genoeg tyd tot my beskikking om my programmeringsopdragte te voltooi.	3.059	4.050	0.866**	<0.050
Faktor en items	**50-74%**	**75-100%**	**Effekgrootte (d)**	**p**
Na my mening het ek genoeg tyd tot my beskikking om my programmeringsopdragte te voltooi.	3.544	4.050	0.492	<0.001

** *groot praktiese betekenisvolle verskil (d ≥ 0.8)*

4.2.2.7 Klasgrootte

Soos genoem in Afdeling 1.1, neem die aantal voorgraadse studente wat rekenaar- en inligtingswetenskappe studeer toe wat dit moeiliker maak om vas te stel of studente saamwerk of kopieer. Na aanleiding van die vraag of daar nog enige opmerkings aangaande kopiëring of plagiaat is, is daar in die kwalitatiewe data bevind dat die klasgrootte sommige studente negatief beïnvloed wat tot plagiaat lei.

"It can be difficult not to use other sources when I don't understand what we are doing and feel lost in class. Class size is for me, is the biggest setback in my learning

environment and feel that the amount of students in the class leaves me feeling disconnected."

4.3 Gevolgtrekking

In die studie is daar gevind dat die grafiese gebruikerskoppelvlak-programmeringsstudente aandui dat hulle wel oor die nodige kennis en inligting beskik aangaande plagiaat. Daar kom wel verskille voor waar die vrouestudente, die studente wat Turnitin ken, dié wat goeie tydsbestuur toepas, die Swart respondente en die studente wat hul eie akademiese prestasie tussen 0%-49% beoordeel, aandui dat hulle oor meer kennis beskik en beter ingelig is oor plagiaat.

Daar is ook gevind dat die studente dit nie as verkeerd ervaar om hul programmeringsopdragte aan 'n medestudente te verskaf nie, solank as wat hulle die kodering verstaan. Die meerderheid studente verskaf hul kode aan hul medestudent deur dit af te neem en by wyse van sosiale media te stuur of deur dit op 'n geheuestokkie te laai en vir die persoon te gee.

'n Teenstrydigheid bestaan onder die studente aangaande die aanvaarbaarheid van plagiaat. Sommige studente beskou plagiaat as aanvaarbaar en ander studente beskou dit as verkeerd. 38% van die studente het al een of twee keer 'n programmeringsopdrag gekopieer en ingehandig teenoor 55.6% van die studente wat dit nog nooit gedoen het nie.

Die studie het gevind dat die studente dink die drie algemeenste metodes wat gebruik word om plagiaat in programmering te pleeg is:

1. om die kode te verander;
2. om die kode van 'n foto af te kopieer;
3. om kode van die internet te kopieer.

Die drie metodes wat dié studente gebruik het om plagiaat te pleeg verskil effens van dié hier bo genoem; hulle

1. kopieer kode van 'n foto wat geneem is;
2. kopieer kode van die internet af;
3. kopieer kode van 'n klasmaat se opdrag na 'n nuwe projek.

Die grafiese gebruikerskoppelvlakprogrammeringsstudente het aangedui dat hulle dink die algemeenste manier om programkode te verander om nie uitgevang te word nie is:

- om slegs die program se naam te verander;
- om die uitleg van die program (GGK) te verander; en
- om kommentaar te verander of by te voeg.

Dié studente het die kode deur een van die volgende maniere verander sodat hulle nie uitgevang sal word nie:

- verandering van veranderlike name;
- verandering of byvoeging van kommentaar; of
- verandering van die uitleg van die program (GGK).

Die studie het ook gevind dat die grafiese gebruikerskoppelvlakprogrammeringsstudente aandui dat hulle nie dit eers sal oorweeg om programmeringstoetse of -eksamens te kopieer en in te handig nie. Nie eers een student sal dit oorweeg nie, al ontstaan daar ’n geleentheid.

HOOFSTUK 5
BESPREKING VAN BEVINDINGS, GEVOLGTREKKING EN AANBEVELINGS

5.1 Opsomming van die studie

Die navorsingsverslag bestaan uit vyf hoofstukke en vervolgens word daar eers ŉ opsomming van elke hoofstuk weergegee waarna die bespreking van die bevindings, gevolgtrekking en aanbevelings volg.

In Hoofstuk 1 is aandag geskenk aan die oriëntering van die toename in plagiaat en die erns van die probleem. Verdere oriëntering oor programkodeplagiaat en die behoefte aan ŉ gepaste programkodeplagiaatherkenningstelsel (PPHS) spesifiek vir grafiese gebruikerskoppelvlakprogrammering is bespreek. Dit het gelei tot die navorsingsprobleem, navorsingsdoelwitte en navorsingsmetodes.

In Hoofstuk 2 is ŉ literatuurstudie oor plagiaat en plagiaatherkenning onderneem. Spesifieke aandag is aan programkodeplagiaat en die verskeie verdoeselingmetodes van plagiaat in programmeringsopdragte geskenk.

In Hoofstuk 3 is die uitvoering van die eksperiment waar ses geselekteerde PPHS'e geëvalueer word, bespreek. Die resultate van hierdie eksperiment word ook weergegee.

In Hoofstuk 4 is die opname aangaande die omvang van plagiaat in die grafiese gebruikerskoppelvlak I-klas vervat. ŉ Verslag van die bevindings word gelewer.

In hierdie hoofstuk is daar ŉ bespreking van die resultate wat uit die literatuurstudie, eksperiment en die opname ten opsigte van die navorsingsvrae en doelwitte van hierdie studie verkry is. Die gevolgtrekkings word van die resultate afgelei en daar word ook aanbevelings gemaak op grond van hierdie resultate.

5.2 Bespreking van die bevindings van hierdie studie

Die interpretasie en bespreking van die studie se resultate word volgens die uiteensetting van die navorsingsdoelwitte soos in Afdeling 1.2 weergegee. Die bespreking fokus eksplisieter op die bevindings van die opname aangaande die omvang van plagiaat, en die ses PPHS'e as op die literatuurstudie. Die literatuurstudie is volledig in Hoofstuk 2 bespreek en het gedien as ŉ grondslag vir data-insameling en -ontleding vir die evaluering van die PPHS'e.

’n Opsomming van die bevindings van die literatuurstudie volg.

5.2.1 Navorsingsdoelwit 1: Opsomming van die literatuurstudie

5.2.1.1 Plagiaat

Verskeie definisies aangaande plagiaat is verkry. Hannabuss (2001) se definisie het die meeste voorgekom. Hy definieer plagiaat as die ongemagtigde gebruik of nabootsing van iemand anders se idee, taal of uitdrukkings. Daar is gevind dat ’n definisie van plagiaat van belang is om verwarring onder studente te voorkom (kyk 2.1).

Plagiaat kom in verskeie vorms voor, te wete woord vir woord-, parafrasering-, sekondêre bronne-, vorm van die bron-, idees-, outeurskap- en vertaalplagiaat is as vorms van plagiaat teëgekom (kyk 2.2).

Verskeie redes waarom studente daartoe neig om plagiaat te pleeg is ontdek (kyk 2.3). Die prominentste redes was:

- die studente voel dat plagiaat en wette aangaande plagiaat swak gedefinieer is, asook die gevolge daarvan;
- die studente pas swak tydsbestuur toe;
- die studente swig onder groepsdruk; en
- indien die relevansie van die aktiwiteit nie deur die student verstaan word nie.

Verskeie maniere om plagiaat te hanteer en te vermy is gevind (kyk 2.4).

5.2.1.2 Programkodeplagiaat

Programkodeplagiaat verskil van natuurlike taalteksplagiaat. Programkode is makliker om te kopieer en bevat ’n meer beperkte sintaksis wat dit moeiliker maak om ooreenkomste op te spoor. Nog ’n verskil kom voor in die metodes wat gebruik word om plagiaat op te spoor. ’n Wetenskaplike of amptelike definisie aangaande programkodeplagiaat kom nie te voorskyn nie. Die definisies wat gevind is, is deur akademici vir publikasiedoeleindes geskep (kyk 2.5.1).

Verskeie metodes vir verdoeseling van programkodeplagiaat is gevind (kyk Tabel 2-2). Transformasies en veranderinge word in die programkode aangebring om plagiaat te verdoesel. Hierdie transformasies kan leksikale of strukturele veranderings wees. Leksikale veranderings kan die volgende insluit: herbewoording, verandering van formatering, byvoeging of weglating van kommentaar en verandering van veranderlikes

se name. Hierdie veranderings benodig geen kennis aangaande programmering nie en die veranderings kan in enige geskikte teksskrywer aangebring word. Strukturele veranderings benodig meer kennis van programmering om sodoende die struktuur van die program te verander sonder om enige foute te begaan. Hierdie veranderings kan ekwivalente iterasiestrukture of operatore insluit (kyk 2.5.2).

Verskeie studies het aangedui dat akademiese personeel en studente se houding teenoor plagiaat verskil. Elke instelling is verantwoordelik vir sy eie beleid aangaande plagiaat, die gevolge daarvan, asook hoe dit hanteer word. Die verskillende maniere waarop plagiaat hanteer kan word, kan in vyf kategorieë geklassifiseer word, naamlik straf, vermyding, voorkoming, delegering en opvoeding (kyk 2.6).

5.2.1.3 Plagiaatherkenningstelsels

Sommige PPHS'e kan as 'n attribuut-tellingstelsel of 'n struktuurmetingstelsel geklassifiseer word. Daar bestaan PPHS'e wat nie in een van hierdie twee klasse val nie. 'n Attribuut-tellingstelsel is 'n stelsel wat een of ander eienskap van 'n individuele stelsel meet, dit wil sê die telling van 'n attribuut. 'n Struktuurmetingstelsel is 'n stelsel wat na enige ooreenkomste van die strukturele aspekte van die programkode soek (kyk 2.8).

Verskeie tegnieke wat deur PPHS'e toegepas word om plagiaat op te spoor is gevind. Hierdie tegnieke word as vingerafdrukgebaseerde benaderings en konteksvergelykingstegnieke gekategoriseer. Die konteksvergelykingstegnieke bestaan uit verskeie toepassingsalgoritmes, naamlik stringpassingalgoritmes, parameterpassingsalgoritmes en verdelingsboomvergelykingsalgoritmes (kyk 2.9).

'n Bespreking van die bevindings wat van die empiriese studie verkry is, volg.

5.3 Bespreking van die bevindings van die empiriese studie

5.3.1 Navorsingsdoelwit 2: die omvang van plagiaat in programmeringsopdragte in voorgraadse modules

Daar is gevind dat hierdie groep studente ingelig is aangaande plagiaat en ook verstaan wat dit behels, wat in teenstelling is met Stevens en Stevens (1987), Joy en Luck (1999), Zobel en Hamilton (2002), Sheard *et al.* (2003) en Devlin en Gray (2007) wat gevind het dat studente juis geneig is om plagiaat te pleeg omdat die wette aangaande plagiaat en die gevolge daarvan swak gedefinieer word.

Die groep studente het aangetoon dat hul dit nie as verkeerd beskou om hul programmeringsopdrag aan ’n medestudent te verskaf nie, solank hy/sy die kodering verstaan. Daar is verder gevind dat die grafiese gebruikerskoppelvlakprogrammeringsstudente aandui dat hulle bang is dat hulle uitgevang sal word indien hulle plagiaat pleeg, veral in die geval van ’n programmeringstoets of –eksamen. Ten spyte daarvan is daar gevind dat 38% van die studente al een of twee keer ’n programmeringsopdrag gekopieer het en ingedien het as hul eie (kyk Tabelle 4-4 en 4-6).

Een van die redes waarom studente daartoe neig om plagiaat te pleeg is as gevolg van swak tydsbestuur (kyk Tabel 2-1). 56.4% van die studente het aangedui dat hulle tydsbestuur matig is, 36.1% se tydsbestuur is goed en 7.5% het aangedui dat hul tydsbestuur sleg is. Verder het die studente genoem dat hulle wel genoeg tyd tot hul beskikking het om die programmeringsopdragte te voltooi. ’n Verskil kom wel voor tussen die Swart respondente en Wit respondente, waar die Wit respondente sterker voel dat hulle genoeg tyd het om die programmeringsopdragte te voltooi.

Die studie het gevind dat die studente dink dat die algemeenste metode om te plagieer is om net die programkode van ’n gekopieerde programmeringsopdrag te verander, maar die helfte van die studente in die studie het geplagieer deur die kode van ’n foto wat geneem is te kopieer (kyk Tabel 4-6). Die manier wat die studente in hierdie studie gebruik het om die kodering aan ’n medestudent te verskaf is om ’n foto deur middel van sosiale media te stuur. Hierdie manier om die programkode te verkry, is anders as dié van Cosma en Joy (2008) wat gevind het dat studente iemand anders se programkode steel of die opdrag saam met ’n medestudent as ’n groepsopdrag in plaas van individuele werk voltooi.

Die grafiese gebruikerskoppelvlakprogrammeringsstudente het aangedui dat hulle dink die algemeenste manier om programkode te verander om nie uitgevang te word nie is om net die program se naam te gaan verander. 58.5% van die studente het gerapporteer dat hulle die veranderlikes se name verander het toe hulle geplagieer het. Die veranderlikes se name kan baie gou deur middel van die soek- en vervangmetode verander word. Hierdie manier van programkodeplagiaatverdoeseling is ook verkry deur Whale (1990); Jones (2001a) en Faidhi en Robinson (1987).

Daar sou verwag word dat die studente in die grafiese gebruikerskoppelvlak-programmeringsmodule eerder die uitleg van die program (GGK) sal verander. Die studie het aangedui dat 24.4% van die studente dink dat hierdie verandering die

algemeenste manier is om programkodeplagiaat te verdoesel en 22% van die studente het aangedui dat hulle hierdie metode gebruik het om plagiaat te verdoesel. Hierdie verdoeselingstegniek kom in die top drie voor ten opsigte van die maniere wat die studente te werk gaan om programkodeplagiaat te verdoesel (kyk Tabel 4-6).

Die studie se kwalitatiewe data het die volgende uitgelig:

- die studente verskil aangaande die aanvaarbaarheid van plagiaat (kyk Tabel 4-5);
- die verdoeselingmetodes uit die literatuur was voldoende om in die vraelys op te neem , maar daar is wel twee nuwe metodes ook verkry, naamlik die verandering aan metodes en die volgorde van die programkode (kyk Tabelle 4-7 en 4-8);
- die studente neig daartoe om die demi tydens praktiese periodes vir die antwoorde van die programmeringsopdragte te vra (kyk 4.2.2);
- die studente voel dat indien daar in ag geneem word dat almal nie IT as skoolvak gehad het nie, die plagiaat minder kan wees (kyk 4.2.2.2); en
- die klasgrootte het ŉ negatiewe invloed op die studente wat tot plagiaat kan lei (kyk 4.2.2.7).

5.3.2 Navorsingsdoelwit 3: evalueer huidige geoutomatiseerde programkodeplagiaatherkenners

5.3.2.1 Samestelling en uitwysing van bevindings

In Hoofstuk 3 is elke PPHS, asook die resultate wat deur die verskillende opstellings wat beskikbaar was verkry is, bespreek. Hierdie resultate is gebruik om Tabelle 5-1 tot 5-10 op te stel wat elke PPHS se beste ooreenkomspersentasie vir elk van die tien tipes verandering uitbeeld. Tesame hiermee word die opstellingsopsies verskaf wat hierdie ooreenkomspersentasies gelewer het. Met hierdie resultate kan die verskillende PPHS'e met mekaar vergelyk word.

Uit die studie is daar gevind dat al ses die PPHS'e nie geflous word deur verandering aan kommentaar en indentering nie. ACNP, *CodeMatch*®, JPlag en Simian het 100% ooreenkomspersentasies behaal in toetsgevalle 1 en 2. Die opstellingsopsies wat hierdie persentasies behaal het, word in Tabel 5-1 weergegee.

Die verandering van veranderlikes wat ook die meeste deur die grafiese gebruikerskoppelvlakprogrammeringsstudente gebruik word, word deur JPlag opgespoor met ŉ ooreenkomspersentasie van 100% vir beide toetsgevalle 1 en 2 (kyk

Tabel 5-2). *CodeMatch*® word ook nie om die bos gelei nie en toon 'n hoë ooreenkomspersentasie van 87.50% vir toetsgeval 1 en 97.67% vir toetsgeval 2. ACNP en Simian word nie aanbeveel vir hierdie verandering nie. Vanuit die hoër ooreenkomspersentasie vir toetsgeval 2 is dit duidelik dat met die geval van die verandering van veranderlikes die PPHS (behalwe MOSS) baat as die ontwerplêer bygevoeg word. Dit lei dus tot 'n akkurater ooreenkomspersentasie.

Tabel 5-1 Beste ooreenkomspersentasie deur PPHS behaal vir tipe verandering 1

Tipe verandering 1: Kommentaar en indentering Toetsgeval 1		
PPHS	**Opstelling**	**Ooreenkoms %**
ACNP	Opstelling 3	100.00%
CodeMatch®	Verklaring-Identifiseerder-Instruksievolgorde- algoritme	100.00%
CPD	Drumpelwaarde 3, 6, 9, 12	99.68%
JPlag	Kentekengrootte 3, 6, 9, 12	100.00%
MOSS	M-opsie 3, 6, 9, 12	97.17%
Simian	Drumpelwaarde 3, 6, 9, 12	100.00%
Toetsgeval 2		
ACNP	Opstelling 3	100.00%
CodeMatch®	Verklaring-Identifiseerder-Instruksievolgorde- algoritme	100.00%
CPD	Drumpelwaarde 3, 6, 9, 12	95.44%
JPlag	Kentekengrotte 3, 6, 9, 12	100.00%
MOSS	M-opsie 3, 6, 9, 12	78.58%
Simian	Drumpelwaarde 3, 6, 9, 12	100.00%

Tabel 5-2 Beste ooreenkomspersentasie deur PPHS behaal vir tipe verandering 2

Tipe verandering 2: Veranderlikes Toetsgeval 1		
PPHS	**Opstelling**	**Ooreenkoms %**
ACNP	Opstelling 1	25.18%
CodeMatch®	Identifiseerder-algoritme	87.50%
CPD	Drumpelwaarde 6	63.00%
JPlag	Kentekengrootte 3, 6, 9, 12	100.00%
MOSS	M-opsie 3, 6, 9, 12	97.17%
Simian	Drumpelwaarde 3	10.13%
Toetsgeval 2		
ACNP	Opstelling 1	81.92%
CodeMatch®	Identifiseerder-algoritme	97.67%
CPD	Drumpelwaarde 6	81.35%
JPlag	Kentekengrotte 3, 6, 9, 12	100.00%
MOSS	M-opsie 3, 6, 9, 12	78.46%
Simian	Drumpelwaarde 3	70.93%

Die studie het gevind dat die PPHS wat die geskikste is wanneer die verklarings verander word, is CPD met die opstelling van drumpelwaarde 6 vir toetsgevalle 1 en 2. ACNP, MOSS en Simian word nie aanbeveel in die geval van verklaring verandering met net die kodelêer nie (kyk Tabel 5-3).

Tabel 5-3 Beste ooreenkomspersentasie deur PPHS behaal vir tipe verandering 3

Tipe verandering 3: Verklarings Toetsgeval 1		
PPHS	**Opstelling**	**Ooreenkoms %**
ACNP	Opstelling 1	52.42%
CodeMatch®	Verklaring-algoritme	92.00%
CPD	Drumpelwaarde 6	93.33%
JPlag	Kentekengrootte 3	92.10%
MOSS	M-opsie 3, 6, 9, 12	53.58%
Simian	Drumpelwaarde 3	58.38%
Toetsgeval 2		
ACNP	Opstelling 1	95.50%
CodeMatch®	Verklaring-algoritme	95.58%
CPD	Drumpelwaarde 6	98.17%
JPlag	Kentekengrootte 3	96.05%
MOSS	M-opsie 3, 6, 9, 12	56.83%
Simian	Drumpelwaarde 3	86.80%

CodeMatch® met die identifiseerder-algoritme word aanbeveel wanneer die programmodules verander is. *CodeMatch*® het die hoogste ooreenkomspersentasie van >99% getoon vir toetsgevalle 1 en 2 (kyk Tabel 5-4). CPD en JPlag het ook ŉ hoë ooreenkomspersentasie van >90% vir albei toetsgevalle behaal.

Tabel 5-4 Beste ooreenkomspersentasie deur PPHS behaal vir tipe verandering 4

Tipe verandering 4: Programmodules Toetsgeval 1		
PPHS	**Opstelling**	**Ooreenkoms %**
ACNP	Opstelling 1	78.89%
CodeMatch®	Identifiseerder-algoritme	99.25%
CPD	Drumpelwaarde 6	92.28%
Jplag	Kentekengrootte 3	93.94%
MOSS	M-opsie 3, 6, 9, 12	67.42%
Simian	Drumpelwaarde 3	67.99%
Toetsgeval 2		
ACNP	Opstelling 1	96.21%
CodeMatch®	Identifiseerder-algoritme	99.63%
CPD	Drumpelwaarde 6	95.19%
Jplag	Kentekengrootte 3	96.97%
MOSS	M-opsie 3, 6, 9, 12	63.73%
Simian	Drumpelwaarde 3	90.01%

Die studie het gevind dat *CodeMatch*® aandui dat twee programmeringsopdragte 100% dieselfde is in albei toetsgevalle (kyk Tabel 5-5), alhoewel die stellings in die een opdrag verander is. JPlag dui ook aan dat daar wel ŉ >92% ooreenkoms is.

Tabel 5-5 Beste ooreenkomspersentasie deur PPHS behaal vir tipe verandering 5

Tipe verandering 5: Stellings Toetsgeval 1		
PPHS	**Opstelling**	**Ooreenkoms %**
ACNP	Opstelling 1	81.15%
CodeMatch®	Identifiseerder-algoritme	100.00%
CPD	Drumpelwaarde 6	84.35%
JPlag	Kentekengrootte 3	92.90%
MOSS	M-opsie 3, 6, 9, 12	77.75%
Simian	Drumpelwaarde 3	64.98%
Toetsgeval 2		
ACNP	Opstelling 2	87.53%
CodeMatch®	Identifiseerder-algoritme	100.00%
CPD	Drumpelwaarde 6	92.10%
JPlag	Kentekengrootte 3	96.45%
MOSS	M-opsie 3, 6, 9, 12	68.88%
Simian	Drumpelwaarde 3	88.47%

Weer eens het *CodeMatch*® met die identifiseerder-algoritme die beste ooreenkomspersentasie behaal wanneer daar aan die besluitlogika gepeuter word met ’n ooreenkomspersentasie van 99.58% vir toetsgeval 1 en 99.79% vir toetsgeval 2. JPlag vaar ook nie sleg nie met slegs 1.01% laer as CodeMatchvir toetsgeval 1 en 0.51% laer vir toetsgeval 2 (kyk Tabel 5-6). Al die ander PPHS’e het nie ’n ooreenkomspersentasie onder 80% in toetsgeval 1 verkry nie.

Tabel 5-6 Beste ooreenkomspersentasie deur PPHS behaal vir tipe verandering 6

Tipe verandering 6: Besluitlogika Toetsgeval 1		
PPHS	**Opstelling**	**Ooreenkoms %**
ACNP	Opstelling 1	86.69%
CodeMatch®	Identifiseerder-algoritme	99.58%
CPD	Drumpelwaarde 6	88.55%
JPlag	Kentekengrootte 3, 6	98.57%
MOSS	M-opsie 3, 6, 9, 12	82.29%
Simian	Drumpelwaarde 3	86.57%
Toetsgeval 2		
ACNP	Opstelling 1	96.29%
CodeMatch®	Identifiseerder-algoritme	99.79%
CPD	Drumpelwaarde 6	96.74%
JPlag	Kentekengrootte 3	99.28%
MOSS	M-opsie 3, 6, 9, 12	71.15%
Simian	Drumpelwaarde 3	95.67%

Daar is gevind dat die veranderings op die grafiese gebruikerskoppelvlak (GGK) vir MOSS geflous het wanneer slegs die kodelêer gebruik word. Vir die skuif van die kontroles op die GGK het *CodeMatch*®, JPlag en CPD ’n 100% ooreenkoms verkry vir toetsgeval 1 en CPD het net nie 100% behaal vir toetsgeval 2 nie (kyk Tabel 5-7).

CodeMatch® en JPlag is nie mislei deur die herbenaming van die kontroles in die GGK nie. Hulle het ’n 100% ooreenkoms verkry vir toetsgeval 1. Simian was wel mislei en sy beste ooreenkoms was 41.53% vir toetsgeval 2. Hierdie PPHS word dus nie aanbeveel vir hierdie tipe verandering nie (kyk Tabel 5-8).

Die verandering van teks in die GGK bedrieg nie vir *CodeMatch*® nie. *CodeMatch*® met die identifiseerder-algoritme het weereens ’n 100% ooreenkoms verkry vir toetsgevalle 1 en 2. Die verandering het ook nie JPlag mislei nie en ’n ooreenkomspersentasie van meer as 99% is vir albei toetsgevalle behaal. MOSS is wel onkant betrap deur hierdie verandering (kyk Tabel 5-9).

Die laaste verandering wat die kontroles in 'n GGK rondskuif en herbenaam, het daarin geslaag om Simian te flous. Simian, met opstellingdrumpelwaarde 3 het 'n ooreenkomspersentasie van 17.08% vir toetsgeval 1 verkry en 39.21% vir toetsgeval 2. *CodeMatch*® en JPlag aan die ander kant het vir hierdie verandering 'n 100% ooreenkoms vir toetsgeval 1 getoon, terwyl slegs JPlag weer 100% vir toetsgeval 2 behaal het (kyk Tabel 5-10).

Tabel 5-7 Beste ooreenkomspersentasie deur PPHS behaal vir tipe verandering 7

Tipe verandering 7: Skuif kontroles		
Toetsgeval 1		
PPHS	**Opstelling**	**Ooreenkoms %**
ACNP	Opstelling 1	84.88%
***CodeMatch*®**	Instruksievolgorde-algoritme	100.00%
CPD	Drumpelwaarde 6	100.00%
JPlag	Kentekengrootte 3-12	100.00%
MOSS	M-opsie 3, 6, 9, 12	60.00%
Simian	Drumpelwaarde 3	78.67%
Toetsgeval 2		
ACNP	Opstelling 1	94.84%
***CodeMatch*®**	Instruksievolgorde- algoritme	100.00%
CPD	Drumpelwaarde 6, 9, 12	99.11%
JPlag	Kentekengrootte 3-12	100.00%
MOSS	M-opsie 3, 6, 9, 12	78.58%
Simian	Drumpelwaarde 3	85.58%

Tabel 5-8 Beste ooreenkomspersentasie deur PPHS behaal vir tipe verandering 8

Tipe verandering 8: Herbenamming van kontroles in GGK		
Toetsgeval 1		
PPHS	**Opstelling**	**Ooreenkoms %**
ACNP	Opstelling 1	33.25%
***CodeMatch*®**	Instruksievolgorde-algoritme	100.00%
CPD	Drumpelwaarde 6	82.29%
JPlag	Kentekengrootte 3-12	100.00%
MOSS	M-opsie 3, 6, 9, 12	60.00%
Simian	Drumpelwaarde 3	20.67%
Toetsgeval 2		
ACNP	Opstelling 1	56.39%
***CodeMatch*®**	Identifiseerder-algoritme	93.46%
CPD	Drumpelwaarde 6	92.34%
JPlag	Kentekengrootte 3-12	100.00%
MOSS	M-opsie 3, 6, 9, 12	78.58%
Simian	Drumpelwaarde 3	41.53%

Tabel 5-9 Beste ooreenkomspersentasie deur PPHS behaal vir tipe verandering 9

Tipe verandering 9: Verander teks in GGK		
Toetsgeval 1		
PPHS	**Opstelling**	**Ooreenkoms %**
ACNP	Opstelling 1	96.17%
***CodeMatch*®**	Identifiseerder-algoritme	100.00%
CPD	Drumpelwaarde 6	82.29%
JPlag	Kentekengrootte 3	99.46%
MOSS	M-opsie 3, 6, 9, 12	55.96%
Simian	Drumpelwaarde 3	81.89%
Toetsgeval 2		
ACNP	Opstelling 1	95.11%
***CodeMatch*®**	Identifiseerder-algoritme	100.00%
CPD	Drumpelwaarde 6	91.04%
JPlag	Kentekengrootte 3	99.73%
MOSS	M-opsie 3, 6, 9, 12	76.63%
Simian	Drumpelwaarde 3	87.67%

Tabel 5-10 Beste ooreenkomspersentasie deur PPHS behaal vir tipe verandering 10

Tipe verandering 10: Skuif en herbenaam kontroles in GGK		
Toetsgeval 1		
PPHS	**Opstelling**	**Ooreenkoms %**
ACNP	Opstelling 1	27.56%
***CodeMatch*®**	Instruksievolgorde-algoritme	100.00%
CPD	Drumpelwaarde 6	78.63%
JPlag	Kentekengrootte 3-12	100.00%
MOSS	M-opsie 3, 6, 9, 12	60.00%
Simian	Drumpelwaarde 3	17.08%
Toetsgeval 2		
ACNP	Opstelling 1	53.68%
***CodeMatch*®**	Identifiseerder-algoritme	92.58%
CPD	Drumpelwaarde 6	88.26%
JPlag	Kentekengrootte 3-12	100.00%
MOSS	M-opsie 3, 6, 9, 12	78.58%
Simian	Drumpelwaarde 3	39.21%

5.3.2.2 Opsomming van PPHS-evaluering bevindings

In Tabel 5-11 word 'n opsomming gegee van die PPHS met die gekose opstellingsopsie wat die effektiefste plagiaat vir elke tipe verandering wat in hierdie studie ondersoek is opgespoor het.

Tabel 5-11 Die geskikste PPHS en opstellingsopsie vir elke tipe verandering

Tipe verandering	PPHS	Opstelling
1: Kommentaar en indentering	ACNP	Opstelling 3
	CodeMatch®	Verklaring-Identifiseerder-Instruksievolgorde-algoritme
	JPlag	Kentekengrootte 3, 6, 9, 12
	Simian	Drumpelwaarde 3, 6, 9, 12
2: Veranderlikes	JPlag	Kentekengrootte 3, 6, 9, 12
3: Verklarings	CPD	Drumpelwaarde 6
4: Programmodules	*CodeMatch*®	Identifiseerder-algoritme
5: Stellings	*CodeMatch*®	Identifiseerder-algoritme
6: Besluitlogika	*CodeMatch*®	Identifiseerder-algoritme
7: Skuif kontroles	*CodeMatch*®	Instruksievolgorde-algoritme
	JPlag	Kentekengrotte 3-12
8: Herbenaming van kontroles in GGK	*CodeMatch*®	Instruksievolgorde-algoritme
	JPlag	Kentekengrootte 3-12
9: Verander teks in GGK	*CodeMatch*®	Identifiseerder-algoritme
10: Skuif en herbenaam kontroles in GGK	*CodeMatch*®	Instruksievolgorde-algoritme

Na aanleiding van hierdie studie is daar gevind dat *CodeMatch*® die mees effektiewe resultate teen al tien die verdoeselingstegnieke vir toetsgeval 1 gelewer het en JPlag vir toetsgeval 2. As daar na die gesamentlike gemiddelde verwys wordhet JPlag die mees effektiewe resultaat gelewer. *CodeMatch*® sal ook 'n goeie opsie wees, omdat hierdie stelsel die tweede beste resultate gelewer het vir die totale ooreenkomspersentasie wat slegs met 0.42% verskil van JPlag se gesamentlike gemiddeld (kyk Tabelle 5-12 en 5-13).

JPlag het ook bo 99% ooreenkoms verkry vir toetsgevalle 1 en 2 waar daar spesifiek gefokus word op die verandering van die GGK (tipes verandering 7 tot 10). *CodeMatch*® het 'n 100% ooreenkoms verkry vir toetsgeval 1 waar die GGK verander word.

Tabel 5-12 Gesamentlike resultate van die PPHS vir toetsgevalle 1 en 2

Gesamentlike gemiddeld vir tipe veranderings 1-10	
Toetsgeval 1	
PPHS	**Ooreenkoms %**
CodeMatch®	97.83%
JPlag	97.70%
CPD	86.44%
MOSS	71.13%
ACNP	66.62%
Simian	58.64%
Toetsgeval 2	
JPlag	98.85%
CodeMatch®	97.87%
CPD	92.97%
ACNP	85.75%
Simian	78.59%
MOSS	73.00%

Tabel 5-13 Toetsgevalle 1 en 2 se resultate gekombineer

Gesamentlike gemiddeld vir tipe veranderings 1-10 van albei toetsgevalle	
PPHS	**Ooreenkoms %**
JPlag	**98.27%**
CodeMatch®	**97.85%**
CPD	**89.71%**
ACNP	**76.18%**
MOSS	**72.07%**
Simian	**68.61%**

5.3.3 Navorsingsdoelwit 4: stel riglyne spesifiek vir die hantering van plagiaat in programmering op

5.3.3.1 Riglyne aan dosente

Volgens Harris (1994), is die eerste stap om plagiaat teë te werk voorkoming en die tweede is die opsporing daarvan. Om plagiaat te voorkom, is dit noodsaaklik om eerstens die studente op te voed aangaande plagiaat deur

- die instelling se definisie aangaande plagiaat en programkodeplagiaat bekend te maak en te beklemtoon;
- voorbeelde van plagiaat te voorsien met die regstelling daarvan;
- verskeie strategieë om plagiaat te vermy weer te gee;
- die wyse waarop die instelling plagiaatgevalle hanteer weer te gee; en
- deurlopende ondersteuning te voorsien deur dosente en ander bronne.

Indien moontlik, maak gebruik van programkodeplagiaatherkenningstelsels vir programmeringsopdragte, soos uitgewys deur die tweede stap van plagiaatteëwerking. Dosente het gevind dat indien een of twee studente wel uitgevang word vir plagiaat, en daar opgetree word, dit dikwels tot gevolg het dat die aantal plagiaatgevalle daal. Die studente neem waar dat daar aksie geneem word en dat die plagiaatbeleid nie net ŉ bangmaakstorie is nie.

Indien daar ŉ vermoede is dat die studente se programmeringsopdrag geplagieer is, vra vir die studente om die logika en bewerking van die program te verduidelik.

Indien die programmeringsopdragte geassesseer word, kan daar op die volgende gelet word:

- **Die opdrag se naam/gidsnaam**
  Dikwels is daar dieselfde gidsnaam met presies dieselfde spelfoute of karaktergebruik. Selfs ŉ (1) langs die naam van die gids dui aan dat die gids direk gekopieer is.
- **Die skeppingsdatum van die opdrag**
  Programmeringsopdragte word meestal op dieselfde dag uitgevoer; soms ook op dieselfde tyd. Die datum kan dus in eksamens of toetse waar dit op ŉ spesifieke dag geskep moet word, gebruik word sodat dit nie vroeër geskep is nie. Vir die opdragte kan die jaar in gedagte gehou word sodat studente nie vorige jare se opdragte probeer indien nie.
- **Uitleg van die program**
  Daar kan in beide konsole- en grafiese gebruikerskoppelvlakgebaseerde benaderings gekyk word na die uitleg, skrif, bewoording en algehele voorkoms van die program of daar herhalings voorkom, behalwe as daar vooraf spesifieke uitleg-spesifikasies aan die studente deurgegee is.

Dosente moet ook daarop let dat die assistente wat gedurende praktiese sessies help ook op hoogte is na aanleiding van die plagiaatbeleid en beklemtoon dat die hulpmiddels (bv. memorandum) slegs beskikbaar gestel is om die assistent te help en dus nie net so aan die studente gegee moet word nie.

5.3.3.2 Riglyne aan studente

Die volgende vrae, soos opgestel deur Devlin (2005) kan studente help wat programmeringsmodules neem om vas te stel of hulle in programmeringsopdragte

plagiaat pleeg. Indien die student nie geplagieer het nie, sal die eerlike antwoord op hierdie vrae ‘ja’ wees.

- Het ek die program self gekodeer?
- Indien daar gevra word, kan ek die oplossing aan die dosent verduidelik?
- Het ek die kommentaar self gekodeer?
- Kan ek die doel van elke veranderlike, verklaring, funksie en lus verduidelik?

Soortgelyke vrae kan oor die skriftelike opdragte gevra word, soos byvoorbeeld die handleiding of sagtewareontwerpdokumentasie. Jy het nie plagiaat gepleeg nie indien jy eerlik ‘nee’ antwoord op die volgende vrae:

- Het enige iemand aan my 'n dokument gegee om vanaf te kopieer?
- Het enigiemand anders as ek of lede van my span van dieselfde oplossing gebruik gemaak?
- Het ek 'n ander oplossing gelees om uit te vind wat om te doen, en nie erkenning verleen of nie na die dokument verwys nie?
- Het enigiemand anders as ek of lede van my span bygedra tot die ontwerp van die oplossing?
- Is enige van hierdie skriftelike werk van die Internet gekopieer en nie as sodanig erken nie?

Hierdie bostaande vrae kan deur die studente as riglyne gebruik word vir die opsporing en voorkoming van plagiaat. Vervolgens is ’n voorbeeld van ’n waarskuwing teen programkodeplagiaat wat aan die studente in programmeringsmodules gegee kan word deur insluiting in die studiegids of op die kursus se webwerf in die e-leerstelsel van die universiteit.

5.3.3.3 Waarskuwing teen programkodeplagiaat

Programmeringsopdragte is individuele werk en nie groepsaktiwiteite nie (tensy daar uitdruklik aangedui word dat dit wel ’n groepsaktiwiteit is).

Kopiëring van programkode van ander klasmaats of van ander bronne (byvoorbeeld direk vanaf die Internet, ’n foto of voorgeskrewe studiemateriaal) is ontoelaatbaar. Die bronne kan gebruik word om bewerkingsmetodes beter te verstaan en te dien as voorbeelde. U moet steeds u eie programmeringsmetodes en bewerkings skryf, soos aangedui deur die opdrag.

Dit is nie aanvaarbaar om bestaande programkode bloot oor te tik met ander veranderlike name nie – u behoort in staat te wees om die bewerking of metode weer te gee sonder om die oorspronklike kode woordeliks te herhaal.

Die doel van die opdragte is nie die blote weergee van bestaande programme/kode nie, maar om vas te stel of u oor die vermoë beskik om te programmeer, deur u eie logiese, gestruktureerde en analitiese denke te gebruik om 'n kreatiewe oplossing vir bestaande probleme te bied.

5.4 Aanbevelings

Om te voorkom dat die studente net die veranderlikes se name verander (kyk 4.3), kan daar vooraf vaste benamings vir die veranderlikes aan die studente gegee word wanneer die opdrag aan hulle bekend gemaak word. Hierdie metode kan ook toegepas word op die grafiese gebruikerskoppelvlak waar die dosent die grafiese gebruikerskoppelvlak alreeds opgestel het en aan die studente gee waarna hulle net die programkode moet voltooi. Op hierdie manier kan die nasieners makliker opspoor indien die student veranderings aan die GGK gemaak het. Die dosent kan ook aan die studente bekendmaak dat indien hulle nie die gegewe GGK gebruik nie, dit as plagiaat beskou sal word.

’n Ander moontlike benadering kan wees om ’n programkodeplagiaatherkenningstelsels te verkry. Volgens hierdie studie sal dit belangrik wees om te verseker dat die programkodeplagiaatherkenningstelsels in staat is om te onderskei tussen:

- die verandering van die veranderlikes se name;
- verandering of byvoeging van kommentaar; en
- verandering van die program se uitleg (GGK).

Na aanleiding van hierdie studie kan die dosente dus *CodeMatch*®, JPlag of CPD as PPHS aanskaf wat effektief onderskei tussen die bogenoemde veranderings.

BIBLIOGRAFIE

ACNP, A.S. 2003. AntiCutAndPaste—slashing software maintenance costs. https://www.anticutandpaste.com/anticutandpaste/ Datum van gebruik: 20 Jun. 2017.

Ahtiainen, A., Surakka, S. & Rahikainen, M. 2006. Plaggie: GNU-licensed source code plagiarism detection engine for Java exercises. (*In* Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006 organised by: ACM. p. 141-142).

Aiken, A. 1994. MOSS: a system for detecting software plagiarism. http://theory.stanford.edu/~aiken/moss/ Datum van gebruik: 16 Aug. 2016.

Aiken, A. 2005. MOSS: A system for detecting software similarity. https://theory.stanford.edu/~aiken/moss/ Datum van gebruik: 10 Apr. 2017.

Amir, A., Farach, M. & Muthukrishnan, S. 1994. Alphabet dependence in parameterized matching. *Information Processing Letters*, 49(3):111-115.

Anderson, R.E. & Obenshain, S.S. 1994. Cheating by students: findings, reflections, and remedies. *Academic Medicine*, 69(5):323-332.

Arabyarmohamady, S., Moradi, H. & Asadpour, M. 2012. A coding style-based plagiarism detection. (*In* Interactive Mobile and Computer Aided Learning (IMCL), 2012 International Conference on Interactive Mobile and Computer Aided Learning organised by: IEEE. p. 180-186).

Baker, B.S. 1995a. On finding duplication and near-duplication in large software systems. (*In* Reverse Engineering, 1995. Proceedings of 2nd Working Conference on Reverse Engineering organised by: IEEE. p. 86-95).

Baker, B.S. 1995b. Parameterized pattern matching by Boyer-Moore-Type algorithms. (*In* Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms organised by: ACM. p. 541-550).

Barnhart, R.K. 1988. Chambers dictionary of etymology. Edinburgh: Chambers Harrap Publishers.

Beasley, J.D. 2004. The impact of technology on plagiarism prevention and detection: Research process automation, a new approach for prevention. (*In* Proceedings of "Plagiarism: Prevention, Practice and Policies 2004": Joint Information Systems Committee Plagiarism Advisory Service Conference organised by: Citeseer. p. 28-30).

Belkhouche, B., Nix, A. & Hassell, J. 2004. Plagiarism detection in software designs. (*In* Proceedings of the 42nd Annual Southeast Regional Conference organised by: ACM. p. 207-211).

Bennett, R. 2005. Factors associated with student plagiarism in a post-1992 university. *Assessment & Evaluation in Higher Education*, 30(2):137-162.

Berghel, H.L. & Sallach, D.L. 1984. Measurements of program similarity in identical task environments. *ACM Special Interest Group on Programming Languages Notices*, 19(8):65-76.

Bjorklund, M. & Wenestam, C.G. 1999. Academic cheating: frequency, methods and causes. (*In* European Conference on Educational Research (ECER), Lahti, Finland).

Brandt, D.S. 2002. Techman's techpage: copyright's (not so) little cousin, plagiarism. *Computers in Libraries*, 22(5):39-41.

Burrows, S., Tahaghoghi, S.M. & Zobel, J. 2004. Efficient and effective plagiarism detection for large code repositories. (*In* Proceedings of the Second Australian Undergraduate Students' Computing Conference organised by: Citeseer. p. 8-15).

Burrows, S., Tahaghoghi, S.M. & Zobel, J. 2007. Efficient plagiarism detection for large code repositories. *Software-Practice and Experience*, 37(2):151-176.

Calabrese, R.L. & Cochran, J.T. 1990. The relationship of alienation to cheating among a sample of American adolescents. *Journal of Research & Development in Education,* 23(2):65-72.

Center for Higher Education Trust (CHET). 2013. South African higher education performance indicator data 2009 to 2013. http://www.chet.org.za/data/sahe-open-data. Datum van gebruik: 1 Sep. 2016.

Chuda, D., Navrat, P., Kovacova, B. & Humay, P. 2012. The issue of (software) plagiarism: A student view. *IEEE Transactions on Education*, 55(1):22-28.

Clough, P. 2000. Plagiarism in natural and programming languages: an overview of current tools and technologies. http://ir.shef.ac.uk/cloughie/papers/plagiarism2000.pdf Datum van gebruik: 9 Feb. 2017.

Clough, P. 2003. Old and new challenges in automatic plagiarism detection. http://ir.shef.ac.uk/cloughie/papers/pas_plagiarism.pdf Datum van gebruik: 12 Feb. 2017.

Copeland, T. 2003. Detecting duplicate code with PMD's CPD. http://archive.oreilly.com/pub/a/onjava/2003/03/12/pmd_cpd.html Datum van gebruik: 17 Jun. 2017.

Cosma, G. & Joy, M. 2006. Source-code plagiarism: A UK academic perspective. Paper presented at the 7th Annual Conference of the HEA Network for Information and Computer Sciences, Dublin, Eire, 29- 31 August. http://www.dcs.warwick.ac.uk/report/pdfs/cs-rr-422.pdf Datum van gebruik: 10 Feb. 2017.

Cosma, G. & Joy, M. 2008. Towards a definition of source-code plagiarism. *IEEE Transactions on Education*, 51(2):195-200.

Cosma, G. & Joy, M. 2012. An approach to source-code plagiarism detection and investigation using latent semantic analysis. *IEEE Transactions on Computers*, 61(3):379-394.

Culwin, F. & Lancaster, T. 2001. Plagiarism issues for higher education. *Vine*, 31(2):36-41.

Culwin, F., MacLeod, A. & Lancaster, T. 2001. Source code plagiarism in UK HE computing schools. *Issues, Attitudes and Tools, South Bank University Technical Report SBU-CISM-01-02*.

Davis, S.F., Grover, C.A., Becker, A.H. & McGregor, L.N. 1992. Academic dishonesty: Prevalence, determinants, techniques, and punishments. *Teaching of Psychology*, 19(1):16-20.

Deokate, M.B.V. & Hanchate, D.B. 2016. Software source code plagiarism detection: A survey. *Journal of Multidisciplinary Engineering Science and Technology*, 3(1):3747-3750.

Devlin, M. 2005. Avoiding plagiarism and cheating: a guide for students at Swinburne University of Technology. https://www.swinburne.edu.au/media/swinburneeduau/ current-students/ docs/pdf/plagiarism_guide.pdf Datum van gebruik 10 Feb. 2018.

Devlin, M. & Gray, K. 2007. In their own words: A qualitative study of the reasons Australian university students plagiarize. *High Education Research & Development*, 26(2):181-198.

Donaldson, J.L., Lancaster, A.M. & Sposato, P.H. 1981. A plagiarism detection system. (*In* ACM Special Interest Group on Computer Science Education Bulletin organised by: ACM. p. 21-25).

Faidhi, J.A. & Robinson, S.K. 1987. An empirical approach for detecting program similarity and plagiarism within a university programming environment. *Computers & Education*, 11(1):11-19.

Flint, A., Clegg, S. & Macdonald, R. 2006. Exploring staff perceptions of student plagiarism. *Journal of Further and Higher Education*, 30(02):145-156.

Fredriksson, K. & Mozgovoy, M. 2006. Efficient parameterized string matching. *Information Processing Letters*, 100(3):91-96.

Freire, M., Cebrian, M. & Del Rosal, E. 2007. AC: An integrated source code plagiarism detection environment. (Tegniese verslag: cs.IT/0703136).

Gitchell, D. & Tran, N. 1999. Sim: a utility for detecting similarity in computer programs. (*In* ACM Special Interest Group on Computer Science Education Bulletin organised by: ACM. p. 266-270).

Green, P., Lane, P.C., Rainer, A., Bennett, S. & Scholz, S.B. 2012. Same difference: Detecting collusion by finding unusual shared elements. (*In* Proceedings of the Fifth International Plagiarism Conference, UK: Newcastle. p. 1-25).

Grier, S. 1981. A tool that detects plagiarism in Pascal programs. (*In* ACM Special Interest Group on Computer Science Education Bulletin organised by: ACM. p. 15-20).

Halstead, M.H. 1977. Elements of software science. Vol. 7: Elsevier New York.

Hannabuss, S. 2001. Contested texts: issues of plagiarism. *Library Management*, 22(6/7):311-318.

Harris, J.K. 1994. Plagiarism in computer science courses. (*In*. Proceedings of the conference on Ethics in the computer age organised by: ACM. p. 133-135).

HAT. 2005. 5de uitg. Kaapstad: Pearson Education.

Higbee, J.L. & Thomas, P.V. 2000. Preventing academic dishonesty. *Research and Teaching in Developmental Education*, 17(1):63-66.

Hoad, T.C. & Zobel, J. 2003. Methods for identifying versioned and plagiarized documents. *Journal of the Association for Information Science and Technology*, 54(3):203-215.

Howard, R.M. 2002. Don't police plagiarism: just teach! *Education Digest*, 67(5):46-49.

Idury, R.M. & Schäffer, A.A. 1996. Multiple matching of parameterized patterns. *Theoretical Computer Science*, 154(2):203-224.

Jones, E.L. 2001a. Metrics based plagarism monitoring. *Journal of Computing Sciences in Colleges*, 16(4):253-261.

Jones, E.L. 2001b. Plagiarism monitoring and detection-towards an open discussion. (*In* 7th Annual Consortium for Computing Sciences in Colleges Central Plains Conference, Branson, Missouri: Consortium for Computing Sciences in Colleges. p. 229-236).

Joy, M. & Luck, M. 1999. Plagiarism in programming assignments. *IEEE Transactions on Education*, 42(2):129-133.

JPlag. 2013. Credits: Who, when, why? https://svn.ipd.kit.edu/trac/jplag/wiki/Credits Datum van gebruik: 19 Jun. 2017.

Karp, R.M. & Rabin, M.O. 1987. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249-260.

Kilinc, D., Bozyigit, F., Kut, A. & Kaya, M. 2015. Overview of source code plagiarism in programming courses. *International Journal of Soft Computing and Engineering (IJSCE)*, 5(2):79-85.

Lancaster, T. 2003. Effective and efficient plagiarism detection. London: South Bank University. (Proefskrif – PhD).

Lancaster, T. & Culwin, F. 2004. A comparison of source code plagiarism detection engines. *Computer Science Education*, 14(2):101-112.

Lancaster, T. & Culwin, F. 2005. Classifications of plagiarism detection engines. *Innovation in Teaching and Learning in Information and Computer Sciences*, 4(2):1-16.

Levenshtein, V.I. 1966. Binary codes capable of correcting deletions, insertions, and reversals*. Soviet Physics Doklady,* 10: 707-710.

Liao, M. & Tseng, C. 2010. Students' behaviors and views of paraphrasing and plagiarism in an EFL academic setting. *Pan-Pacific Association of Applied Linguistics,* 1(1):171-193.

Liaqat, A.G. & Ahmad, A. 2011. Plagiarism detection in java code*.* Sweden: Linnaeus University. (Verhandeling – MSc).

Love, P.G. & Simmons, J. 1998. Factors influencing cheating and plagiarism among graduate students in a college of education. *College Student Journal,* 32(4):539-550.

Luquini, E. & Omar, N. 2011. Programming plagiarism as a social phenomenon*.* (*In* Global Engineering Education Conference (EDUCON), 2011 IEEE organised by: IEEE. p. 895-902).

Manchester Metropolitan University. 2014. Plagiarism. https://www.theunionmmu.org/pageassets/your-advice-centre/plagiarism-web.pdf Datum van gebruik: 22 Mrt. 2017.

Marshall, S. & Garry, M. 2005. How well do students really understand plagiarism*.* (*In* Goss, H. (Ed.), Proceedings of the 22nd annual conference of the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE) organised by Queensland University of Technology, Brisbane, Australia. p. 457-467).

Martin, B. 1994. Plagiarism: a misplaced emphasis. *Journal of Information Ethics*, 3(2):36.

Maurer, H.A., Kappe, F. & Zaka, B. 2006. Plagiarism-a survey. *Journal of Universal Computer Science*, 12(8):1050-1084.

Moss. 2014. A system for detecting software similarity. http://theory.stanford.edu/~aiken/moss/ Datum van gebruik: 20 Jun. 2017.

Mozgovoy, M. 2006. Desktop tools for offline plagiarism detection in computer programs. *Informatics in Education-An International Journal* 5(1):97-112.

Mozgovoy, M., Kakkonen, T. & Cosma, G. 2010. Automatic student plagiarism detection: future perspectives. *Journal of Educational Computing Research*, 43(4):511-531.

MyBroadband. 2017. The programming languages in high demand in South Africa. https://mybroadband.co.za/news/software/217814-the-programming-languages-in-high-demand-in-south-africa.html Datum van gebruik: 11 Sep. 2017.

Nakov, P. 2000. Latent semantic analysis of textual data. (*In* Proceedings of the conference on Computer systems and technologies Sofia, Bulgaria organised by ACM. p. 5031-5035).

NWU (Noordwes-Universiteit). 2013. Noordwes-Universiteit: Beleid oor plagiaat en ander vorme van akademiese oneerlikheid en wangedrag, 2P/2.4.3.2:5.

NWU (Noordwes-Universiteit). 2016. Undergraduate student support @ Potchefstroom campus: Plagiarism. http://libguides.nwu.ac.za/information-commons-potch/plagiarism Datum van gebruik: 22 Mrt. 2017.

Oates, B.J. 2005. Researching information systems and computing. London:Sage.

Ober, H., Simon, S.I. & Elson, D. 2013. Five simple rules to avoid plagiarism. *Annals of Biomedical Engineering*, 41(1):1.

Ohno, A. en Murao, H. 2011. A two-step in-class source code plagiarism detection method utilizing improved CM algorithm and SIM. *International Journal of Innovative Computing, Information and Control*, 7(8):4729-4739.

Ottenstein, K.J. 1976. A program to count operators and operands for ansi fortran modules. Computer Sciences Report TR 196, Purdue University.

Parker, A. & Hamblen, J.O. 1989. Computer algorithms for plagiarism detection. *IEEE Transactions on Education*, 32(2):94-99.

Pieterse, V. 2014. Decoding code plagiarism. (*In* Proceedings of the 44th Annual Conference of the Southern African Computer Lecturers' Association (SACLA) organised by University of the Witwatersrand, Johannesburg. p. 25-26).

Plagiarism.org. 2012. What Is Plagiarism? http://www.plagiarism.org/plagiarism-101/what-is-plagiarism Datum van gebruik: 16 Aug. 2016.

PMD Contributors. 2011. Finding duplicate code. http://pmd.sourceforge.net/pmd-4.3.0/cpd.html Datum van gebruik: 16 Jun. 2017.

Power, L.G. 2009. University students' perceptions of plagiarism. *The Journal of Higher Education*, 80(6):643-662.

Prechelt, L., Malpohl, G. & Philippsen, M. 2002. Finding plagiarisms among a set of programs with JPlag. *Journal of Universal Computer Science*, 8(11):1016-1038.

Raffetto, W.G. 1985. The Cheat. *Community and Junior College Journal*, 56(2):26-27.

Rutar, N., Almazan, C.B. & Foster, J.S. 2004. A comparison of bug finding tools for Java. (*In*. Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium organised by: IEEE. p. 245-256).

SAFE Coporation. 2015a. CodeMatch. http://www.safe-corp.com/products_codematch.htm Datum van gebruik: 15 Jun. 2017.

SAFE Coporation. 2015b. CodeMatch Algorithms. http://www.safe-corp.com/CodeMatch_algorithms.htm Datum van gebruik: 14 Jun. 2017.

Saxon, S. 2000. Comparison of plagiarism detection techniques applied to student code: Computer science project (Pt. II). Cambridge: Trinity College.

Schleimer, S., Wilkerson, D.S. & Aiken, A. 2003. Winnowing: local algorithms for document fingerprinting. (*In* Proceedings of the 2003 ACM Special Interest Group on Management of Data International Conference on Management of Data organised by: ACM. p. 76-85).

Schoeman, I.L. & Pieterse, V. 2004. Managing programming assignments in the computer science classroom (*In* 34th Annual Conference of the Southern African Computer Lecturers' Association organised by University of Kwazulu-Natal, Durban. p. 774-783).

Sheard, J., Carbone, A. & Dick, M. 2003. Determination of factors which impact on IT students' propensity to cheat. (*In* Proceedings of the Fifth Australasian Conference on

Computing Education Volume 20 organised by: Australian Computer Society, Inc. p. 119-126).

Simian. 2003. Simian - Similarity Analyser: Purpose. http://www.harukizaemon.com/simian/index.html Datum van gebruik: 25 Jun. 2017.

Sraka, D. & Kaucic, B. 2009. Source code plagiarism. (*In* Proceedings of the 31st International Conference on Information Technology Interfaces organised by: IEEE. p. 461-466).

Stein, B. & Zu Eissen, S.M. 2006. Near similarity search and plagiarism analysis. (*In* 29th Annual Conference of the German Classification Society organised by Germany, Magdeburg: Springer. p. 430-437).

Stevens, G.E. & Stevens, F.W. 1987. Ethical inclinations of tomorrow's managers revisited: How and why students cheat. *Journal of Education for Business*, 63(1):24-29.

Vamplew, P. & Dermoudy, J. 2005. An anti-plagiarism editor for software development courses. (*In* Proceedings of the 7th Australasian Conference on Computing Education Volume 42 organised by: Australian Computer Society, Inc. p. 83-90).

Verco, K.L. & Wise, M.J. 1996a. Plagiarism à la mode: a comparison of automated systems for detecting suspected plagiarism. *The Computer Journal*, 39(9):741-750.

Verco, K.L. & Wise, M.J. 1996b. Software for detecting suspected plagiarism: Comparing structure and attribute-counting systems. *American Society of Civil Engineers*, 96.

Whale, G. 1990. Identification of program similarity in large populations. *The Computer Journal*, 33(2):140-146.

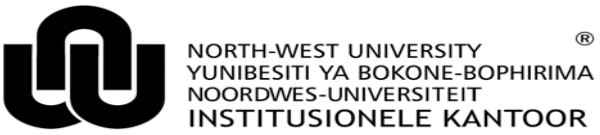
Wilhoit, S. 1994. Helping students avoid plagiarism. *College Teaching*, 42(4):161-164.

Wise, M.J. 1996. YAP3: Improved detection of similarities in computer program and other texts. *ACM Special Interest Group on Computer Science Education Bulletin*, 28(1):130-134.

Zobel, J. & Hamilton, M. 2002. Managing student plagiarism in large academic departments. *The Australian Universities' Review,* 45(2):23.

BYLAAG A

BELEID OOR PLAGIAAT EN ANDER VORME VAN AKADEMIESE ONEERLIKHEID EN WANGEDRAG



Verwysingsnommer	2P/2.4.3.2
Verantwoordelike uitvoerende bestuurder	Institusionele Registrateur
Beleideienaar	Kampusregistrateurs
Verantwoordelike afdeling	Regsdienste
Status	Goedgekeur
Goedgekeur deur	Raad
Datum van goedkeuring	Aanvanklik goedgekeur op 21 September 2007. Hersiening goedgekeur deur die Raad op 10 Junie 2011.
Wysigings	Wysigings is gemaak in 2010, en die gewysigde weergawe vervang die beleid wat in 2007 deur die Raad goedgekeur is.
Datum van wysigings	Hersien in 2011
Hersieningsdatum	Junie 2013
Webadres van hierdie beleid	http://www.nwu.ac.za/opencms/export/NWU/html/gov-man/policy/index.html
Adres op die beleid-databasis	SHARE management\2.1.3 policy development and review\2.1.3.2 review\database/policy documents\2P-2.4.3.2_plagiarism and dishonesty_a.docx

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
INSTITUSIONELE KANTOOR

Beleid oor Plagiaat en ander vorme van Akademiese Oneerlikheid en Wangedrag

1 Aanhef

As 'n toonaangewende universiteit in Afrika, gedryf deur die strewe na kennis en innovasie, met 'n unieke institusionele kultuur gegrond op die waardes wat die Universiteit voorstaan, het die Noordwes-Universiteit hierdie beleid oor Plagiaat en ander vorme van Akademiese Oneerlikheid en Wangedrag op 10 Junie 2011 goedgekeur.

Hierdie beleid vervang die Beleid oor Akademiese Oneerlikheid wat op 21 September 2007 deur die Raad aanvaar is.

2 Beleidsverklaring

'n Universiteit se fokus is op wetenskapsbeoefening, die praktiese toepassing van wetenskaplike dissiplines. Wetenskap word ontdek/nagevors, onderrig, bestudeer en bruikbaar gemaak ten einde legitieme behoeftes te bevredig.

Wetenskaplike kennis word deurlopend versprei deur dié wat onderrig en onderrig ontvang, dié wat leer en dié wat navorsingsresultate lewer. Wetenskaplike kommunikasie lê dus die aktiwiteite van die Universiteit ten grondslag.

Die onderskeid tussen wetenskaplik en vóórwetenskaplik is gesetel in die manier waarop wetenskaplike inligting gekommunikeer word, ongeag of dit in die konteks van leer, onderrig, navorsing of toepassing van kennis is. Wetenskaplike kommunikasie moet dus die toets vir korrektheid, verdedigbaarheid en (veral in die geval van navorsing) oorspronklikheid deurstaan. Die wetenskaplike praktyk van studente en wetenskaplikes word deurlopend beoordeel en geëvalueer. Om hierdie rede is integriteit en dus eerlikheid – benewens voor die hand liggende morele oorwegings – 'n onontbeerlike faktor in wetenskapsbeoefening.

In die geval van enigiemand wat betrokke is by wetenskaplike werk, word daar dus van die betrokke individu verwag om verantwoordelikheid te aanvaar vir die inhoud van sy of haar wetenskaplike kommunikasie. Binne die konteks van onderrig, leer, navorsingsaktiwiteite of die toepassing van kennis is dit onaanvaarbaar om iemand anders se kennis, insigte, bewoording of formulering op enige vlak sonder die nodige erkenning as jou eie aan te bied, en dit kom neer op diefstal en bedrog.

Akademiese oneerlikheid kom neer op ernstige wangedrag, of dit mondeliks, deur optrede of skriftelik gebeur, gedurende eksamens of in die konteks van ander vorms van assessering soos referate, proefskrifte, verslae en publikasies.

Daarom is dit die beleid van die Noordwes-Universiteit dat geen vorm van akademiese oneerlikheid geduld sal word nie, en indien enige sodanige gedrag aangemeld of bespeur word, sal die oortreder ná skuldigbevinding strafbaar wees ingevolge die Universiteit se dissiplinêre beleide, reëls en prosedures. Die Universiteit het die verantwoordelikheid om integriteit en gepaardgaande akademiese eerlikheid by alle studente en personeel in te skerp, veral by diegene in akademiese poste.

3 Woordbepalings

Verskeie aksies wat in verband staan met akademiese wangedrag of akademiese oneerlikheid kan binne die kontekste van eksamenprosesse en ander vorms van akademiese assessering geïdentifiseer word. Dit sal hieronder in meer besonderhede uiteengesit word.

3.1 Eksamenoneerlikheid

Die generiese definisie van eksamenoneerlikheid sluit in:

- Die aanbied vir doeleindes van akademiese assessering van enige materiaal as synde die werk van die persoon wat dit aanbied, terwyl dit elemente bevat van werk wat deur iemand anders verrig is, sonder erkenning aan laasgenoemde.
- Die onregmatige gebruik (d.w.s. sonder die uitdruklike toestemming van die assessor) van akademiese bronne sluit in: eie aantekeninge, ongeag die formaat of modus van sulke aantekeninge (elektronies of handgeskrewe), sowel as skootrekenaars, selfone en ander elektroniese toebehore, programmeerbare sakrekenaars, handboeke en studiegidse sonder die uitdruklike vooraf toestemming van die assessor. Dit kan ook die volgende insluit:
    - Afskryf van ander studente se werk;
    - Besit en gebruik van aantekeninge en ongemagtigde grafiese sakrekenaars, selfone;
    - Vervalsing van identifikasie; en
    - Vervanging van materiaal.
    - Enige optrede deur 'n student wat daarop gemik is om 'n ander student te help om plagiaat of enige ander vorm van akademiese oneerlikheid te pleeg, met inbegrip van oneerlikheid gedurende of met betrekking tot eksamens.

3.2 Ander vorme van akademiese oneerlikheid

Oneerlikheid vind nie net tydens eksamens plaas nie, maar ook in die konteks van ander vorms van assessering, soos referate en ander werksopdragte wat ingedien moet word. Dit sluit in aangeleenthede soos afskryf van iemand anders se werk, en die verbloeming van iemand ander se werk met die doel om voor te gee dat dit die oortreder se eie werk is.

Die volgende kom eweneens neer op akademiese oneerlikheid of wangedrag:

- Die vervalsing van inligting t.o.v. toelatings, uitgestelde assessering of verlof tot afwesigheid, of valse bewerings met die oog op erkenning van gevorderde status;
- Die aanbied van afgeskrewe, vervalsde of onbehoorlik bekomde data as synde die resultaat van navorsing of laboratorium- of veldwerk wat deur die student self onderneem is;
- Die aanbied van individuele werkmateriaal wat die resultaat is van 'n ontoelaatbare en onaanvaarbare hoeveelheid hulp van 'n ander persoon;
- Hulpverlening aan 'n ander student met aanbieding van individuele werk wat strydig is met instruksies of riglyne vir die betrokke werk.

3.3 Plagiaat en kopieregoortreding

Plagiaat is die aanbied, sonder toestemming of bronverwysing, van 'n ander persoon se teks of ander gepubliseerde intellektuele produk deur dit voor te doen as die oorspronklike werk van die persoon wat hoop om voordeel daaruit te trek, of, soos die Oxford Dictionary en The Essential English Dictionary dit stel: "(to) take the work or an idea of somebody else and pass it off as its own" en "to present the ideas or words of another as one's own".

Oortreding van kopiereg is 'n statutêre misdryf wat kan lei tot kriminele vervolging en boetes. Die oortreder kan ook gedagvaar word vir skadevergoeding in 'n siviele geding deur die kopieregeienaar, terwyl plagiaat kan neerkom op ongeoorloofde gedrag ingevolge die gemene reg, wat ook kan lei tot hofgedinge teen die oortreder.

Benewens die regsgevolge is dit onteenseglik dat plagiaat en kopieregoortreding die integriteit van akademisme in diskrediet bring en indruis teen wetenskaplike etiek en die samelewing se persepsie van morele waardes. Dit moet dus verbied word, en waar dit wel voorkom, moet dit as onwettige praktyke deur die Universiteit gestraf word.

Plagiaat kan in meer besonderhede soos volg geïdentifiseer word:

- ’n Voorbedagte daad om iemand anders se werk of eiendom aan te bied sonder verwysing na oferkenning aan die regmatige eienaar, soos:
- gebruik van ’n ander persoon se idees, werk of navorsingsinligting sonder erkenning
- woordelikse oorskrywing van sinne of paragrawe uit een of meer bronne sonder erkenning
- parafrasering van sinne of paragrawe sonder erkenning
- indiening van rekenaarmateriaal, gedeeltelik of in die geheel, sonder om die oorsprong bekend te maak
- vir groepprojekte, die wanvoorstelling van groeplede se individuele bydraes waar individuele insette geïdentifiseer moet word.

4 Reëls

4.1 Verantwoordelikhede

Die Raad en die Senaat is verantwoordelik vir die bestaan en monitering/hersiening van ’n Beleid oor Plagiaat en ander vorme van Akademiese Oneerlikheid en Wangedrag.

Die Universiteit het die verantwoordelikheid om integriteit en akademiese eerlikheid by alle studente en personeel in te skerp, veral by diegene in akademiese poste. Daar kan aan hierdie verantwoordelikheid voldoen word deur die formulering en publikasie van ’n duidelike beleid oor eksamen-oneerlikheid en plagiaat, deur studente van vroeg af te onderrig oor etiese navorsings- en verwysingspraktyke en oor algemene akademiese eerlikheid, deur die prominente aanbringing van waarskuwings teen akademiese oneerlikheid en plagiaat (o.a. in studiemateriaal, fotokopieermasjiene en skandeerders) en deur van studente te verwag om by die inhandiging van onafhanklike akademiese werk vir assessering te bevestig dat sodanige werk inderdaad hul eie oorspronklike werk is (deur middel van die aangehegte **Addendum A**), en deur gepaste strawwe op te lê waar akademiese oneerlikheid en plagiaat aan die lig kom. Die Universiteit het dus ’n opvoedkundige, maatskaplike en dissiplinêre verantwoordelikheid. Elke akademikus en student wat aan die Universiteit verbonde is, het die verantwoordelikheid om alle materiaal wat vir akademiese assessering of navorsingspublikasie bestem is op ’n etiese en wetenskaplik verantwoordbare wyse aan te bied.

Die dosent se verantwoordelikhede in hierdie verband kan soos volg uitgespel word:

- Kennisgewing aan Studente: Studente wat modules in ’n spesifieke Skool neem, moet inligting ontvang oor aanvaarbare standaarde van integriteit en akademiese eerlikheid, bv. met betrekking tot assessering, verwysing, erkenning en bibliografie.
- Alle module-oorsigte moet ’n herinneringsklousule bevat oor plagiaat, akademiese oneerlikheid of wangedrag en outeursregbeskerming.
- Hulle moet studente aan die begin van elke kursus van standaarde vir verwysings en bibliografiese inskrywings voorsien.
- Studente moet gewaarsku word teen die gee of ontvang van hulp met assessering of met werk wat bedoel is om die werk van ’n individuele student te wees.

4.2 Dissiplinêre prosesse

In gevalle van akademiese oneerlikheid is daar twee moontlike vlakke van dissiplinêre optrede: opvoedkundige teregwysing en penalisering in die konteks van die onderrig-leerproses, en die formele dissiplinêre proses.

Teregwysing word in die eerste instansie tydens ’n gesprek deur die betrokke dosent (met inbegrip van ’n studieleier, promotor en assesseerder op enige vlak) hanteer, nadat die direkteur of dekaan oor die saak ingelig is. Die direkteur of dekaan kan besluit om self die saak te hanteer of om dit aan die dosent oor te laat.

Indien ’n dosent weet of vermoed dat ’n student oneerlik opgetree het in die aanbieding van werk, byvoorbeeld vir doeleindes van akademiese assessering deur bv. ’n ander student se werk af te skryf, materiaal uit die werk van ’n ander student, ’n artikel, ’n verslag, dokument of die internet af te skryf of te parafraseer sonder vermelding van die feit dat dit ’n aanhaling is, moet die dosent die student daarmee konfronteer tydens ’n persoonlike gesprek. Die student moet ’n geleentheid gebied word om lig te werp op die insident. Die gesprek moet die dosent in staat stel om:

- die saak af te sluit; of
- die student met of sonder 'n nulpunt te waarsku, en die direkteur en dekaan op hoogte van sake te bring, en rekord te hou van die stappe wat geneem is; of
- 'n aanbeveling te maak by die direkteur of dekaan dat die student formeel aangekla moet word;
- Die dosent moet in elke geval 'n geskrewe verslag aan die direkteur voorlê met geloofwaardige bewyse, ongeag die uitkoms van die saak;
- 'n Student moet die ontvangs van 'n teregwysing of 'n waarskuwing altyd skriftelik bevestig;
- Indien 'n student hom/haar nie by die besluit van 'n dosent wil berus nie, kan hy/sy 'n versoek rig aan die betrokke skooldirekteur dat die saak hersien moet word. Enige verdere versoeke vir hersiening kan eers gemaak word na afloop van 'n formele dissiplinêre proses.

Die *dissiplinêre proses* verloop volgens die Universiteit se Dissiplinêre Reëls vir Studente en Personeel (afhangende daarvan of die betrokke student ook 'n personeellid is), met behoorlike inagname van die volgende oorwegings:

- Die betrokke skooldirekteur doen 'n voorlopige ondersoek na beskuldigings van plagiaat, akademiese oneerlikheid of wangedrag, en lê dan die verslag voor aan die tugkomitee vir studentesake op kampusvlak indien die persoon 'n student is, of aan die Bestuurder: Arbeidsverhoudinge indien die persoon 'n personeellid is.
- Die betrokke dosent mag as getuie geroep word deur die tugkomitee.
- Indien die student/personeellid deur die tugkomitee skuldig bevind word aan plagiaat, akademiese oneerlikheid of wangedrag, word 'n gepaste straf opgelê ingevolge paragraaf 6(1)-(9) van die Universiteit se Dissiplinêre Reëls vir Studente of paragraaf 8 van die Dissiplinêre Reëls vir Personeel. Indien die tugkomitee die student teregwys ingevolge paragraaf 6(10), moet dit met 'n bykomende strafmaatreël gekombineer word.
- Enige versoeke vir hersiening of appèl word volgens die Statuut en die Universiteit se Dissiplinêre Reëls vir Studente en Personeel hanteer.

4.3 Rekordhouding

Skooldirekteurs hou rekords van akademiese teregwysings van studente en alle akademiese teregwysings moet gehou word volgens die reëls van die NWU se Rekordbestuursbeleid. Rekords oor akademiese teregwysings moet in die amptelike studentelêer kom wat deur Akademiese Administrasie/Dienste bygehou word.

Indien 'n tugkomitee 'n student skuldig bevind aan plagiaat, akademiese oneerlikheid of wangedrag, word die uitspraak en opgelegde vonnis op die student se akademiese rekord aangeteken, veral sodat akkurate inligting verskaf kan word aan persone en instellings wat 'n reg het op sodanige getuienis oor 'n student se gedrag tydens sy/haar studieloopbaan.

4.4 Opsporing van gevalle van akademiese bedrog

Akademiese bedrog met betrekking tot toelating, EVL, tweedegeleentheid-assessering of 'n keuringsproses moet onverwyld by die Direkteur: Akademiese Dienste en die betrokke kampusregistrateur aangemeld word, wat dit dan onder die aandag van die Institusionele Registrateur moet bring.

1 **AANHANGSEL A**

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
INSTITUSIONELE KANTOOR

PLEGTIGE VERKLARING

Plegtige verklaring deur student

Ek_____verklaar hiermee dat die skripsie/verhandeling/proefskrif getiteld

_____

_____

wat ek hiermee aan die Noordwes-Universiteit voorlê ter (gedeeltelike) voldoening van die vereistes vir die graad_____, my eie werk is, teksversorg is en nog nie aan enige ander universiteit voorgelê is nie.

Ek verstaan en aanvaar dat die afskrifte wat ingedien word vir eksaminering die eiendom van die Universiteit is.

Handtekening van student_____Universiteitsnommer _____

Onderteken te_____op hierdie_____dag van_____20.....

Verklaar voor my op hierdie_____dag van_____20.....

Kommissaris van Ede: _____

---

2 Verklaring deur studieleier/promotor/navorsingsdirekteur

Die ondergetekende verklaar:

1.1 dat die student 'n goedgekeurde studiemodule vir die betrokke kwalifikasie bygewoon het en dat die werk vir die kursus voltooi is of dat werk gedoen is wat deur die Senaat goedgekeur is;

1.2 dat die student voldoen het aan die minimum studietydperk soos gestel in die jaarboek;

1.3 dat toestemming hiermee aan die student verleen word om sy/haar skripsie, verhandeling of proefskrif in te dien;

1.4 dat registrasie/wysiging van die titel goedgekeur is;

1.5 dat die aanwysing/wysiging van eksaminators gefinaliseer is, en

1.6 dat al die prosedures gevolg is ooreenkomstig die Handleiding vir Nagraadse Studie.

Handtekening van Studieleier/Promotor:_____Datum:_____

Handtekening van Navorsingsdirekteur:_____Datum:_____

Oorspronklike gegewens: Amanda van der Merwe(10935746) SHARE:\2P-2.4.3.2
10 June 2011

Huidige gegewens: Jacoline Jansen van Vuuren(10225676) SHARE management\2.1.3 policy development and review\2.1.3.2 review\database/policy documents\2P-2.4.3.2_plagiarism and dishonesty_a.docx
4 Julie 2011 êerverwysingsnommer:

BYLAAG B

DIE NOORDWES-UNIVERSITEIT SE RIGLYNE AANGAANDE PLAGIAAT

Tree met integriteit op

Plagiaat en hoe om dit te vermy

Alle lede van die akademiese gemeenskap regoor die wêreld, wat die vakdissipline ook al mag wees, is daartoe verbind om kennis te skep en ontdek en idees vryelik uit te ruil. Hierdie ideaal maak staat op die gemeenskaplike begrip van die gedagte van **akademiese eerlikheid**, wat in sy eenvoudigste vorm beteken dat die uitslag van enige navorsing nooit vervals mag word nie en dat **volle erkenning** aan ander se bydraes tot ons eie prestasies, gegee word.

Omdat dit so noodsaaklik is om akademiese integriteit te beskerm, moet 'n aantal konvensies gehoorsaam word om te voorkom dat ons onbedoeld 'n ernstige akademiese oortreding begaan.

Oneerlike akademiese optrede is 'n ernstige oortreding, ongeag of dit mondeling, deur optrede of op skrif plaasvind, tydens eksamens of in die konteks van ander vorms van evaluering soos opdragte, tesisse, verslae en publikasies.

Die beleid van die Universiteit is dat **geen** vorm van akademiese oneerlikheid verdra sal word nie en indien enige sodanige optrede aangemeld of waargeneem word en die oortreder skuldig bevind word, sal hy of sy ingevolge die Universiteit se dissiplinêre beleide, reëls en prosedures gestraf kan word.

Vir die NWU-skakel oor plagiaat, gaan na: *http://www.nwu.ac.za/export/sites/default/nwu/gov_man/policy/2P-2.4.3.2-academic_dishonesty_e.pdf*

'n Vorm van akademiese wangedrag

Tradisioneel word plagiaat omskryf as die neem van woorde, beelde, idees, ens. van 'n skrywer en die aanbieding daarvan as jou eie. Dit word dikwels met frases soos "die roof van woorde", "die roof van idees", "bedrog", en "literêre diefstal" verbind.

Plagiaat kan op 'n verskeidenheid maniere manifesteer en word nie tot studente se geskrifte of gepubliseerde artikels of boeke beperk nie.

Met die opkoms van die internet is dit natuurlik nie net gedrukte werk wat gebruik kan word nie. *Die knip en plak vanaf webblaaie word in hoër onderwys op sigself as plagiaat beskou indien die webblaaie nie behoorlik erken en aangehaal word nie.*

> die oorneem van idees, metodes of geskrewe woorde van 'n ander sonder erkenning daarvan en met die doel dat dit as die werk van die bedrieër beskou moet word.
>
> Vertaalde definisie van *American Association of University Professors (September/Oktober, 1989).*

Maar wat daarvan as groot stukke werk gekopieer word, aanhalingstekens gebruik word en al die stukke akkurate verwysings kry en dit met die student se eie menings gekoppel word?

Op die vlak van hoër onderwys word daar van jou verwag om jou **eie** stem en menings te ontwikkel en op ander mense se werk voort te bou, eerder as om daaragter te skuil. Dit sou dus as swak akademiese praktyk beskou word, maar nie as plagiaat nie.

"Maar dit was nie my bedoeling nie. . ."

Wat die bron van die materiaal of die beoogde uitkoms ook al is, plagiaat is kullery en onaanvaarbaar. Plagiaat is die woord wat toegewys word aan 'n spesifieke soort akademiese oneerlikheid – om voor te gee dat iemand anders se werk, idees of woorde jou eie is. Dit kan soms onopsetlik wees, veral wanneer jou vorige onderwyservaring dalk aktief 'n benadering tot kursuswerk aangemoedig het wat daarop konsentreer om materiaal aan die hand van buitebronne saam te stel.

Dit is belangrik dat jy moet verstaan dat opset nie 'n rol in die definisie van plagiaat speel nie. Dit is dus NIE 'n verskoning dat jy nie BEDOEL het om dit te doen, of nie GEWEET het jy doen dit nie – en noudat jy hierdie stuk gelees het, weet jy wat plagiaat is.

Plagiaat kan die vorm aanneem van die herhaling van iemand anders se woorde, of selfs die aanbied van iemand anders se gedagtegang asof dit jou eie is. Kortom, om plagiaat te pleeg, is om die indruk te skep dat jy iets geskryf of gedink het wat jy in werklikheid by iemand anders geleen het. Hoewel 'n skrywer ander mense se woorde en gedagtes kan gebruik, moet dit as sodanig **erken** word.

(Vertaling uit *Modern Languages Association (1977). Handbook for writers of research papers, theses and dissertations New York: MLA 5)*

Die skrywer erken met dank die werk van die *UK Centre for Legal Education*, Pauline Ridley, *University of Brighton*, en die Universiteit van Pretoria se *Plagiarism Prevention Policy* oor die onderwerp van akademiese plagiaat.

Verdere leiding en hulp

Die doel van hierdie riglyne is om basiese algemene inligting oor die aard van plagiaat te verskaf en bewustheid te skep. Dosente kan verdere leiding oor behoorlike verwysings van gepubliseerde werke en webwerwe verskaf, asook oor enige ander vakspesifieke konvensies in hulle dissipline.

Daar is ook 'n magdom advies beskikbaar in die biblioteek en op die internet om jou te help om die beginsels van behoorlike verwysings te begryp en dit toe te pas.

Die sleutel
om plagiaat te vermy

Bly ingelig! Jy wil nie van plagiaat beskuldig word nie, dus:

- Leer hoe om in die styl van jou dissipline te skryf. Jou skryfwerk moet **jou** skryfwerk wees.
- Leer om krities en onafhanklik te dink. Lesers stel in **jou** begrip van 'n idee belang. Skryf is 'n waardevolle oefening wat jou vermoë toets om 'n onderwerp te verduidelik. Dit is 'n belangrike deel van leer.
- Gee altyd die nodige erkenning vir verwysings wat gebruik word. Enige eties verantwoordelike skrywer erken **altyd** die bydraes van ander en die bron van sy/haar idees.
- Enige verbatim teks van 'n ander skrywer wat gebruik word, **moet** tussen aanhalingstekens geplaas en akkuraat aangehaal word.
- Erken altyd elke bron wat jy in jou skryfwerk gebruik, of jy dit parafraseer, opsom, of tussen aanhalingstekens plaas.

- Wanneer jy ander se werk parafraseer en/of opsom, gee die presiese betekenis van die ander skrywer se idees of feite in jou eie woorde en sinstruktuur weer.
- Verantwoordelike skrywers het 'n etiese verantwoordelikheid teenoor lesers en die skrywers van wie hulle leen om ander se idees en woorde te respekteer en erkenning te gee aan diegene van wie hulle leen – en om, waar moontlik, hulle **eie** woorde te gebruik wanneer hulle parafraseer.

Wees slim!

Maak seker dat jy die definisie van plagiaat ten volle begryp en dat jy vertroud is met beleide en regulasies wat op plagiaat betrekking het.

Hoewel plagiaat 'n ernstige akademiese oortreding is, is jy op die regte spoor as jy duidelik, versigtig en eerlik is. Moenie dat vrees vir plagiaat voorkom dat jy die ryk hulpbronne wat beskikbaar is, ten volle benut nie.

Turnitin.com en *Research Resources* verskaf 'n kontrolelys om plagiaat te vermy.* Gaan kyk by *http://www.plagiarism.org/resources/documentation/plagiarism/learning/preventing_plagiarism_students.doc*

Die kern van die akademiese lewe is om te leer hoe om beskikbare werk behoorlik en verantwoordelik te gebruik om jou **eie** begrip van 'n onderwerp te ontwikkel. Die lees van goeie akademiese werk behoort jou ook nuttige voorbeelde en modelle van goeie praktyk te gee. Jy moet aktief soek na maniere waarop dit jou kan help om jou **eie** skryfwerk te verbeter.

'n Laaste woord: as jy twyfel, **vra**!

** Turnitin laat toe dat hierdie dokument gratis versprei en vir gebruik sonder 'n winsoogmerk in opvoedkundige omgewings aangewend word.*

Moenie

met jou toekoms dobbel nie

Plagiaat *is 'n* oortreding

Innovasie deur diversiteit ™

NORTH-WEST UNIVERSITY ®
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
POTCHEFSTROOMKAMPUS

BYLAAG C
TEGNIESE ASPEKTE VAN ACNP

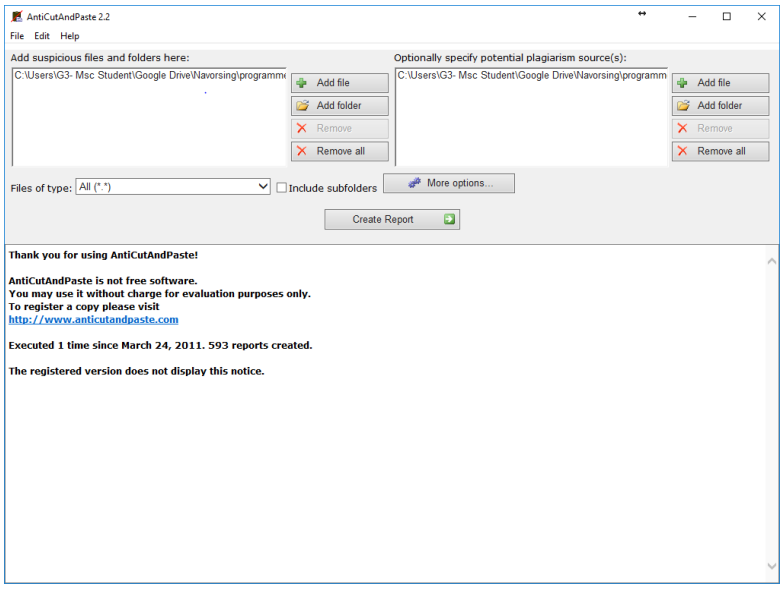
1 Opstelling van ACNP

ACNP kan gratis verkry word vanaf die webtuiste: https://www.anticutandpaste.com/download/ as 'n evalueringspakket vir 'n maksimum van 15 dae, waarna 'n bestelling vir die produk teen 'n koste geplaas moet word. ACNP is beskikbaar vir Windows- of Linux-bedryfstelsels. Die ACNP-evalueringspakket word afgelaai as 'n installasielêer (.exe). Die installasie van ACNP maak gebruik van 'n opstellingsassistent grafiese gebruikerskoppelvlak wat die installasieproses vergemaklik (kyk Figuur 1).

Figuur 1: ANCP se installasiekoppelvlak

2 Uitleg van ACNP

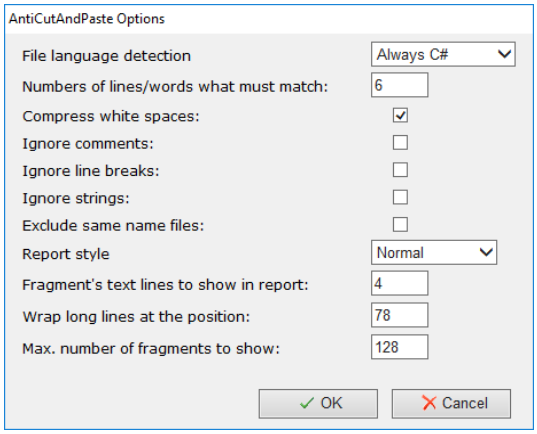
Die ACNP maak gebruik van 'n grafiese gebruikerskoppelvlak (Figuur 2) om toevoer te ontvang en die afvoer weer te gee aan die gebruiker.



Figuur 2: ANCP se grafiese gebruikerskoppelvlak

Die programme wat gebruik kan word gedurende die toetsing, kan as 'n spesifieke lêer of as 'n hele vouer geselekteer word. Meer as een lêer of vouer kan bygevoeg word tot die gekose afdeling (verdagte lêers / potensiële plagiaatbronne).

ANCP het ook verskillende opsies wat gestel kan word voor die toetsing van programmeringsopdragte plaasvind (kyk Figuur 3).



Figuur 3: Die opstellingsopsies van ANCP

Die ***file language detection*-opsie** laat die gebruiker toe om te selekteer watter programmeringstaal gebruik word vir die toetsing. Op hierdie manier hoef die program nie staat te maak op die uitbreidingnaam van 'n lêer nie. Indien die verstek-opsie gebruik word, moet die program die uitbreidingsnaam van die lêer gebruik om af te lei watter programmeringstaal ter sprake is. Indien die uitbreidingnaam onbekend is, word die lêer uit die toetsing gegooi.

Die ***number of lines/words what must match*-opsie** stel die gebruiker in staat om te spesifiseer hoeveel soortgelyke fragmente geïgnoreer moet word, indien hulle minder is as die gegewe aantal lyne/woorde. Die verstekwaarde is agt met 'n minimumwaarde van vier en maksimumwaarde van 64.

Die ***compress white spaces*-opsie** stel die PPHS in staat om die leë spasies tussen fragmente te verwyder, sodat toetsing gedoen kan word op fragmente, alhoewel die aantal spasies, indentering of leë lyne tussen hulle verskil.

Die ***ignore comments*-opsie** stel die program in staat om fragmente met mekaar te vergelyk al bevat die lêer programmeringskommentaar tussen die fragmente.

Die ***ignore line breaks*-opsie** laat die program toe om fragmente met mekaar te vergelyk alhoewel dit verskillende aantalle lyne bevat.

Die ***ignore strings*-opsie** stel die program in staat om twee fragmente met mekaar te vergelyk al kom daar programmeringskonstantes in die lêer voor.

Die ***exclude same name files*-opsie** laat die PPHS toe om lêers wat dieselfde name bevat nie in te sluit nie.

Die ***report style*-opsie** bevat vyf verskillende wyses waarop die afvoer aan die gebruiker voorgehou word.

1. ***Compiler style*** – elke ooreenkomste wat verkry is, word weergegee in een lyn. Lyk soos die programmeringstaal C se waarskuwingslys.
2. ***Brief –*** *vertoon slegs die lêername en die aantal ooreenstemmende karakters wat verkry is sonder enige fisiese fragmente.*
3. ***Normal*** – vertoon die lêername en lynnommers met die eerste en laaste lyntjie teksfragmente wat verkry is.
4. ***Verbose*** – vertoon die normale afvoer met 'n lys van uitgesluite lêers en 'n verduideliking van die verskillende opsies wat geselekteer is.
5. ***Full*** – vertoon alle moontlike besonderhede, insluitende volledige teksfragmente en alle lynnommers sonder beperkings.

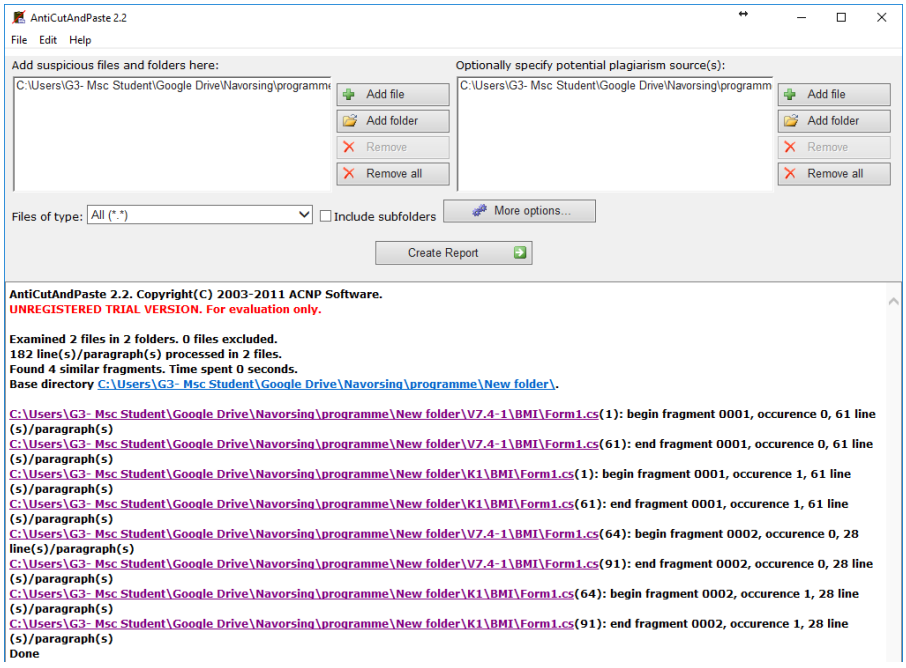
Die ***fragment's test lines to show in report*-opsie** laat die gebruiker toe om te spesifiseer hoeveel toetslyne vanaf die begin tot die einde van die opgespoorde fragmente moet vertoon.

***Wrap long lines at the position*-opsie** laat die gebruiker toe om te spesifiseer by watter posisie die teks moet oorvloei na die volgende lyntjie.

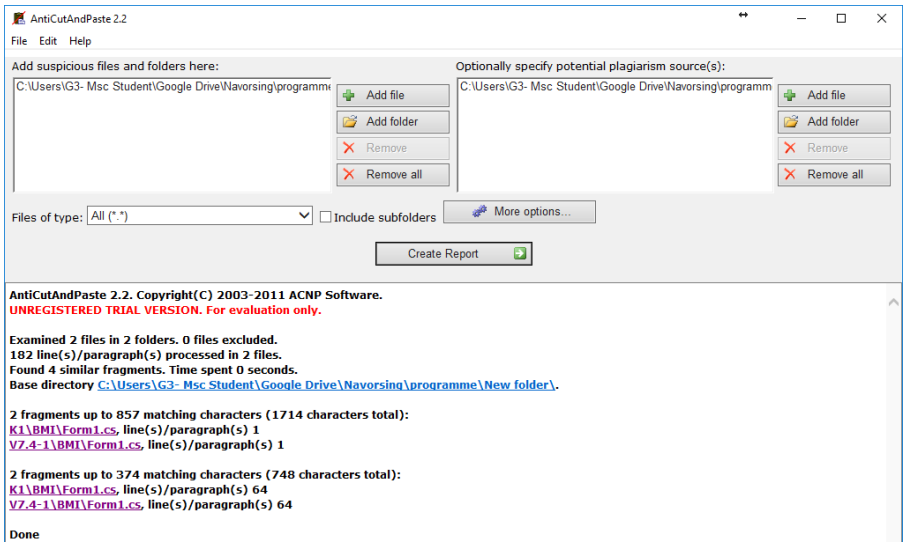
Die ***max. number of fragments to show*-opsie** laat die gebruiker toe om te spesifiseer wat die maksimum aantal fragmente is wat kan voorkom in die verslag.

3 Afvoer/verslaggewing van ACNP

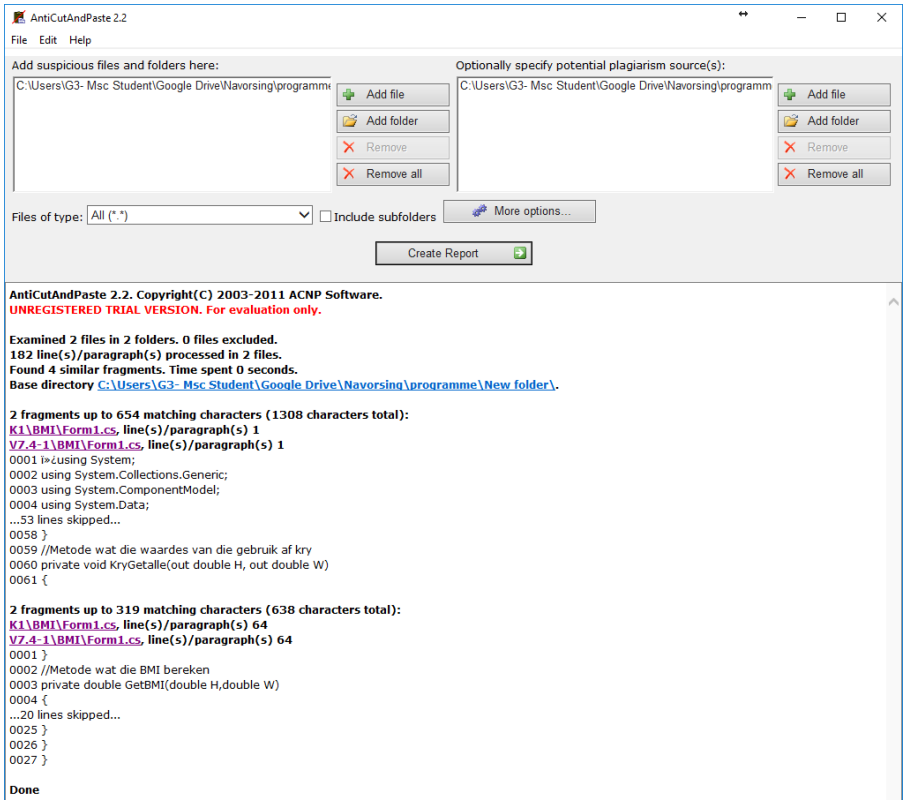
Soos bespreek in 2, beskik ACNP oor vyf verskillende verslagstyle. Hierdie verslae word direk weergegee in die grafiese koppelvlak nadat die *Create report*-knoppie geklik is. Die gekoopte weergawe laat die gebruiker toe om hierdie verslae te stoor. Met die evalueringsweergawe kan die teks nie eers gekopieer word nie, slegs skermafdrukke kan geneem word. Die verslae gee inligting oor die ooreenkomste wat verkry is en moet dus bestudeer word deur 'n kenner om afleidings te maak. Voorbeelde van hierdie vyf verskillende verslae wat deur ACNP gegenereer word, word weergegee in Figuur 4 tot Figuur 8.



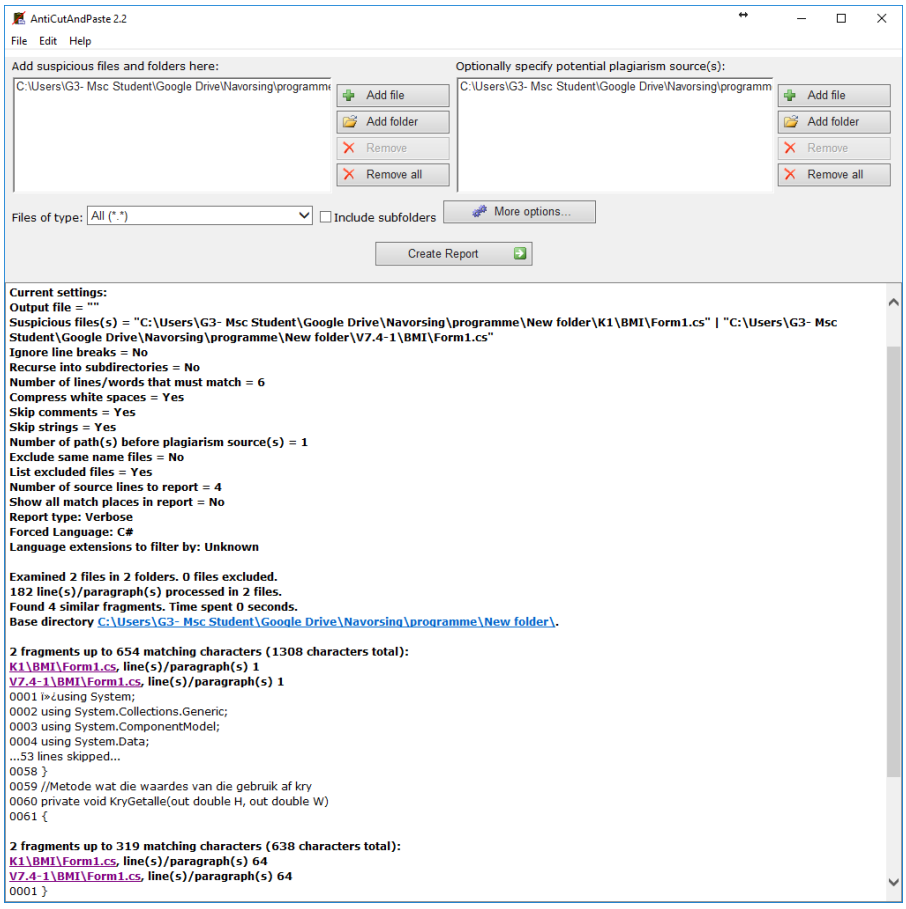
Figuur 4: Voorbeeld van die *Compiler* verslagstyle van ACNP



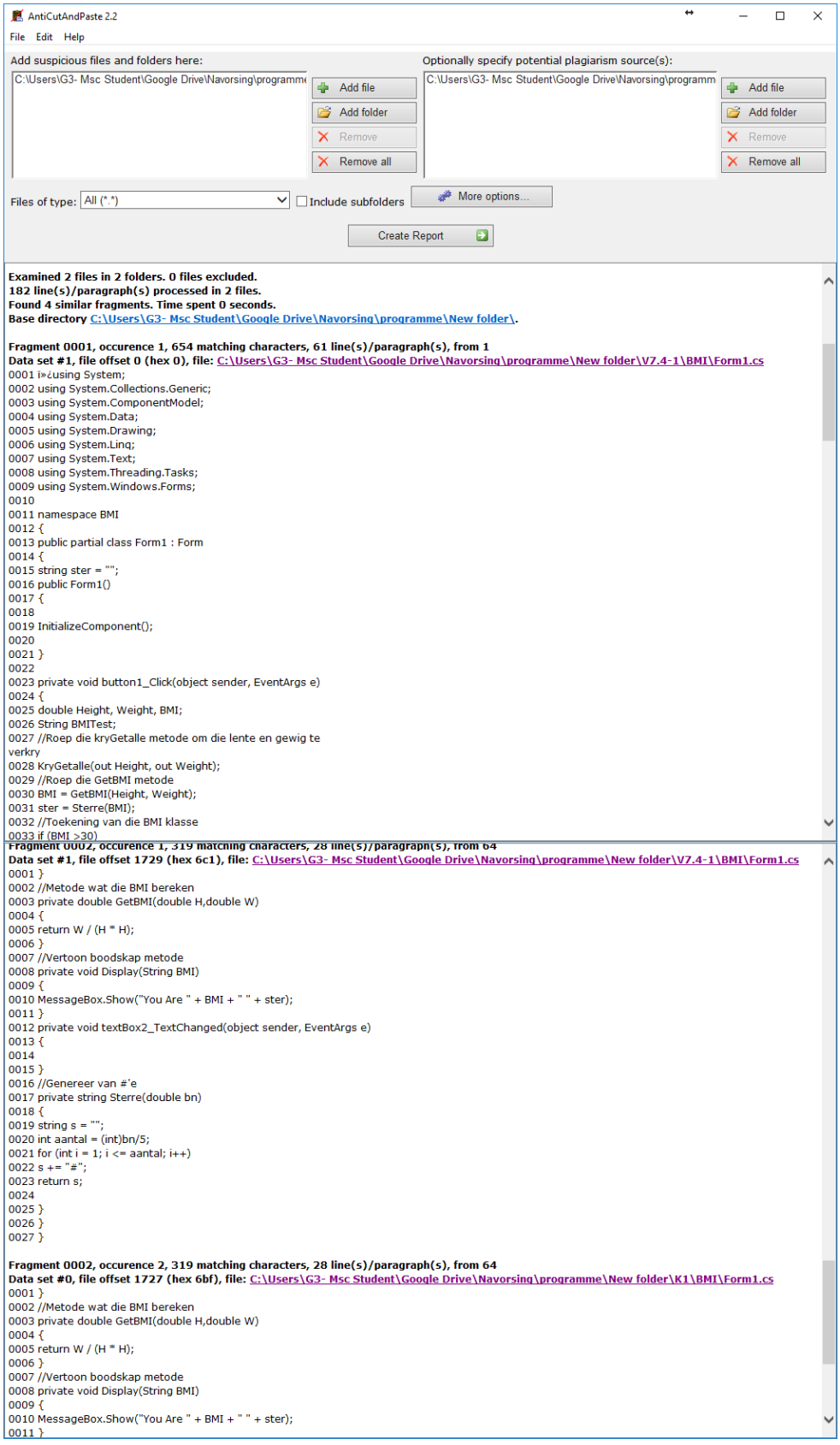
Figuur 5: Voorbeeld van die *Brief* verslagstyle van ACNP



Figuur 6: Voorbeeld van die *Normal* verslagstyle van ACNP



Figuur 7: Voorbeeld van die *Verbose* verslagstyle van ACNP



Figuur 8: Voorbeeld van die *Full* verslagstyle van ACNP

BYLAAG D

TEGNIESE ASPEKTE VAN *CODEMATCH*®

1 Verkryging en opstelling van *CodeMatch*®

Die *CodeSuite*® PPHS-pakket is nie gratis beskikbaar nie. Verskillende opsies is beskikbaar om 'n lisensie te verkry:

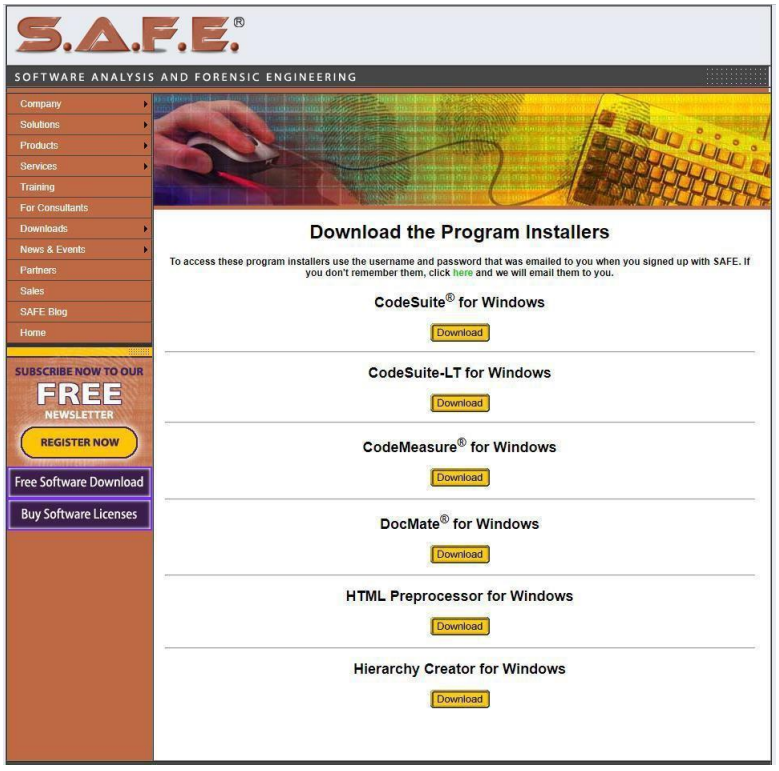
- 'n Drie maande, ses maande of jaarlikse lisensie kan aanlyn gekoop word om *CodeSuite*® onbeperk te kan gebruik.
- 'n Projek lisensie waar *CodeSuite*® bepaal hoeveel lisensies benodig word vir die spesifieke projek, waarna elke lisensie teen $10 aangekoop kan word.
- Vyftig proeflisensies is beskikbaar vir evaluering wat die gebruiker toelaat om 'n maksimum van 1Mb programkode te vergelyk. Met die aansoek vir hierdie lisensie vereis *CodeSuite*® verdere inligting vanaf die verskaffers oor die evaluering:
    - Die doel van die evaluering: besigheid, akademies, persoonlik of ander;
    - Watter tipe kodevergelyking getoets gaan word: binêre, bron of beide;
    - In watter *CodeSuite*® herkenner die gebruiker belangstel: *CodeCLOC*®, *CodeDiff*®, *CodeMatch*®, *CodeCross*®, *BitMatch*® of *SourceDetective*®; en
    - Na die suksesvolle evaluering, waarvoor word dit benodig: verkryging van inligting, een spesifieke projek of 'n ander doel.
- Aansoek vir 'n universiteitsprogramlisensie kan ingedien word, wat vir 'n maand 'n onbeperkte hernubare lisensie verleen vir 'n akademiese navorsingsprojek.

Om *CodeMatch*® PPHS te verkry vir hierdie navorsing is aansoek gedoen vir die *CodeSuite*® universiteitprogramlisensie. Eerstens moet die lisensie aansoekvorm verkry word deur aansoek te doen om *CodeSuite*® af te laai vanaf http://www.safe-corp.com/downloads_software.htm. Die gebruiker moet eers persoonlike inligting invul en indien waarna verdere instruksies aan die gebruiker per e-pos gestuur word. Die e-pos bevat 'n gebruikersnaam en wagwoord saam met 'n skakel om die sagteware af te laai, asook verdere inligting aangaande die verskillende produkte en lisensies.

Die aansoekvorm vir die universiteitsprogramlisensie word afgelaai vanaf die skakel in die e-pos. Die aansoekvorm/ooreenkomsvorm moet ingevul en geteken word deur 'n amptenaar van die universiteit soos 'n professor of administrateur. Hierdie aansoekvorm, saam met 'n beskrywing van die gebruik van die lisensie en die navorsingsprojek, moet per e-pos gestuur word aan sales@safe-corp.biz. Indien die aansoek goedgekeur is, ontvang die gebruiker 'n e-pos terug met die aansoekvorm/ooreenkomsvorm wat geteken is deur 'n SAFE (*software*

analysis and forensic engineering) verteenwoordiger. *CodeSuite*® moet dan geïnstalleer word om die unieke registrasiekode van die program te verkry wat saam met die aansoekvorm/ooreenkomsvorm gestuur word na authorize@SAFE-corp.com om die lisensiekode te ontvang.

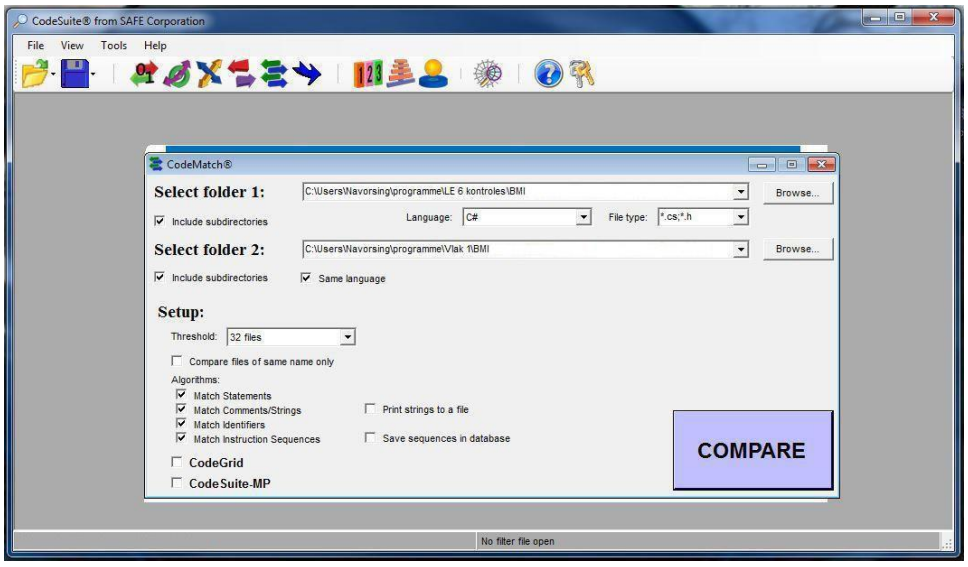
Om *CodeSuite*® af te laai en te installeer, moet die skakel in die e-pos gevolg word. Wanneer die skakel geklik word om die webblad te vertoon wat die gebruiker in staat stel om te selekteer watter produk hy/sy wil aflaai, vereis die webblad eers die gebruikersnaam en wagwoord, wat gegee is in die e-pos, voordat die webblad, soos in Figuur 1 gelaai word. Die *CodeSuite*® vir Windows word afgelaai as 'n installasielêer (.exe) met behulp van die grafiese gebruikerskoppelvlakinstallasie-assistent. Die stappe in die installasie-assistent moet net gevolg word om *CodeSuite*® suksesvol te installeer.



Figuur 1: Die webblad wat vertoon om die gekose *CodeSuite*® af te laai

2 Uitleg van *CodeMatch*®

CodeMatch® maak gebruik van 'n grafiese gebruikerskoppelvlak wat binne die koppelvlak van *CodeSuite*® hardloop, soos in Figuur 2. Die gebruikerskoppelvlak van *CodeMatch*® maak dit moontlik vir die gebruiker om die programmeringsopdragvouer op die rekenaar te gaan soek en daar kan ook gespesifiseer word of sub-afdelings van die vouer ingesluit moet word. Die programmeringstaal van die programmeringsopdrag kan volgens die gegewe lys gekies word, asook die lêertipe wat in ag geneem moet word.



Figuur 2 Die gebruikerskoppelvlak van *CodeMatch*® binne die *CodeSuite*® koppelvlak

Onder die opstellingafdeling kan die gebruiker die drumpelwaarde stel met die gegewe lys, selekteer of net lêers met dieselfde name gebruik moet word gedurende die vergelyking en selekteer watter algoritmes gebruik moet word. Saam met van die algoritmes is daar 'n addisionele opsie wat geselekteer kan word om die spesifieke bevindings wat saam met daardie algoritme gaan, te stoor. Vervolgens word die algoritmes wat deur *CodeMatch*® gebruik kan word, bespreek.

2.1 Identifiseerderpassingsalgoritme

Vir elke lêerpaar tel die algoritme die aantal ooreenstemmende identifiseerders wat nie programmeringstaalsleutelwoorde is nie. Om te bepaal of die identifiseerder 'n programmeringstaalsleutelwoord is, word die woorde vergelyk met 'n lys wat die sleutelwoorde van die programmeringstaal bevat. Slegs nie-sleutelwoorde word met mekaar vergelyk om sodoende ooreenstemmende funksiename, veranderlikes en ander identifiseerders te vind. Die algoritme bestudeer elke nie-sleutelwoordidentifiseerder in die een lêer en vind alle identifiseerders in die ander lêer waarvan die patroon ooreenstemmend is, soos byvoorbeeld as die identifiseerder abc in die een lêer gevind is en die identifiseerders abbbc, abc1111 of aaabc word in die ander lêer gevind, bestaan daar 'n gedeeltelike ooreenstemming met die identifiseerder abc. Gedeeltelike ooreenstemmings vind algemene funksiename, veranderlikes en ander identifiseerders wat iemand probeer het om op 'n redelike grondslag te vermom. Hierdie algoritme lewer 'n korrelasietelling wat die aantal ooreenstemmende en gedeeltelik ooreenstemmende identifiseerders in die twee lêers verteenwoordig (SAFE Corporation, 2015).

Hierdie algoritmes kan een vir een voor toetsing geselekteer word, dus kan hulle elkeen op hul eie of die geselekteerde pare kan saam uitgevoer word.

Die *CodeGrid*® en *Code Suite-MP* opsies heel onder aan die grafiese koppelvlak is opsioneel en is nie in hierdie eksperiment gebruik nie.

2.2 Stellingpassingsalgoritme

Hierdie algoritme vergelyk elke funksionele programkodelyn van albei lêers, uitsluitend die kommentaarlyne, met die verwydering van alle leë spasies. Programkodelyne word beskou as ooreenstemmend wanneer hulle minstens een nie-sleutelwoord bevat, soos 'n veranderlike naam of funksienaam en om te voorkom dat lyne wat basiese bewerkings bevat nie aangemeld word as ooreenstemmend nie. Hierdie algoritme lewer 'n korrelasietelling wat die aantal ooreenstemmende instruksielyne in die twee lêers verteenwoordig (SAFE Corporation, 2015).

2.3 Kommentaar-/Stringpassingsalgoritme

Hierdie algoritme vergelyk elke kommentaarlyn en stringlyn van albei lêers. Opeenvolgende leë spasies word omskep in enkelspasies sodat die sintaksisstruktuur van die lyn behou word. Alle kommentaar of stringe word vergelyk, ongeag of daar 'n sleutelwoord in die kommentaar of string is. Hierdie algoritme lewer 'n korrelasietelling wat die aantal ooreenstemmende kommentaarlyne en stringlyne in die twee lêers verteenwoordig (SAFE Corporation, 2015).

2.4 Instruksievolgordepassingsalgoritme

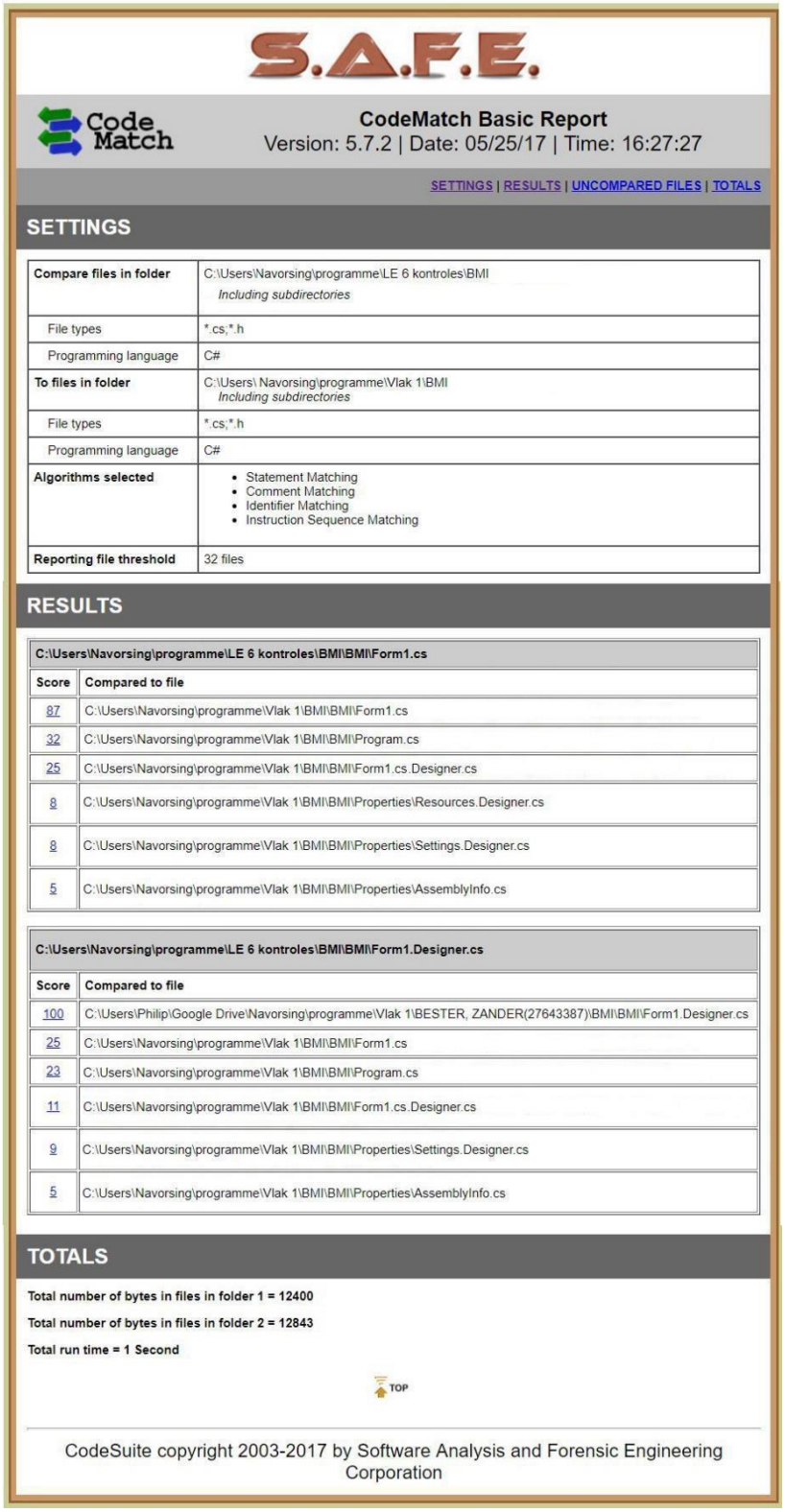
Hierdie algoritme vergelyk die eerste instruksie van elke programkodelyn van die gegewe lêers en ignoreer die kommentaar- en leë lyne. Hierdie algoritme vind reekse kode wat dieselfde funksie uitoefen ten spyte van die verandering van kommentaar en veranderlike name. Hierdie algoritme vind die langste algemene volgorde in beide lêers. Hierdie algoritme lewer 'n korrelasietelling wat die langste ooreenstemming in instruksievolgorde tussen die twee lêers verteenwoordig. Selfs al is die veranderlike name verander in die een lêer, sal *CodeMatch*® steeds die ooreenkomste tussen die lêers kan uitwys (SAFE Corporation, 2015).

3 Die afvoer/verslaggewing van *CodeMatch*®

Nadat die vergelyking uitgevoer is, word die resultate as 'n *CodeSuite*® databasis-lêer (.cbd) gestoor. Hierdie lêer moet weer deur middel van *CodeSuite*® oopgemaak word waarna dit as 'n HTML-verslag, CLOC-sigblad, verspreiding-sigblad, soektogsigblad, opsommende sigblad, PID-lys en XML gestoor kan word. Die HTML-afvoer is maklik verstaanbaar en

gebruikersvriendelik, dus is dit in hierdie eksperiment gebruik en word dit vervolgens hier bespreek.

CodeMatch® produseer ’n basiese HTML-verslag, soos in Figuur 3, wat die opstelling, resultate en totale weergee. Die totale korrelasietelling van elke C# lêer wat in die vouer voorkom, word weergegee van klein na groot in die resultate. Die gebruiker kan op die korrelasietellingskakel klik om na ’n gedetailleerde HTML-verslag te navigeer wat die presiese ooreenkomste tussen die twee lêers vertoon, kyk Figure 4 en 5. Elke algoritme wat gedurende die vergelyking se bevindings gebruik word, word apart weergegee onder die spesifieke algoritmenaam. Skakels in hierdie verslae maak dit maklik om in ’n verslag te navigeer en die vorentoe- en agtertoe-knoppie op die gedetailleerde verslag maak dit moontlik om na die volgende gedetailleerde verslag te navigeer. Die gedetailleerde verslag stel kenners in staat om self deur die ooreenkomste te gaan en laat dus toe dat die kenner sy eie oordeel kan fel.



S.A.F.E.

Code Match

CodeMatch Basic Report
Version: 5.7.2 | Date: 05/25/17 | Time: 16:27:27

SETTINGS | RESULTS | UNCOMPARED FILES | TOTALS

SETTINGS

Compare files in folder	C:\Users\Navorsing\programme\LE 6 kontroles\BMI *Including subdirectories*
File types	*.cs;*.h
Programming language	C#
To files in folder	C:\Users\ Navorsing\programme\Vlak 1\BMI *Including subdirectories*
File types	*.cs;*.h
Programming language	C#
Algorithms selected	• Statement Matching • Comment Matching • Identifier Matching • Instruction Sequence Matching
Reporting file threshold	32 files

RESULTS

C:\Users\Navorsing\programme\LE 6 kontroles\BMI\BMI\Form1.cs

Score	Compared to file
87	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Form1.cs
32	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Program.cs
25	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Form1.cs.Designer.cs
8	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Properties\Resources.Designer.cs
8	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Properties\Settings.Designer.cs
5	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Properties\AssemblyInfo.cs

C:\Users\Navorsing\programme\LE 6 kontroles\BMI\BMI\Form1.Designer.cs

Score	Compared to file
100	C:\Users\Philip\Google Drive\Navorsing\programme\Vlak 1\BESTER, ZANDER(27643387)\BMI\BMI\Form1.Designer.cs
25	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Form1.cs
23	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Program.cs
11	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Form1.cs.Designer.cs
9	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Properties\Settings.Designer.cs
5	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Properties\AssemblyInfo.cs

TOTALS

Total number of bytes in files in folder 1 = 12400

Total number of bytes in files in folder 2 = 12843

Total run time = 1 Second

TOP

Figuur 3 Voorbeeld van *CodeMatch*® se basiese HTML-verslag

S.A.F.E.

Code Match

CodeMatch Detailed Report
Version: 5.7.2 | Date: 05/25/17 | Time: 16:27:27

BACK NEXT **SCORE 87**

SETTINGS

Compare file 1:	C:\Users\Navorsing\programme\LE 6 kontroles\BMI\BMI\Form1.cs
To file 2:	C:\Users\Navorsing\programme\Vlak 1\BMI\BMI\Form1.cs
Links to results:	**Matching Statements** **Matching Comments and Strings** **Matching Instruction Sequences** **Matching Identifiers** **Partially Matching Identifiers** **Score**

RESULTS

Matching Statements

File1 Line#	File2 Line#	Statement
1	1	using System
2	2	using System.Collections.Generic
3	3	using System.ComponentModel
4	4	using System.Data
5	5	using System.Drawing
6	6	using System.Linq
7	7	using System.Text
8	8	using System.Threading.Tasks
9	9	using System.Windows.Forms
11	11	namespace BMI
13	13	public partial class Form1 : Form
15	15	string ster =
16	16	public Form1()
19	19	InitializeComponent()
23	23	private void button1_Click(object sender, EventArgs e)
25	25	double Height, Weight, BMI
26	26	String BMITest
28	28	KryGetalle(out Height, out Weight)
30	30	BMI = GetBMI(Height, Weight)
31	31	ster = Sterre(BMI)
33	33	if (BMI >30)
35 41 47 51	35 41 47 51	BMITest =
39	39	if (BMI > 25)
45	45	if (BMI > 18.5)
56	56	Display(BMITest)
60	60	private void KryGetalle(out double H, out double W)
62	62	H = Convert.ToDouble(textBox2.Text)
63	63	W = Convert.ToDouble(textBox1.Text)
66	66	private double GetBMI(double H,double W)
68	68	return W / (H * H)
71	71	private void Display(String BMI)
73	73	MessageBox.Show(+ BMI + + ster)

Figuur 4: Voorbeeld van *CodeMatch*® se gedetailleerde HTML-verslag

75	75	private void textBox2_TextChanged(object sender, EventArgs e)
80	80	private string Sterre(double bn)
82	82	string s =
83	83	int aantal = (int)bn/5
84	84	for (int i = 1
84	84	i <= aantal
84	84	i++)
85	85	s +=
86	86	return s

TOP

Matching Comments and Strings		
File1 Line#	**File2 Line#**	**Comment/String**
35	35	Obese
41	41	Overweight
47	47	Normal
51	51	Underweight
73	73	You Are

TOP

Matching Instruction Sequences		
File1 Line#	**File2 Line#**	**Number of matching instructions**
1	1	47

TOP

Matching Identifiers							
18	25	30	aantal	BMI	BMITest	bn	button1_Click
Collections	ComponentModel	Convert	Data	Display	Drawing	EventArgs	Form
Form1	Forms	Generic	GetBMI	Height	InitializeComponent	KryGetalle	Linq
MessageBox	sender	Show	ster	Sterre	String	System	Tasks
Text	textBox1	textBox2	textBox2_TextChanged	Threading	ToDouble	Weight	Windows

TOP

Partially Matching Identifiers
*** NONE ***

TOP

SCORE 87

Figuur 5: Voorbeeld van die gedetailleerde HTML-verslag (vervolg)

4 Bibliografie

SAFE Corporation. 2015. CodeMatch Algorithms. http://www.safe-corp.com/CodeMatch_algorithms.htm Datum van gebruik: 14 Jun. 2017.

BYLAAG E
TEGNIESE ASPEKTE VAN CPD

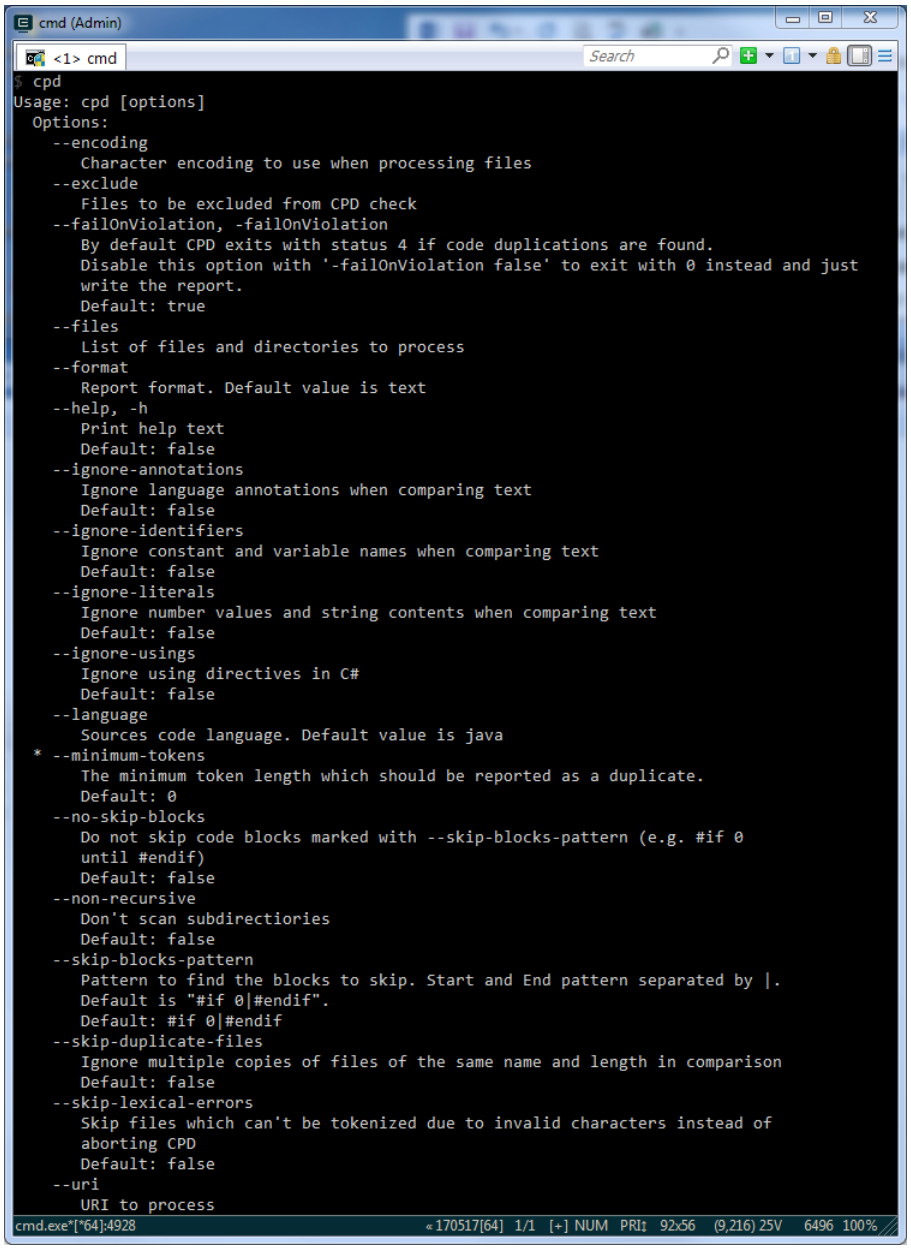
1 Verkryging en opstelling van CPD

Die nuutste weergawe van PMD kan gratis afgelaai word vanaf https://sourceforge.net/projects/pmd/files/ as 'n kompaklêer. Die voorvereistes om die program vir Windows- en Unix-gebruikers te kan hardloop, is dat die toestel Java bevat met 'n Java-ontwikkelingstel (JDK) van 1.4 of hoër. Om die kompaklêer te dekompakteer, benodig die gebruiker 'n program om dit uit te voer soos, byvoorbeeld Winzip of InfoZip. Die gebruiker kan die PMD-lêer dekompakteer na enige plek op die toestel se C-skyf of plaaslike skyf en binne die baliegedeelte van die uitgepakte PMD-lêer is die CPD-module.

2 Uitleg van CPD

Die CPD-module kan uitgevoer word deur gebruik te maak van bevelafvoering in die bevelaanporring of deur gebruik te maak van die grafiese gebruikerskoppelvlak.

Die CPD-module se uitvoeringinstruksies in die bevelaanporring gee verskeie opsies aan die gebruiker wanneer die cpd.bat-lêer gehardloop word, asook 'n voorbeeld van hoe so 'n instruksie aan CPD sal lyk. Die opsies en hulle verstekinstelling word in Figuur 1 weergegee. Omdat die grafiese gebruikerskoppelvlak gebruikersvriendelik is, word daar in die eksperiment meer daarop gefokus.



Figuur 1: CPD as bevelaanporring

Die grafiese gebruikerskoppelvlak van CPD, kyk Figuur 2, word verkry deur die cpdgui.bat-lêer uit te voer wat te vinde is in die balievouer.

Die *browser*-knoppie vir die *root source directory*opsie**,** stel die gebruiker in staat om die vouer te gaan selekteer wat die programmeringsopdragte bevat wat vir duplikate vergelyk moet word.

Langs die *report duplicate chunks larger than*-opsie kan die gebruiker spesifiseer hoe groot die duplikaatstukke (kentekens) moet wees waarna gesoek moet word en weergegee moet word in die afvoer.

Die *language*-opsie gee die gebruiker die opsie om tussen die agtien programmeringstale wat vroeër in Tabel 3-7 genoem is te kies. Die verskillende uitbreidings wat ondersteun word vir die gekose taal word onder die keuselys gegee. Die laaste opsie in die taalkeuselys stel die gebruiker in staat om self die uitbreiding te spesifiseer wat gebruik word.

Die *scan subdirectories*-opsie kan geselekteer word indien die program dieper moet delf in die gespesifiseerde vouer.

Die *ignore literals*-opsie kan geselekteer word om getalswaardes en stringinhoude te ignoreer wanneer teks vergelyk word.

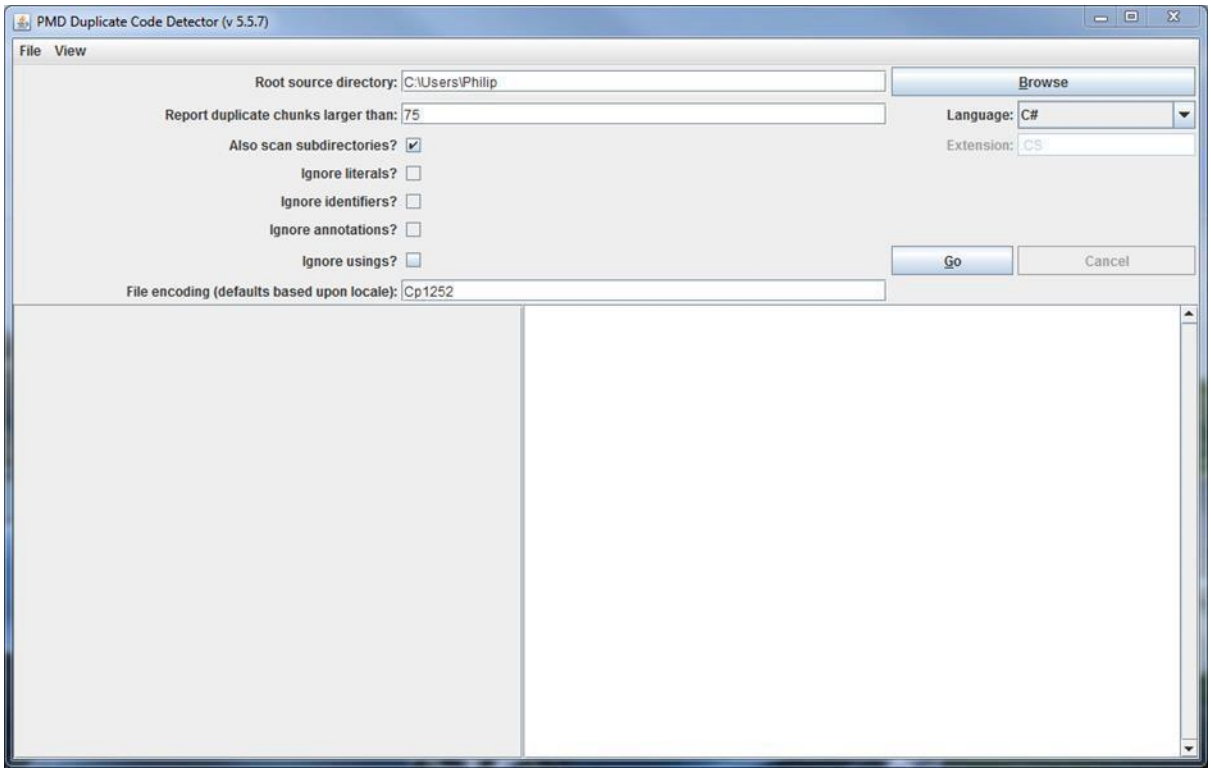
Die *ignore identifiers*-opsie kan gekies word om konstante en veranderlike name te ignoreer wanneer die teks vergelyk word.

Die *ignore annotations*-opsie kan geselekteer word om die kommentaar uit te sluit wanneer die vergelyking plaasvind.

Die *ignore usings*-opsie stel die gebruiker in staat om te selekteer of die gebruiksvoorskrifte van die programmeringstaal C# ingesluit moet word in die teks-vergelyking.

Die *file encoding*-opsie laat die gebruiker toe om te spesifiseer watter karakterenkodering gebruik moet word wanneer die lêer verwerk word vir die vergelyking.

Met die gebruik van die grafiese gebruikerskoppelvlak kan die verskeie opsies vir die gespesifiseerde programmeringstaal geaktiveer en gedeaktiveer word, dus varieer die opsies beskikbaar by elke programmeringstaal.

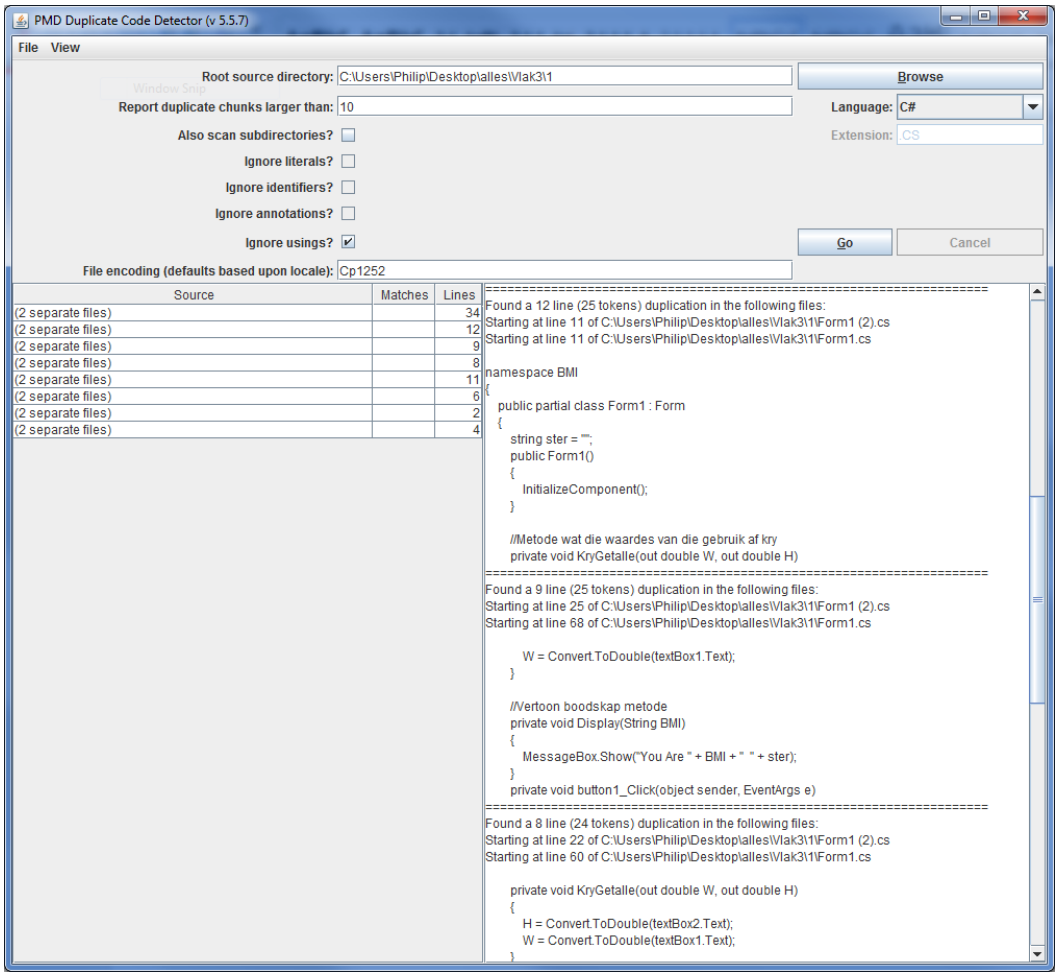


Figuur 2: Die grafiese gebruikerskoppelvlak van CPD se 5.5.7-weergawe

3 Die afvoer/verslaggewing van CPD

Wanneer die *Go*-knoppie geklik word, verwerk CPD die gegewe opsies en vertoon die afvoer onder aan die skerm in twee afdelings, soos in Figuur 3. Aan die regterkant word al die gedupliseerde inhoud en die aantal merktekens wat verkry is, vertoon. Aan die linkerkant word die aantal lêers wat die duplikate bevat en die aantal lyne wat as duplikate gerapporteer is, vertoon. Die gebruiker kan aan die linkerkant 'n spesifieke lêer se ooreenkomste selekteer, waarna die gedupliseerde inhoud aan die regterkant vertoon word.

Die gebruiker kan die afvoer stoor in teks-, XLM- of as CSV-formaat maar, slegs die lyn-nommers en programmeringskodeooreenkomste word in hierdie lêers gestoor soos wat dit weergegee word in grafiese gebruikerskoppelvlak regs onder.



Figuur 3: 'n Voorbeeld van CPD se afvoer

BYLAAG F
TEGNIESE ASPEKTE VAN JPLAG

1 Verkryging en opstelling van JPlag

Volgens JPlag (2015) aanvaar hulle nie meer nuwe registrasies vir die webdienste nie en moet nuwe gebruikers JPlag aflaai vanaf https://github.com/jplag/jplag/releases. Die JPlag[weergawe].jar Java lêer van om en by 1.5MB moet afgelaai word, waarna dit verkieslik geskuif word na die toestel se plaaslike skyf of C-skyf. Om JPlag te hardloop moet die Java-lêer uitgevoer word in die konsole.

2 Uitleg van JPlag

Die gebruiker kan '`java -jar jplag-jouWeergawe.jar`' in die konsole tik en uitvoer om die instruksie-opsies van JPlag te sien, soos in Figuur 1.

```
Philip@PHILIP-PC C:\
$ java -jar jplag-2.11.9-SNAPSHOT-jar-with-dependencies.jar
JPlag (Version 2.11.9-SNAPSHOT), Copyright (c) 2004-2015 KIT - IPD Tichy, Guido Malpohl, and others.
Usage: JPlag [ options ] <root-dir>
 <root-dir>        The root-directory that contains all submissions

options are:
 -v[qlpd]        (Verbose)
                 q: (Quiet) no output
                 l: (Long) detailed output
                 p: print all (p)arser messages
                 d: print (d)etails about each submission
 -d              (Debug) parser. Non-parsable files will be stored.
 -S <dir>        Look in directories <root-dir>/*/<dir> for programs.
                 (default: <root-dir>/*)
 -s              (Subdirs) Look at files in subdirs too (default: deactivated)

 -p <suffixes>   <suffixes> is a comma-separated list of all filename suffixes
                 that are included. ("-p ?" for defaults)

 -o <file>       (Output) The Parserlog will be saved to <file>
 -x <file>       (eXclude) All files named in <file> will be ignored
 -t <n>          (Token) Tune the sensitivity of the comparison. A smaller
                 <n> increases the sensitivity.
 -m <n>          (Matches) Number of matches that will be saved (default:20)
 -m <p>%         All matches with more than <p>% similarity will be saved.
 -r <dir>        (Result) Name of directory in which the web pages will be
                 stored (default: result)
 -bc <dir>       Name of the directory which contains the basecode (common framework)
 -l <language>   (Language) Supported Languages:
                 java17 (default), java15, java15dm, java12, java11, python3, c/c++, c#-1.2, char, text, scheme
```

Figuur 1: Die verskillende instruksies van JPlag

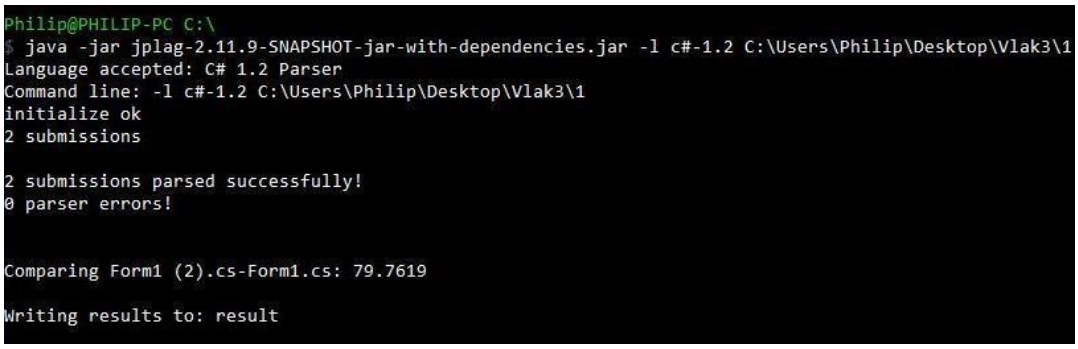
Om 'n vergelyking van lêers in JPlag uit te voer, moet al die programmeringsopdragte wat met mekaar vergelyk gaan word voorkom in een vouer of as 'n subvouer in daardie vouer. Indien al die programmeringsopdragte voorkom in een vouer genaamd 'Opdragte', kan die instruksie in die konsole soos volg wees:

```
java –jar jplag-2.11.9-SNAPSHOT-jar-with-dependencies.jar –l c#-1.2 –r
C:\Users\Desktop\Resultaat  C:\Users\Desktop\Opdragte
```

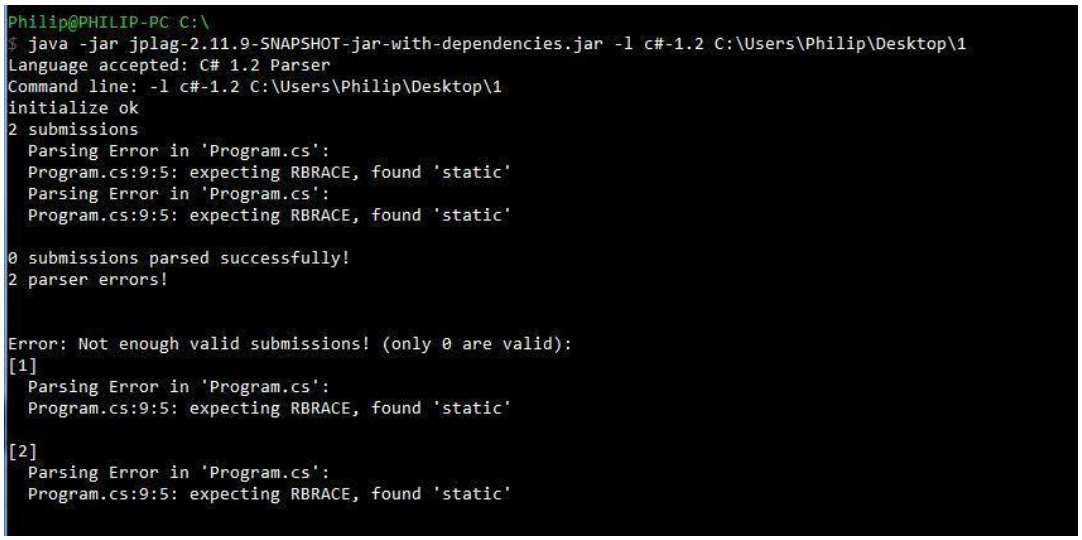
waar die Java-lêer van JPlag eers gespesifiseer moet word en daarna volg die instruksies van Figuur 1 soos benodig. In hierdie studie word die programmeringstaal verander vanaf Java17 na C# deur die '`-l c#-1.2`'-opsie by te voeg. Die spelling van die programmeringstaal moet voorkom soos in die instruksie-opsies van Figuur 1. Die '`-r C:\Users\Desktop\Resultaat`'- opsie spesifiseer dat die resultate gestoor moet word in die resultaatvouer. Die laaste gedeelte van die instruksie spesifiseer die gids waarin die programmeringsopdragte voorkom.

3 Die afvoer/verslaggewing van JPlag

Wanneer die ingevoerde instruksie uitgevoer word, vertoon die konsole of die programmeringsopdragte suksesvol deur JPlag gepers is en of enige foute voorgekom het. Indien die programmeringsopdragte suksesvol gepers is, vertoon die verskillende opdragte wat met mekaar vergelyk is saam met die ooreenkomspersentasie tussen die opdragte, soos in Figuur 2. Indien 'n fout voorkom, word dit vir die gebruiker uitgewys, soos byvoorbeeld in Figuur 3.



Figuur 2: Voorbeeld van 'n suksesvolle afvoering van JPlag in die konsole



Figuur 3: ’n Voorbeeld van ’n foutiewe afvoering van JPlag

Soos bespreek in 2, spesifiseer die gebruiker in watter vouer die resultate gestoor moet word. Die resultate word in HTML-formaat gestoor. Indien ’n vouer nie gespesifiseer word nie, soos in 2, word ’n nuwe vouer-resultaat outomaties geskep waar JPlag se Java-lêer geleë is. Die resultaatvouer word oorgeskryf, dus word die vorige resultate uitgevee deur die nuwe resultate wat verkry is.

In die resultaatvouer kom verskeie HTML-lêers voor. Die HTLM-lêer genaamd *Index* bevat ’n opsommende verslag van die vergelyking(s) wat plaasgevind het. Soos die naam van die lêer beskryf, dien dit as ’n inhoudsopgawe tot die resultate wat gevind is. Die eerste gedeelte bestaan uit algemene inligting oor die vergelyking(s), soos die name van die verskillende opdragte wat ingedien is vir vergelyking, die programmeringstaal wat geselekteer is, die aantal opdragte wat ingedien is en die datum wanneer die vergelyking plaasgevind het, ens. Onder die algemene inligting word die verspreiding van die resultate grafies voorgestel. Die laaste gedeelte van die verslag sorteer die vergelykings volgens die gemiddelde en maksimum ooreenkomste tussen die verskillende opdragte. ’n Voorbeeld van hierdie verslag word in Figuur 4 weergegee.

Search Results

Title:	
Programs:	Form1 V1.cs - Form1 V2.cs - Form1 V3.cs - Form1 V4.cs - Form1 V5.cs - Form1 V6.cs - Form1.cs
Language:	C# 1.2 Parser
Submissions:	7
Matches displayed:	21 (Treshold: 62.6%) (average similarity) 21 (Treshold: 65.2%) (maximum similarity)
Date:	2017-05-21
Minimum Match Length (sensitivity):	8
Suffixes:	.cs, .CS

Distribution:

90% - 100%	10	##
80% - 90%	0	.
70% - 80%	9	##
60% - 70%	2	###############
50% - 60%	0	.
40% - 50%	0	.
30% - 40%	0	.
20% - 30%	0	.
10% - 20%	0	.
0% - 10%	0	.

Matches sorted by average similarity (What is this?):

Form1.cs	->	Form1 V1.cs (100.0%)	Form1 V2.cs (100.0%)	Form1 V6.cs (98.0%)	Form1 V5.cs (93.5%)	Form1 V4.cs (73.8%)	Form1 V3.cs (73.5%)
Form1 V2.cs	->	Form1 V1.cs (100.0%)	Form1 V6.cs (98.0%)	Form1 V5.cs (93.5%)	Form1 V4.cs (73.8%)	Form1 V3.cs (73.5%)	
Form1 V1.cs	->	Form1 V6.cs (98.0%)	Form1 V5.cs (93.5%)	Form1 V4.cs (73.8%)	Form1 V3.cs (73.5%)		
Form1 V5.cs	->	Form1 V6.cs (91.5%)	Form1 V4.cs (72.8%)	Form1 V3.cs (72.6%)			
Form1 V6.cs	->	Form1 V4.cs (72.6%)	Form1 V3.cs (68.4%)				
Form1 V4.cs	->	Form1 V3.cs (62.6%)					

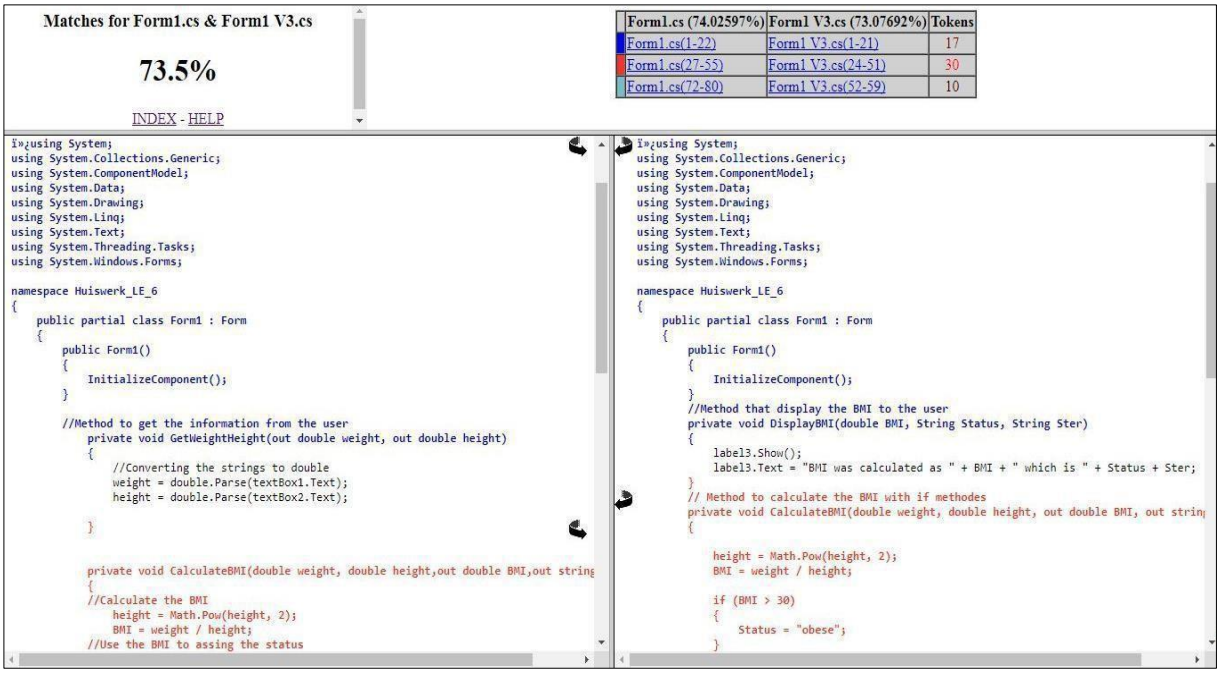
Matches sorted by maximum similarity (What is this?):

Form1.cs	->	Form1 V1.cs (100.0%)	Form1 V2.cs (100.0%)	Form1 V6.cs (100.0%)	Form1 V5.cs (94.8%)	Form1 V4.cs (76.3%)	Form1 V3.cs (74.0%)
Form1 V2.cs	->	Form1 V6.cs (100.0%)	Form1 V1.cs (100.0%)	Form1 V5.cs (94.8%)	Form1 V4.cs (76.3%)	Form1 V3.cs (74.0%)	
Form1 V1.cs	->	Form1 V6.cs (100.0%)	Form1 V5.cs (94.8%)	Form1 V4.cs (76.3%)	Form1 V3.cs (74.0%)		
Form1 V5.cs	->	Form1 V6.cs (94.5%)	Form1 V4.cs (76.3%)	Form1 V3.cs (73.0%)			
Form1 V6.cs	->	Form1 V4.cs (73.6%)	Form1 V3.cs (70.2%)				
Form1 V4.cs	->	Form1 V3.cs (65.2%)					

Figuur 4: ’n Voorbeeld van die opsommingsverslag van die indekslêer van JPlag

Die lêername van die programmeringsopdragte dien as skakels wat die gebruiker toelaat om die ooreenkomste tussen twee opdragte te aanskou. As die gebruiker op een van hierdie skakels klik, word ’n nuwe verslag in die webblaaier oopgemaak. Hierdie verslag word in vier verdeel, soos in Figuur 5. Die boonste raam links bevat die ooreenkomspersentasie van die

twee opdragte, asook skakels om na die indeks of hulpblad te navigeer. Die raam aan die regterkant bo bevat ’n tabel van al die gedeeltes wat tussen die twee opdragte ooreenstem. Elke ry van die tabel gee ’n reeks lynnommers van die twee lêers wat as dieselfde beskou word, saam met die grootte daarvan in merktekens. Die lêername in die tabel is skakels wat die ooreenstemmende gedeeltes van die opdragte in die onderste twee groot rame vertoon. Die twee rame bevat die inhoud van die opdragte en merk die ooreenstemmende gedeeltes met dieselfde tekskleur. Swart teks dui aan dat daar geen ooreenkomste in een van die twee opdragte voorgekom het nie. Elke ooreenstemmende gedeelte word geassosieer met ’n pyltjie wat geklik kan word om die geassosieerde ooreenstemmende gedeelte van die ander opdrag direk langs hierdie een te vertoon.



Figuur 5: ’n Voorbeeld van ’n ooreenkomsteverslag tussen twee opdragte van Jplag

4 Bibliografie

JPlag. 2015. Use JPlag. http://jplag.ipd.kit.edu/ Datum van gebruik: 19 Jun. 2017.

BYLAAG G
TEGNIESE ASPEKTE VAN MOSS

1 Verkryging en opstelling van MOSS

Kommunikasie met MOSS bestaan uit 'n skrip wat oorspronklik in Perl geskryf is en voer op 'n Linux-sisteem uit en kan ook op Windows uitgevoer word indien daar gebruik gemaak word van die Cygwin-program. Omdat die MOSS *skrip* vrylik beskikbaar is, het die gemeenskap dit deur die jare in verskeie ander programmeringstale gaan herskryf. Daar bestaan dus die volgende weergawes van die indiening van die skrip tot MOSS:

a. 'n algemene Lisp-indieningseenheid;
b. 'n Java-weergawe;
c. 'n PHP-weergawe; en
d. 'n grafiese gebruikerskoppelvlak vir Windows-weergawe.

Hierdie verkillende weergawes kan verkry word vanaf MOSS se webblad: http://theory.stanford.edu/~aiken/moss/. Elke weergawe bevat 'n tekslêer wat 'n beskrywing gee van die program, die nodige voorvereistes om die program te hardloop en voorbeelde van hoe die toevoer moet lyk.

Omdat hierdie verhandeling fokus op grafiese gebruikerskoppelvlak, gaan MOSS se GGK vir Windows-weergawe gebruik en verduidelik word. Die weergawe is in C# geskryf deur gebruik te maak van Visual studio vir die ontwerp van die grafiese koppelvlak. Die grafiese gebruikerskoppelvlakweergawe word afgelaai as 'n gekompakteerde Visual studio-projeklêer genaamd MossApp.

Die Visual studio-projeklêer (MossApp) bestaan uit programkode en die MossApp lêer word uitgepak voor dit op een van die twee maniere uitgevoer word:

1. Die gebruiker installeer die Visual studio-program op die Windows-toestel en maak die projeklêer deur middel van Visual studio oop. Die program stel die gebruiker in staat om veranderings in die programkode, sowel as aan die grafiese koppelvlak aan te bring. Elke keer wat MossApp deur Visual studio gehardloop word, gaan die programkode deur toetsing en validering voordat die grafiese gebruikerskoppelvlak vir die gebruiker se toevoere vertoon.

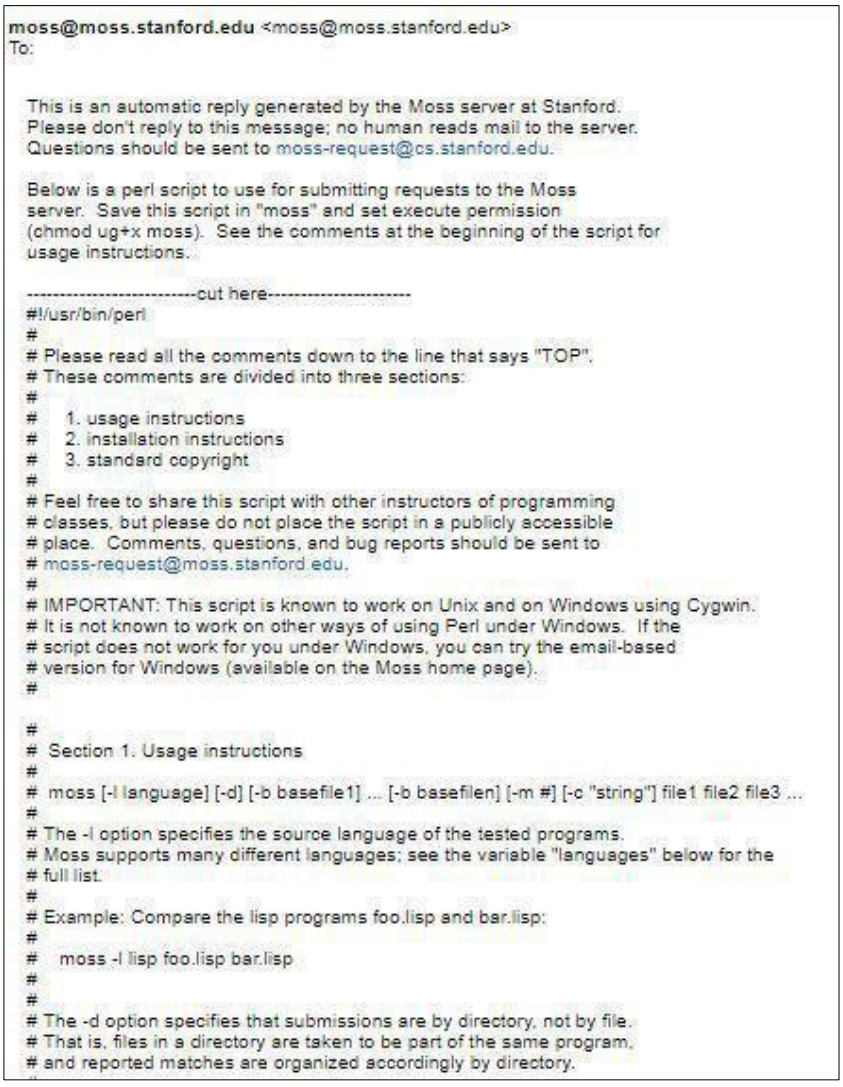
2. Die gebruiker navigeer na die sublêer genaamd Debug deur die MossApp\MossApp\bin\Debug roete te volg en dubbelklik op die toepassinglêer van MossApp, wat dan die grafiese gebruikerskoppelvlak aan die gebruiker vertoon.

Voordat die gebruiker programmeringsopdragte vir vergelyking aan die MOSS-bediener kan stuur, benodig die gebruiker 'n gebruikers-id. MOSS het dit moontlik gemaak vir enige iemand om gratis vir 'n nie-kommersiële gebruikers-id te registreer. Indien die doel is om MOSS vir kommersiële doeleindes te gebruik moet die gebruiker Similix kontak vir 'n lisensie. Om 'n gebruikers-id te verkry, moet die volgende boodskap presies net so aan moss@moss.stanford.edu gestuur word:

registeruser
mail *gebruikers@eposadres*

waar die skuinsgedrukte gedeelte vervang moet word met die gebruiker se e-posadres.

Indien die e-pos korrek gestuur is, ontvang die gebruiker 'n outomatiese geskepte e-pos vanaf MOSS se bediener van Stanford. In die e-pos word die perl skrip en verduideliking gegee om versoeke in te dien tot die MOSS-bediener. 'n Voorbeeld van die eerste gedeelte van hierdie e-pos word in Figuur 1 weergegee. Die gebruiker moet deur hierdie skrip gaan en die gebruikers-id soek wat voorkom, soos in Figuur 2.



moss@moss.stanford.edu <moss@moss.stanford.edu>
To:

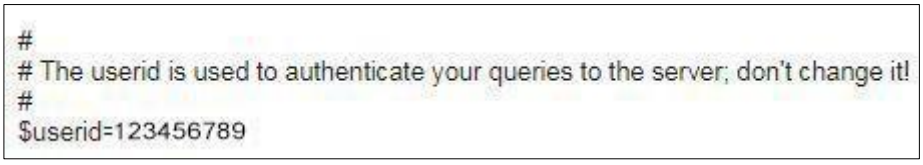
This is an automatic reply generated by the Moss server at Stanford.
Please don't reply to this message; no human reads mail to the server.
Questions should be sent to moss-request@cs.stanford.edu.

Below is a perl script to use for submitting requests to the Moss
server. Save this script in "moss" and set execute permission
(chmod ug+x moss). See the comments at the beginning of the script for
usage instructions.

----------cut here----------
#!/usr/bin/perl
#
Please read all the comments down to the line that says "TOP".
These comments are divided into three sections:
#
1. usage instructions
2. installation instructions
3. standard copyright
#
Feel free to share this script with other instructors of programming
classes, but please do not place the script in a publicly accessible
place. Comments, questions, and bug reports should be sent to
moss-request@moss.stanford.edu.
#
IMPORTANT: This script is known to work on Unix and on Windows using Cygwin.
It is not known to work on other ways of using Perl under Windows. If the
script does not work for you under Windows, you can try the email-based
version for Windows (available on the Moss home page).
#

#
Section 1. Usage instructions
#
moss [-l language] [-d] [-b basefile1] ... [-b basefilen] [-m #] [-c "string"] file1 file2 file3 ...
#
The -l option specifies the source language of the tested programs.
Moss supports many different languages; see the variable "languages" below for the
full list.
#
Example: Compare the lisp programs foo.lisp and bar.lisp:
#
moss -l lisp foo.lisp bar.lisp
#
#
The -d option specifies that submissions are by directory, not by file.
That is, files in a directory are taken to be part of the same program,
and reported matches are organized accordingly by directory.

Figuur 1: Voorbeeld van die e-pos wat MOSS outomaties terugstuur na aansoek om 'n gebruikers-id

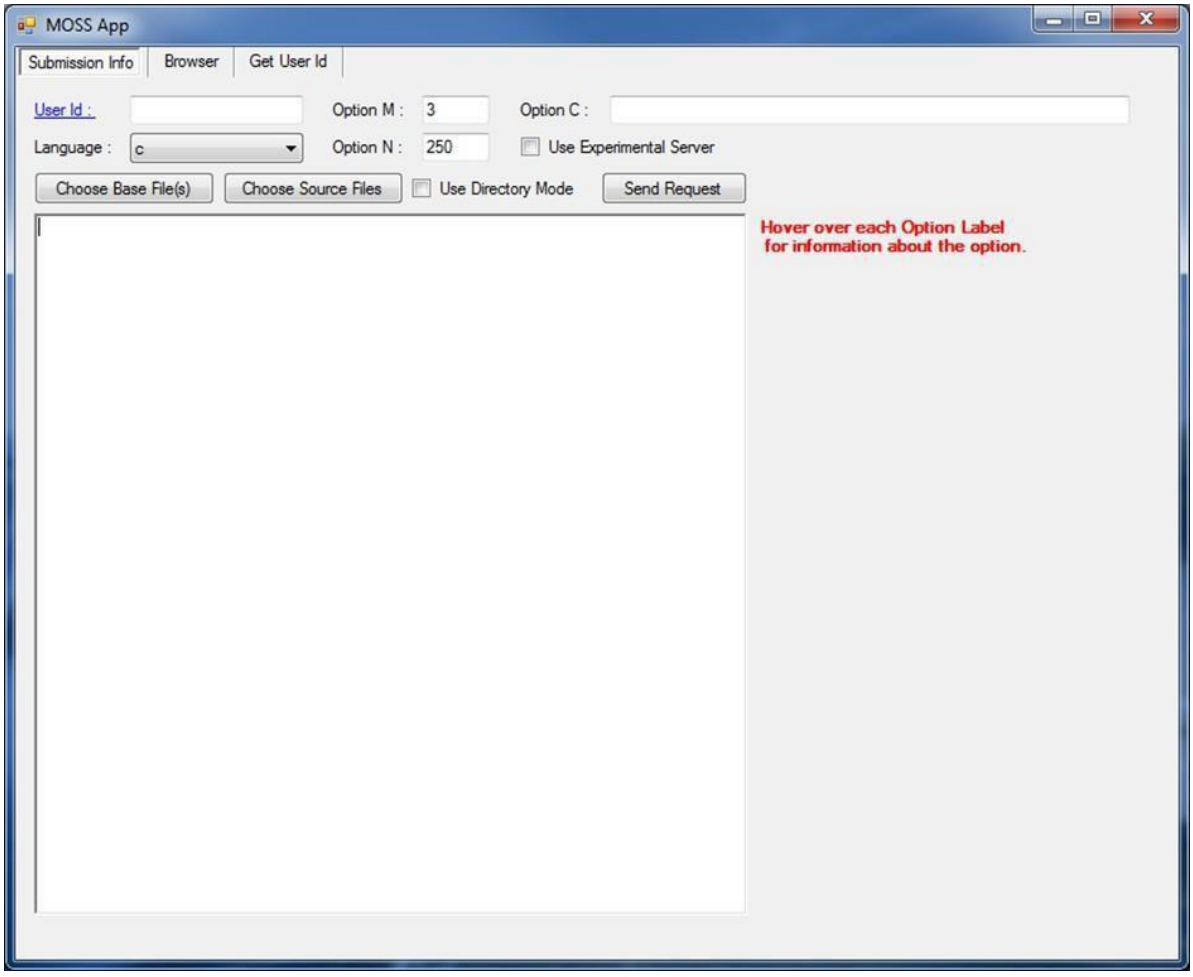


#
The userid is used to authenticate your queries to the server; don't change it!
#
$userid=123456789

Figuur 2: Die wyse waarop die gebruikers-id in die e-pos vanaf MOSS voorkom

2 Uitleg van MOSS

Wanneer MossApp uitgevoer word, vertoon die grafiese gebruikerskoppelvlak soos in Figuur 3.



Figuur 3: MOSS se grafiese gebruikerskoppelvlak

Die grafiese gebruikerskoppelvlak is verdeel in drie dele: *submission info, browser* en *get user id*. Daar kan tussen hierdie drie dele genavigeer word deur op die gepaste naam te klik heel bo-aan die grafiese gebruikerskoppelvlak. Die *get user id*-blad vertoon die inligting soos bepreek in Afdeling 1. Die gebruiker kan sy/haar e-posadres in ’n teksboks toevoer waarna die e-posteks gegenereer word wat gekopieer kan word en aan MOSS gestuur word vir die verkryging van die gebruikers-id. Die *browser*-blad vertoon die afvoer wat verkry word vanaf die MOSS-bediener en word verder in Afdeling 3 bespreek. Die *submission info*-blad word gebruik om die nodige inligting met verskeie opsies aan MOSS se bediener te stuur. Soos daar in rooi geskryf staan, moet die gebruiker die merker oor elke opsie se etiket hou om verdere inligting te verkry oor die opsie.

By die *get user id* moet die gebruiker die id wat in die e-pos vanaf MOSS verkry is, insleutel voordat enige afvoering kan plaasvind.

Die *language* kan deur middel van die gegewe keuselys geselekteer word.

Die *option M* stel die maksimum aantal kere wat 'n gedeelte kan voorkom voordat dit geïgnoreer word. Soms kom gedeeltes in programkode voor wat gedeel mag word en nie plagiaat is nie. As M-opsie gestel is op 2 dan rapporteer MOSS slegs gedeeltes wat voorkom in presies twee programme. Indien die gebruiker verwag dat die ooreenkomste tussen programme baie gaan wees, is die ideale stelling van M-opsie tussen 3 tot 4. Die M-opsie is nuttig vir groot programmeringsopdragte. Die verstekinstelling vir M-opsie is 3.

Die *option N* laat die gebruiker toe om die aantal ooreenkomslêers wat vertoon moet word in die resultate te spesifiseer. Die verstekwaarde is 250.

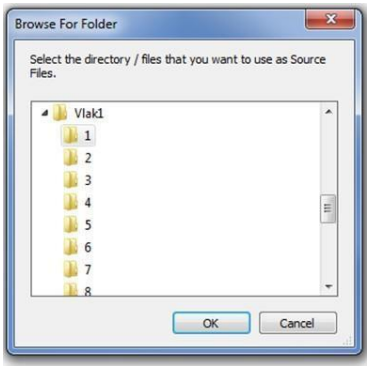
Die *option C* verskaf 'n kommentaarstring wat aan die gegenereerde verslag geheg word. Hierdie opsie fasiliteer 'n ooreenstemmende navraag met die resultate wat ontvang is, veral wanneer verskeie navrae ingedien word.

Die *use experimental server*-opsie kan gekies word indien die gebruiker die huidige eksperimentele weergawe van die bediener wil gebruik. Hierdie bediener bevat die nuutste funksies, maar is gewoonlik minder stabiel.

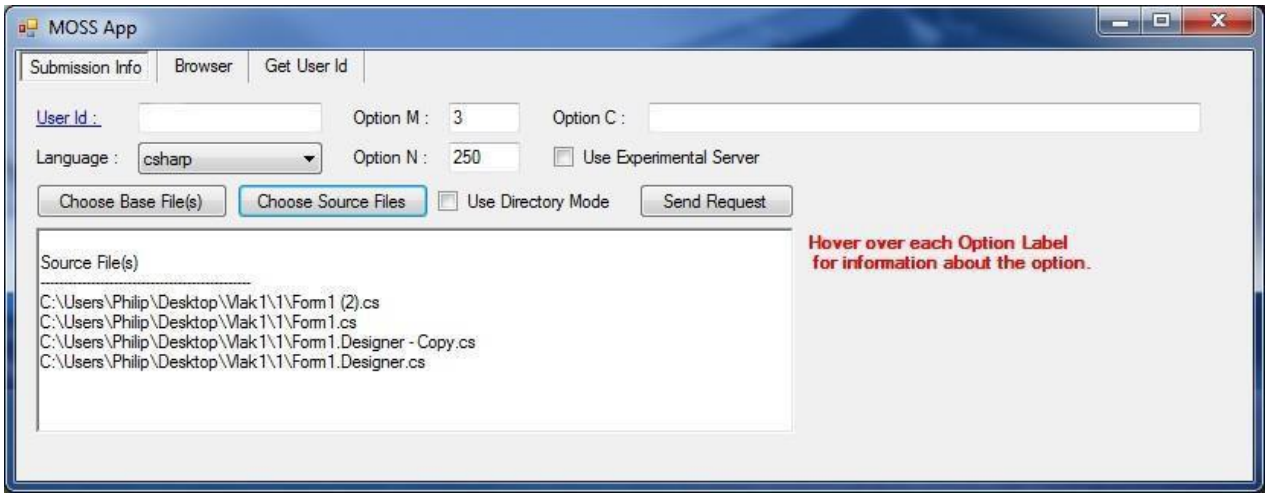
Die *choose base file(s)*-knoppie stel die gebruiker in staat om 'n basislêer by te voeg. Hierdie basislêer is instruksies of voorbeeldprogrammering wat die dosent/instrukteur verskaf vir die programmeringsopdrag, wat nie ingesluit moet word wanneer daar getoets word vir plagiaat nie. Die programmeringskode wat in die basislêer voorkom, word uitgesluit uit die vergelyking tussen die programmeringsopdragte.

Die *choose source files*-knoppie laat die gebruiker toe om die lêers te gaan selekteer wat vir plagiaat deur MOSS vergelyk moet word. Wanneer die knoppie geklik word, vertoon 'n nuwe venster, soos Figuur 4, wat die gebruiker in staat stel om die programmeringslêers te soek en te selekteer. Die geselekteerde programmeringslêers se gids vertoon onder in die teksboks, soos in Figuur 5. Wanneer daar weer op die kies bronlêers-knoppie geklik word, vee dit die lêers uit, dus is dit makliker om al die programmeringsopdragte wat met mekaar vergelyk moet word in een lêer te stoor.

Die *use directory mode*-opsie word geselekteer indien die programmeringsopdragte sublêers onder die hooflêer bevat. Die metode beskou dan elke lêer en sublêer as dieselfde.

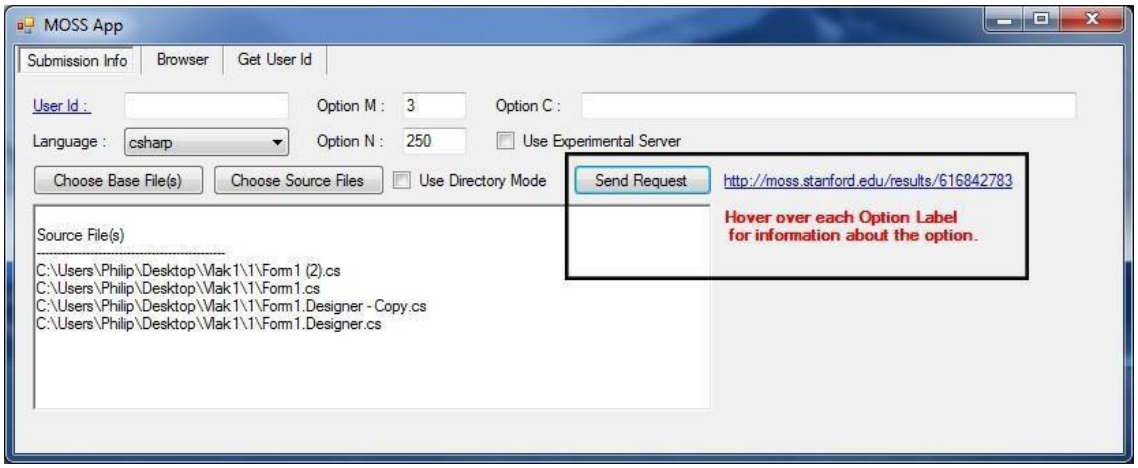


Figuur 4: Die venster wat in MOSS vertoon om die lêers te selekteer



Figuur 5: Die opdatering van die teksboks met die geselekteerde lêers wat aan MOSS gestuur gaan word

Wanneer die gebruiker al die verskeie opsies gestel het, en die programmeringsopdragte is gelaai, kan die gebruiker op die *send request*-knoppie klik om dit na MOSS te stuur. Die gebruikerskoppelvlak stel die gebruiker nie in kennis indien die vergelyking tussen die opdragte suksesvol voltooi is nie. Die gebruiker moet met die eerste indien wag totdat daar 'n skakel vertoon net regs van die knoppie (kyk Figuur 6). Na die eerste indien verander slegs die nommer agteraan die skakel en moet die gebruiker hierdie nommer dophou om te weet of die vergelyking suksesvol was. Hierdie skakel neem die gebruiker na 'n webblaaier waar die resultate wat verkry is, vertoon word.

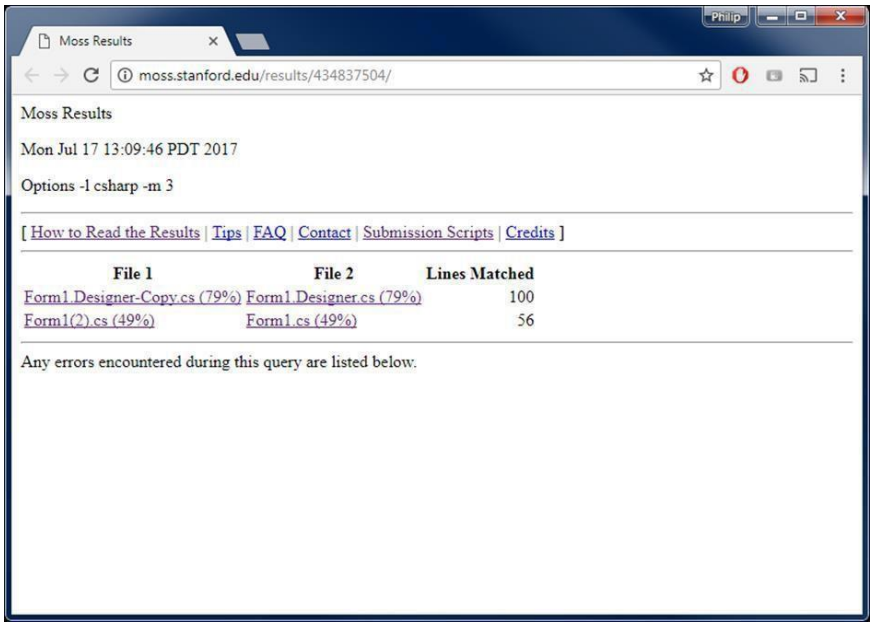


Figuur 6: 'n Voorbeeld van die vertoning van die skakel wanneer vergelyking van opdragte suksesvol deur MOSS uitgevoer is

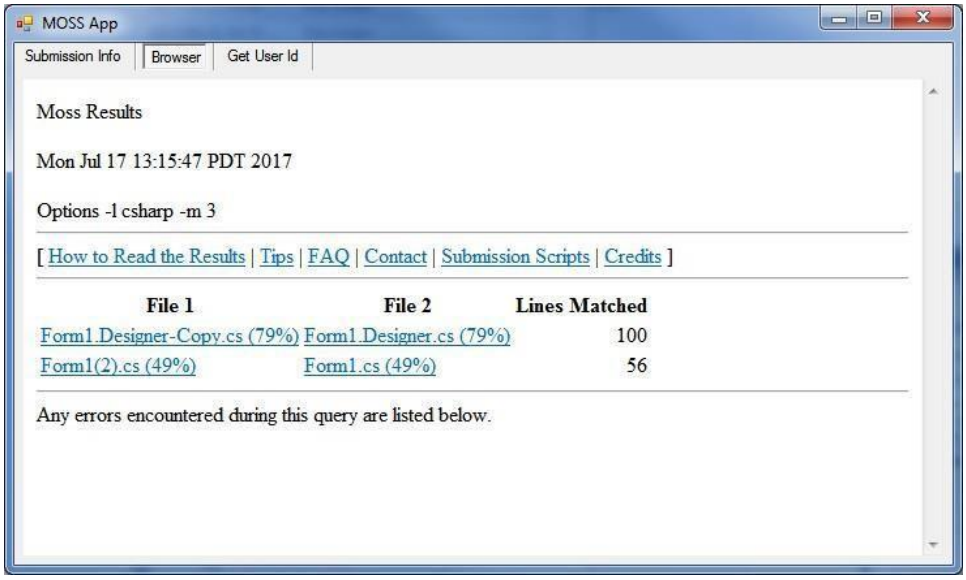
3 Die afvoer/verslaggewing van MOSS

MOSS produseer terugvoer vanaf die bediener aan die gebruiker in HTLM-formaat. Hierdie terugvoer kan op twee maniere bestudeer word. Die gebruiker ontvang 'n skakel nadat die versoek ingedien en voltooi is, soos bespreek in Afdeling 2. Die gebruiker kan op hierdie skakel klik, waarna die webblaaier oopmaak soos in Figuur 7. Anders kan die gebruiker op die *browser*-knoppie bo aan die grafiese gebruikerskoppelvlak klik wat dan die skerm soos in Figuur 8 sal laai.

Die indeksblad, wat bestaan uit 'n tabel met die lêers wat ooreenstemmende programmeringskode bevat, word gelaai. Elke lêernaam bevat 'n persentasietelling wat die persentasie weergee van die ooreenstemming van die een lêer se kode met dié van die ander lêer. Die lynooreenstemming is die aantal lyne kode wat min of meer tussen die twee lêers met mekaar ooreenstem. In albei van hierdie gevalle beteken 'n hoër getal dat ooreenkomste meer is.

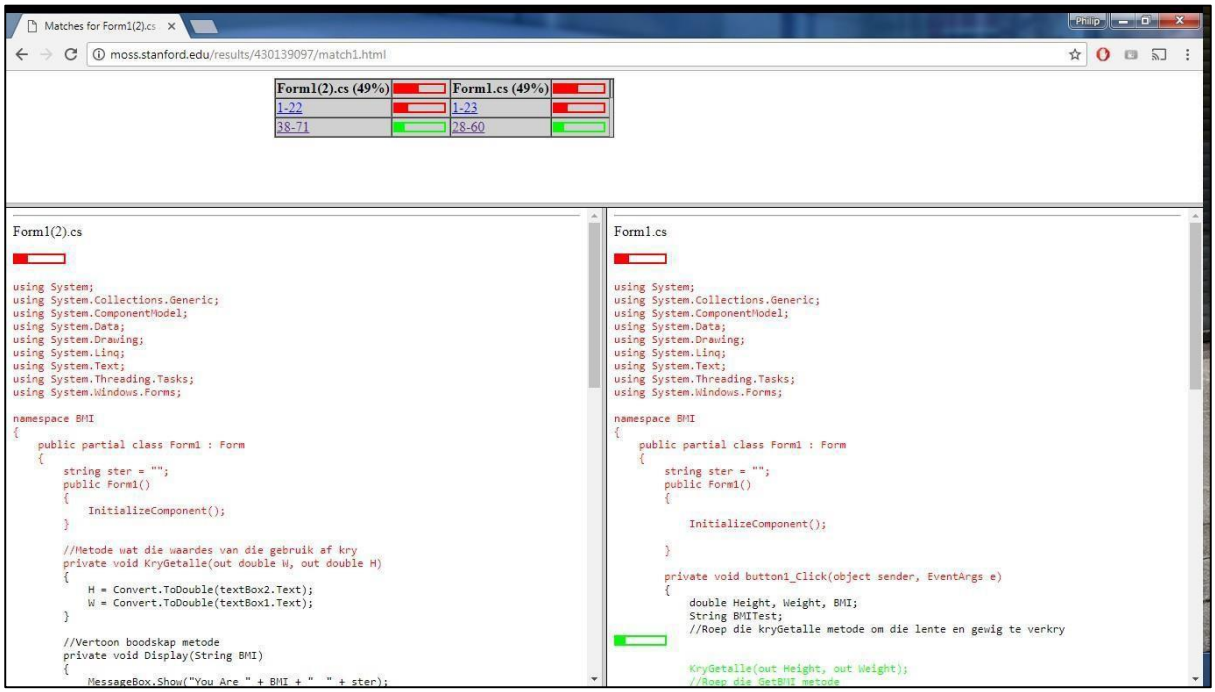


Figuur 7: 'n Voorbeeld van die HTML-afvoer van die resultate in die webblaaier deur MOSS



Figuur 8: 'n Voorbeeld van die HTML-afvoer van die resultate in die grafiese gebruikerskoppelvlak

Die lêername dien as skakels wat geklik kan word om die spesifieke gedeeltes wat ooreenstem tussen die twee lêers te aanskou. Die skakel laai 'n nuwe blad, soos in Figuur 9, wat verdeel is in drie rame: die boonste een vir 'n tabel van al die gedeeltes wat ooreenstemmend is en die onderste twee vir die vertoning van die programkode van die twee lêers.



Figuur 9: 'n Voorbeeld van die drie rame waarin die vergelykingsafvoer van MOSS voorkom

Die tabel wat voorkom, het die volgende formaat:

Leêr1 (49%)		Leêr2 (49%)	
1-22		1-23	
38-71		28-60	

waar elke ry 'n reeks lynnommers gee van lêer 1 en lêer 2 wat beskou word as dieselfde, saam met 'n grafiese termometer wat 'n aanduiding gee van hoeveel van die totale kode ooreenstem met hierdie gedeelte. Die reeks lynnommers of termometer dien as skakels wat die onderste raam opdateer na die ooreenstemmende gedeelte van die programkode in die lêer. Die gedeeltes wat met mekaar se teks ooreenstem, is dieselfde kleur, maar die swart teks is gedeeltes wat nie ooreenstemmend is nie. Die onderste rame bevat ook die grafiese termometer wat geklik kan word om die gedeelte van die kode mooi in die middel van die raam te laai.

Met die gebruik van die grafiese gebruikerskoppelvlak is daar 'n tekortkoming en dis die feit dat die gebruiker nie na die indeksblad terug kan navigeer nie. Gebruikers word dus aangeraai om op die skakel te klik en die resultate in die webblaaier te bestudeer en gebruik te maak van die terugpyltjie om weer by die indeks uit te kom. Die resultate is slegs vir 'n paar dae deur middel van hierdie webadres bekombaar en die gebruiker moet die resultate neerskryf of uitdruk indien nodig.

BYLAAG H

TEGNIESE ASPEKTE VAN SIMIAN

1 Verkryging en opstelling van Simian

Simian beskik oor verskillende lisensie-opsies afhangende van die beoogde gebruik:

a. **Nie-kommersiële projekte en evalueringslisensie**
   i. Simian kan gratis afgelaai word vanaf http://www.harukizaemon.com/simian/get_it_now.html as 'n .taz lêer wat deur WinZip oopgemaak kan word.
   ii. Hierdie lisensie is slegs geldig vir 'n tydperk van 15 dae waar Simian vir evalueringsdoeleindes gebruik word.

b. **Persoonlike / SOHO-lisensie**
   i. Simian kan slegs op een masjien gebruik word.
      - Geen verslagdoening kan gegenereer word nie.
      - Die lisensie is beskikbaar teen $99.
   ii. Hulp word aan jou verleen deur middel van e-pos.
   iii. Gebruiker ontvang slegs klein opdaterings van die sagteware.

c. **Projek / bou van bedienaarlisensie**
   i. Simian kan op enige aantal masjiene uitsluitlik vir die gelisensieerde projek of op 'n enkele masjien vir baie projekte gelaai word.
   ii. Die lisensie is beskikbaar teen $499.
   iii. Hulp word verleen deur middel van telefoon of e-pos.
   iv. Die gebruiker ontvang slegs klein opdaterings van die sagteware.

d. **Onderneming en ander lisensies**
   i. Simian kan op enige aantal masjiene op enige plek en projek gebruik word.
   ii. Vir verdere inligting oor reëlings en koste kan Simon Harris geskakel word. Vir hierdie eksperiment is die evalueringslisensie voldoende en Simian is dus net afgelaai. Die lêer word gedekompakteer en Simian kan in die bevelaanporring uitgevoer word deur die toepassingslêer of Java-lêer se instruksies uit te voer.

2 Uitleg van Simian

Om Simian uit te voer, moet die gids in die bevelaanporring eers gestel word na die baliegedeelte van die Simian-lêer, afhangende van waar die afgelaaide lêer gedekompakteer is. Simian kan in die bevelaanporring op een van die volgende twee maniere uitgevoer word:

1. deur gebruik te maak van die toepassingslêer `simian-[weergawe].exe` in die instruksie; of
2. deur gebruik te maak van die Java-lêer java –jar simian-[weergawe].jar in die instruksie.

Wanneer hierdie instruksie uitgevoer word, vertoon Simian se opsielys soos in Figuur 1. Indien die opsielys vertoon, beteken dit dat Simian suksesvol afvoer, maar verdere inligting word benodig. Hierdie opsies word volledig in Tabel 1 op die volgende bladsy bespreek.

```
Philip@PHILIP-PC C:\Users\Philip
$ cd C:\Simian\bin

Philip@PHILIP-PC C:\Simian\bin
$ java -jar simian-2.4.0.jar
Similarity Analyser 2.4.0 - http://www.harukizaemon.com/simian
Copyright (c) 2003-2015 Simon Harris.  All rights reserved.
Simian is not free unless used solely for non-commercial or evaluation purposes.
Usage: [options] [files]
    -balanceCurlyBraces[+|-]         Accounts for curly braces when breaking lines
    -balanceParentheses[+|-]         Accounts for parentheses when breaking lines
    -balanceSquareBrackets[+|-]      Accounts for square brackets when breaking lines
    -config=FNAME                    Reads the configuration from the specified file
    -defaultLanguage=LANG            Assumes files are in the specified language if none can be inferred
    -excludes=SPEC                   Excludes files matching the specified pattern
    -failOnDuplication[+|-|%]        Exits with a failure return code if duplication detected
    -formatter=TYPE[:FNAME]          Uses the specified output format when reporting
    -ignoreBlocks=START:END          Ignores all lines between START/END
    -ignoreCharacterCase[+|-]        Matches character literals irrespective of case
    -ignoreCharacters[+|-]           Completely ignores character literals
    -ignoreCurlyBraces[+|-]          Completely ignores curly braces
    -ignoreIdentifierCase[+|-]       Matches identifiers irresepctive of case
    -ignoreIdentifiers[+|-]          Completely ignores identifiers
    -ignoreLiterals[+|-]             Completely ignores all literals (strings, numbers and characters)
    -ignoreModifiers[+|-]            Ignores modifiers (public, private, static, etc.)
    -ignoreNumbers[+|-]              Completely ignores numbers
    -ignoreOverlappingBlocks[+|-]    Ignores blocks that wholly or partially overlap
    -ignoreRegions[+|-]              Ignores all lines between #region/#endregion
    -ignoreStringCase[+|-]           Matches string literals irrespective of case
    -ignoreStrings[+|-]              Completely ignores the contents of strings
    -ignoreSubtypeNames[+|-]         Matches on similar type names (eg. Reader and FilterReader)
    -ignoreVariableNames[+|-]        Completely ignores variable names (fields, parameters and locals)
    -includes=SPEC                   Including files matching the specified pattern
    -language=LANG                   Assumes ALL files are in the specified language
    -reportDuplicateText[+|-]        Prints the duplicate text in reports
    -threshold=COUNT                 Matches will contain at least the specified number of lines
```

Figuur 1: 'n Voorbeeld van die afvoer en vertoon van Simian se opsielys deur die Java-lêer te gebruik

Tabel 1: Die beskrywing van die verskillende opsies wat deur Simian aan die gebruiker verskaf word (Simian, 2003)

Opsie	Verstek waarde	Moontlike waardes	Beskrywing	Taal ondersteuning
balanceSquareBrackets	Vals	Waar/Vals	Verseker dat die uitdrukking binne blokhakies wat oor verskeie lyne strek as een beskou word.	Java, C#, C, C++, JavaScript, Ruby, Groovy
balanceCurlyBraces	Vals	Waar/Vals	Verseker dat die uitdrukking binne krulhakies wat oor verskeie lyne strek as een beskou word.	Ruby
balanceParentheses	Vals	Waar/Vals	Verseker dat die uitdrukking binne hakies wat oor verskeie lyne strek as een beskou word.	Java, C#, C, C++, JavaScript, COBOL, Ruby, SQL, Groovy
formatter	Geen	plain, xml, emacs, vs, yaml	Spesifiseer die formaat waarin die verwerkingsresultate geproduseer gaan	Almal
failOnDuplication	Waar	Waar/Vals	Veroorsaak dat die toetser die huidige proses staak indien 'n duplikaat opgespoor word.	Almal
ignoreBlocks	Geen	Stringwaarde	Ignoreer alle lyne wat gespesifiseer is deur die gegewe begin- of eindmerkers.	Almal
ignoreCharacters	Vals	Waar/Vals	Die karakters 'B' en 'R' word albei as dieselfde gesien.	Java, C#, C, C++, JavaScript, COBOL, Ruby, Groovy
ignoreCharacterCase	Waar	Waar/Vals	Hoofletters en kleinletters word as dieselfde gesien.	Java, C#, C, C++, JavaScript, COBOL, Ruby, Groovy
ignoreCurlyBraces	Vals	Waar/Vals	Hakies () word geïgnoreer	Java, C#, C, C++, JavaScript, Ruby, Groov

ignoreIdentifierCase	Waar	Waar/Vals	Identifiseerders word ongeag die geval met mekaar vergelyk.	Java, C#, C, C++, JavaScript, COBOL, Ruby, Groovy
ignoreNumbers	Vals	Waar/Vals	int y = 434; en int y = 2; sal as ooreenstemmend beskou word.	Java, C#, C, C++, JavaScript, COBOL, Ruby, SQL, Groovy
ignoreModifiers	Waar	Waar/Vals	Ignoreer *public*, *protected*, *stactic,* ens.	Java, C#, C, C++, JavaScript, Groovy
ignoreRegions	Vals	Waar/Vals	Ignoreer lyne tussen #streek / #eindstreek.	C#
ignoreStrings	Vals	Waar/Vals	MyVeranderlike en myveranderlike sal as ooreenstemmend beskou word.	Java, C#, C, C++, JavaScript, COBOL, Ruby, SQL, Groovy
ignoreStringCase	Waar	Waar/Vals	"Hello, World" en "HELLO, WORLD" sal as ooreenstemmend beskou word.	Java, C#, C, C++, JavaScript, COBOL, Ruby, SQL, Groovy
ignoreSubtypeNames	Vals	Waar/Vals	"BufferedReader, StringReader en Reader" sal almal as ooreenstemmend beskou word.	Java, C, Groovy
ignoreVariableNames	Vals	Waar/Vals	Ignoreer heeltemal veranderlike se name.	Java, C, Groovy
threshold	6	Heelgetal >= 2	Slegs ooreenkomste tussen hierdie aantal lyne word getoets.	-
language	Geen	Java, C#, CS, Csharp, C, C++, CCP, Cplusplus, JS, Javascript, COBOL, ABAP, RB, Ruby, VB, JSP, HTML, XML, Groovy, ASM390	Spesifiseer die taal wat gebruik moet word vir die toetsing. Dit wil sê dat al die lêers wat gegee word in hierdie gespesifiseerde taal geskryf is.	-

Wanneer die gebruiker opdragte deur middel van Simian met mekaar wil vergelyk, moet die instruksie wat die gebruiker in die bevelaanporring toevoer uit die volgende bestaan: `simian-[weergawe].exe of java –jar simian-[weergawe].jar` gevolg deur opsies waarna die gids waarin die lêers voorkom, gespesifiseer moet word. 'n Voorbeeld van so 'n instruksie word in Figuur 2 gegee, waar Simian geroep word deur die Java-lêer, gevolg deur die verandering van die drumpelwaarde na drie en die taal na C#. Die gids wat gespesifiseer word, moet gevolg word deur die lêerverlenging. In hierdie geval is al die lêers in die vouer C# opdragte met die .cs verlening en dus word *.cs bygevoeg aan die einde om te spesifiseer dat almal(*) C# lêers is.

```
Philip@PHILIP-PC C:\Simian\bin
$ java -jar simian-2.4.0.jar -threshold=3 -language=c# C:\Users\Philip\Desktop\Vlak1\1\*.cs
```

Figuur 2: Voorbeeld van die instruksie wat in Simian ingevoer word om C# opdragte te vergelyk

3 Die afvoer/verslaggewing van Simian

Die afvoer van Simian word gegee onder die instruksie in die bevelaanporring wanneer die instruksie soos in Figuur 2 uitgevoer word. Die verskillende opstellings wat vir die afvoering van hierdie ooreenkomste gemaak is, word eerste weergegee, waarna die aantal duplikate wat met die ooreenstemmende programmeringslyne vir elke opdrag gevind is, volg. Die totale lyne wat gedurende die ondersoek deurgegaan is en die tyd wat dit geneem het, word onderaan die afvoer weergegee. 'n Voorbeeld van die afvoer wat verkry word volgens die voorbeeldinstruksie word in Figuur 3 weergegee.

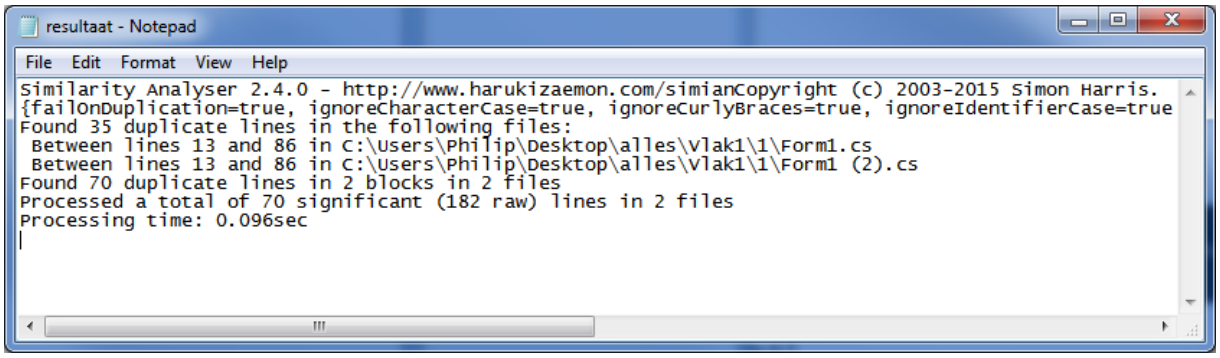
```
Philip@PHILIP-PC C:\Simian\bin
$ simian-2.4.0.exe -threshold=3 -language=c# C:\Users\Philip\Desktop\alles\Vlak1\1\*.cs
Similarity Analyser 2.4.0 - http://www.harukizaemon.com/simian
Copyright (c) 2003-2015 Simon Harris.  All rights reserved.
Simian is not free unless used solely for non-commercial or evaluation purposes.
{failOnDuplication=true, ignoreCharacterCase=true, ignoreCurlyBraces=true, ignoreIdentifierCas
e=true, ignoreModifiers=true, ignoreStringCase=true, language=C#, threshold=3}
Found 35 duplicate lines in the following files:
 Between lines 13 and 86 in C:\Users\Philip\Desktop\alles\Vlak1\1\Form1.cs
 Between lines 13 and 86 in C:\Users\Philip\Desktop\alles\Vlak1\1\Form1 (2).cs
Found 70 duplicate lines in 2 blocks in 2 files
Processed a total of 70 significant (182 raw) lines in 2 files
Processing time: 0.121sec
```

Figuur 3: Voorbeeld van Simian se afvoer in die bevelaanporring

Hierdie afvoer wat in die bevelaanporring gegee word, kan ook gestoor word as 'n teks, *xml, emacs, visual studio of yam*-lêer deur gebruik te maak van die *formatter*-opsie in die instruksie, soos byvoorbeeld as die gebruiker die afvoer as 'n tekslêer genaamd resultaat wil stoor, sal die instruksie soos in Figuur 4 gebruik word. Die resultaat in die tekslêer sal binne die balievouer van Simian gevind word en geen afvoer word vertoon in die bevelaanporring nie. 'n Voorbeeld van die afvoer wat gestoor in die tekslêer is, word in Figuur 5 gegee.

```
Philip@PHILIP-PC C:\Simian\bin
$ simian-2.4.0.exe -threshold=3 -language=c# -formatter=plain:resultaat C:\Users\Philip\D
esktop\alles\Vlak1\1\*.cs
```

Figuur 4: 'n Voorbeeld van die instruksie wat die afvoer in 'n tekslêer, genaamd resultaat stoor, deur gebruik te maak van die *formatter*-opsie van Simian.



Figuur 5: 'n Voorbeeld van die tekslêer wat die afvoer van Simian bevat

4 Bibliografie

Simian. 2003. Simian - Similarity Analyser: Purpose. http://www.harukizaemon.com/simian/index.html Datum van gebruik: 25 Jun. 2017.

BYLAAG I

VOLLEDIGE VRAELYS

Plagiaat in programmeringsmodules / Plagiarism in programming modules

As 'n navorser wil ek graag jou mening oor plagiaat in programeringsmodules verkry.
Voltooi asseblief die volgende vraelys volledig. (Tot en met die nuwe groen vorm) /

As a researcher I would like to obtain your views on plagiarism in programming modules.

Please complete the following questionnaire completely (Untill the new green form).

Vir verdere inligting: / For more information: http://tinyurl.com/qq53lqq

* Required

Hiermee verklaar ek dat / I hereby declare that: *

- ☐ • ek die inhoud van die navorsingsprojek verstaan. / I understand the content of this research project.
- ☐ • ek vrywillig deelneem aan die projek. / I participate voluntarily in the project.
- ☐ • ek toestemming gee dat inligting verkry gebruik mag word vir navorsingsdoeleindes. / I grant permission that information obtained may be used for research purposes.
- ☐ • ek verstaan dat my privaatheid en die vertroulikheid van die inhoud gerespekteer sal word./ I understand that my privacy and the confidentially of the contents will be respected.

Algemene inligting / General information

Voltooi asseblief die volgende vrae en beweeg dan na die volgende afdeling. /
Please complete the following questions and then move on to the next section.

1. **Geslag / Gender** *
   - ◯ Manlik / Male
   - ◯ Vroulik / Female

2. **Hoe sal jy jou etniese agtergrond klassifiseer? / How would you classify your ethnic background?** *
   - ◯ Swart / Black
   - ◯ Kleurling / Coloured
   - ◯ Wit / White
   - ◯ Indiër / Indian
   - ◯ Ek verkies om nie te sê nie / I do not wish to say
   - ◯ Other: _____

3. **Ek het IT as 'n vak op skool gehad. / I had IT as a school subject.** *

- ◯ Ja / Yes
- ◯ Nee / No

4. **Huidige akademiese jaar? / Current academic year?** *

- ◯ 1ste jaar / 1st year
- ◯ 2de jaar / 2nd year
- ◯ 3de jaar / 3rd year
- ◯ Honneurs/ Honours

5. **Studierigting? / Field of study?** *

- ◯ BCom
- ◯ BIng
- ◯ BSc
- ◯ BSc I.T.

6. **Na my mening het ek genoeg tyd tot my beskikking om my programmeringsopdragte te voltooi. /**
In my view I have enough time to my disposal to complete my programming assignments. *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

7. **Hoe sal jy jou tydsbestuur beskryf? / How would you describe your time management?** *

- ◯ Goed / Good
- ◯ Matig / Moderate
- ◯ Sleg / Bad

8. **Gradeer jou gemiddelde akademiese prestasie in jou programmeringsmodules. / Please grade your average academic performance in your programming modules.** *

- ◯ 75 - 100%
- ◯ 50 - 74%
- ◯ 0 - 49%

9. **Ek verstaan wat plagiaat behels. / I understand what plagiarism entails.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

10. **Plagiaat is om iemand anders se werk as jou eie voor te hou. / Plagiarism is to present someone else's work as your own.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

11. **Plagiaat kan gesien word as 'n misdaad. / Plagiarism can be viewed as a crime.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

12. **Ek het al gehoor van die Turnitin-plagiaatherkenningstelsel. / I've heard of the Turnitin plagiarism detection system before.** *

- ◯ Ja / Yes
- ◯ Nee / No

13. **Die Turnitin-plagiaatherkenningstelsel is van belang om onetiese gedrag te ontbloot. / The Turnitin plagiarism detection system is important to expose unethical behaviour.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

14. **Ek mag kopieer solank ek net erkenning gee. / I may copy as long as I give recognition.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

15. **Dosente lig ons in oor plagiaat. / Lecturers inform us about plagiarism.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

16. **Dosente lig ons in oor die gevolge van plagiaat. /**
Lecturers inform us about the consequences of plagiarism. *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

17. **Die programmeringsmodules se studiegidse bevat 'n verduideliking van plagiaat spesifiek in programmering. /**
The programming modules study guides contain an explanation specifically for plagiarism in programming. *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

18. **Die universiteit beskik oor 'n geskikte plagiaatbeleid? /**
The university has a suitable plagiarism policy? *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

19. **Die universiteit beskik oor 'n geskikte plagiaatbeleid ten opsigte van programmering? / The university has a suitable plagiarism policy regarding programming?**

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

20. **Kopiëring van programmering is anders as die van tekskopiëring. / Copying programming is different from that of text copying.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

21. **Solank ek die kodering van 'n gekopieërde programmeringsopdrag verstaan, kan ek dit as my eie indien. / As long as I understand the coding of a copied programming assignment I can submit it as my own.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

Plagiaat in programmering / Plagiarism in programming

Voltooi asseblief die volgende vrae en beweeg dan na die volgende afdeling. / Please complete the following questions and then move on to the next section.

22. **Ek het 'n programmeringsopdrag gekopieer en as my eie ingehandig. / I have copied a programming assignment and submitted it as my own.** *

- ◯ Nooit / Never
- ◯ Een of twee keer / Once or twice
- ◯ Drie tot vyf keer/ Three to five times
- ◯ Ses tot nege keer/ Six to nine times
- ◯ Meer as tien keer/ More than ten times

23. **Volgens jou wat is die mees algemene metode om plagiaat te pleeg in programmering? / According to you what is the most common method to commit plagiarism in programming?** *

- ◯ Kopieer die hele program / Copy the whole program
- ◯ Kopieer kode van 'n klasmaat se opdrag in 'n nuwe projek / Copy code of a classmate's assignment to a new project
- ◯ Kopieer kode vanaf 'n foto / Copy code from a photo
- ◯ Kopieer kode vanaf die Internet / Copy code from the Internet
- ◯ Verander kode / Change kode
- ◯ Other: _____

24. **Volgens jou wat is die mees algemene manier om kode te verander om nie uitgevang te word nie? / According to you what is the most common way to change code in order to not get caught?** *

- ◯ Verander of byvoeg van kommentaar / Change or add comments
- ◯ Verander veranderlikes se name / Change variable's names
- ◯ Verander die spasiëring van die kode / Change the spacing of the code
- ◯ Verander net die program se naam / Just change the program's name
- ◯ Verander die uitleg van die program (GUI) / Change the layout of the program (GUI)

25. **In jou opinie watter ander metodes kan gebruik word om nie uitgevang te word nie? / In your opinion what other methods can be used to not get caught?**

_____

_____

_____

_____

26. **As ek 'n gekopieërde program as my eie sou inhandig sal ek uitgevang word. / If I copy a program and hand it in as my own, I will be caught.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

27. **Ek sal 'n programmeringstoets of -eksamen kopieer en inhandig. / I will copy and submit a programming test or examination.** *

- ◯ Sal dit nie eers oorweeg nie / Would not even consider
- ◯ Sal dit nie doen nie - ek is te bang ek word uitgevang / Would not do it - too afraid to be caught
- ◯ Sal dit doen al is ek bang ek word uitgevang / Would do it although I'm afraid I will get caught
- ◯ Sal dit doen as ek die kans kry / Would do it if I had the change
- ◯ Sal dit doen na sorgvuldige aanpassing of verdoeseling / Will do this after careful adjustment or concealment
- ◯ Other: _____

28. **Ek het vriende/kennisse wie se programme ek kan kopieer. / I have friends/acquaintances whose programs I can copy.**

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

Verskaffing van program / Providing the program

29. **Ek sal vir iemand anders my programmeringsopdragte gee as hulle daarvoor vra. / I will give my programming assignments to someone else if they ask for it.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

30. **Ek het al my kode vir 'n medestudent gegee. / I have given my code to a fellow student before.** *

- ◯ Nooit / Never
- ◯ Een of twee keer / Once or twice
- ◯ drie tot vyf keer/ Three to five times .
- ◯ ses tot nege keer/ Six to nine times
- ◯ Meer as tien keer/ More than ten times

31. **Solank as wat my medestudent die kodering verstaan, kan ek dit maar vir hom/haar gee. / As long as my fellow student understands the coding I may give it to him/her.** *

- ◯ Stem volkome saam / Strongly agree
- ◯ Stem saam / Agree
- ◯ Neutraal / Neutral
- ◯ Stem nie saam nie / Disagree
- ◯ Stem glad nie saam nie / Strongly disagree

32. **Nog enige opmerkings aangaande kopiëring of plagiaat. Any comments regarding copying or plagiarism.**

Programmeringsopdrag kopiëring en inhandiging. / Programming assignment copying and submitting.

Het wel al 'n programmeringsopdrag gekopieer en ingehandig. / Have copied a programming assignment and submitted it.

33. **Watter metode(s) het jy gebruik om plagiaat te pleeg ? / Which method(s) did you use to commit plagiarism?** *

- [ ] Kopieer die hele program / Copy the whole program
- [ ] Kopieer kode van 'n klasmaat se opdrag in 'n nuwe projek / Copy code of a classmate's assignment to a new project
- [ ] Kopieer kode vanaf 'n foto wat ek geneem het / Copy code from a photo I have taken
- [ ] Kopieer kode vanaf die Internet / Copy code from the Internet
- [ ] Other: _____

Dien as bron van programmeringsopdrag / Serves as source of programming assignment.

Het as bron gedien / Was the source of the given program

34. **Watter manier(e) het jy te werk gegaan om die kodering te verskaf aan jou medestudent? / Which approach(es) did you follow to provide the coding to your fellow student?** *

- [ ] Deur die kode af te neem en te stuur oor sosiale media / Through taking a photo of the coding and sending it through social media
- [ ] Deur die kode op 'n geheuestokkie te laai / By loading the code on a memory stick
- [ ] Kode aan te stuur deur middel van e-pos / Sending the code via e-mail
- [ ] Deur die kode op 'n cloud te laai (bv. Google drive of Dropbox) / By placing te code on a cloud (e.g. Google drive of Dropbox)
- [ ] Other: _____

BYLAAG J

FAKTORE SE ITEMS MET BESKRYWENDE STATISTIEK

Faktor	Items	Gemiddelde
Kennis_en_inligting	Ek verstaan wat plagiaat behels.	4.607
	Plagiaat is om iemand anders se werk as jou eie voor te hou.	4.637
	Plagiaat kan beskou word as 'n misdaad.	4.404
	Die Turnitin plagiaatherkenningstelsel is van belang om onetiese gedrag te ontbloot.	4.052
	Dosente lig ons in oor plagiaat.	4.221
	Dosente lig ons in oor die gevolge van plagiaat.	4.191
	Die programmeringsmodules se studiegidse bevat 'n verduideliking van plagiaat spesifiek in programmering.	3.966
	Die universiteit beskik oor 'n geskikte plagiaatbeleid.	4.431
	Die universiteit beskik oor 'n geskikte plagiaatbeleid ten opsigte van programmering.	4.038
Verskaffing_van_ programkode	Ek het vriende/kennisse wie se programme ek kan kopieer.	2.609
	Ek sal vir iemand anders my programmeringsopdragte gee as hulle daarvoor vra.	2.618
	Solank as wat my medestudent die kodering verstaan, kan ek dit maar vir hom/haar gee.	4.169

BYLAAG K

AANSOEKVORM VIR *CODEMATCH®* SE UNIVERSITEITSPROGRAM LISENSIE

SOFTWARE ANALYSIS AND FORENSIC ENGINEERING

UNIVERSITY PROGRAM TERMS

The terms and conditions below ("University Program Terms") amend the End User License Agreement ("Agreement") to provide the terms and conditions applicable to use of the Software by Licensee, when Licensee is a student or a teacher at an institution of higher education participating in SAFE's University Program. Except as amended by the University Program Terms, all terms and conditions of the Agreement shall remain in full force and effect. In the event of a conflict between a provision or provisions of the University Program Terms and a provision or provisions of the Agreement, the provision or provisions of the University Program Terms will control. The University Program Terms shall remain in effect until terminated as provided herein. Capitalized terms not defined herein have the meanings specified in the Agreement.

1. License

Subject to the terms and conditions of this Agreement, SAFE hereby grants to Licensee for the License Term (defined below), a limited, non-transferable, non-assignable, non-exclusive right and license, only for purposes of course work or teaching in connection with a university-sponsored class, or for academic research either sponsored by or conducted under the auspices of Licensee, to (a) install and use the Software, and (b) reproduce and distribute a reasonable number of copies of the Documentation.

2. License Term and Termination

For purposes of the University Program, "License Term" means one month from the date the Authorization Key is emailed to Licensee, unless otherwise agreed to in writing, renewable each month thereafter. This Agreement will terminate upon written notice by either party, unless earlier terminated in accordance with this Agreement.

3. License Restrictions

Licensee shall not (i) allow access to the Software by any user not registered for a course or participating in an academic research project for which use of the Software has been authorized; or (ii) use the Software to design, develop, or otherwise use in connection with any commercial products.

4. Technical Liaison

Licensee shall appoint a Technical Liaison who will serve as the single point of contact between SAFE and Licensee with respect to the subject matter of the University Program Terms. The Technical Liaison will coordinate installation and maintenance of the Software, communicate with SAFE regarding license procedures, administer Licensee's obligations under the University Program Terms, and respond to inquiries by SAFE related to the subject matter of the University Program Terms.

5. Technical Support

Unless otherwise agreed in writing, SAFE will accept emails only from the technical representative designated by Licensee ("Technical Liaison"). No technical support will be provided other than emails from the Technical Liaison relating to installation of the Software. Software upgrades may be obtained from the SAFE Web Site.

6. Use of Licensee Name.

Licensee gives SAFE the right to disclose that Licensee is a member of the University Program.

7. Acknowledge use of SAFE products.

When SAFE products are used, licensee will acknowledge SAFE in any research, conference papers, or course materials and provide SAFE with a copy of the finished work.

If you agree with the University Program Terms, please execute the University Program Terms in the space provided below and return the executed University Program Terms to SAFE via facsimile at (408) 741-5231 or email to sales@SAFE-corp.biz.

ACKNOWLEDGED AND AGREED TO BY:

University representative	**SAFE representative**
Signature J Lieberberg	Signature
Date 24/05/2017	Date 5/24/2017
Name JANET	Name ROBERT ZEIDMAN
Title DR.	Title PRESIDENT
University NORTH-WEST UNIVERSITY	SAFE Corporation
Address1 11 HOFFMAN ST	40863 Stevens Creek Blvd.
Address2 BUILDING G3 ROOM 626	Suite 456
City POTCHEFSTROOM	Cupertino
U.S. State/Country SOUTH AFRICA	CA
Postal Code/Zip Code 2520	95014
Email Address Janet.Lieberberg@nwu.co.za	Email Address bobe SAFE-corp.com

BYLAAG L

BEWYS VAN ETIEK OPLEIDING

NORTH-WEST UNIVERSITY ®
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
POTCHEFSTROOM CAMPUS

Private Bag X6001, Potchefstroom
South Africa 2520

Tel: 018 299-1111/2222
Web: http://www.nwu.ac.za

Health Sciences Ethics Office for Research, Training and Support
Tel: 018 299 2092
Fax:
Email: minrie.greeff@nwu.ac.za

10 February 2017

Dear Mnr Philip Botes (HPCSA registration number: _____)

PROOF OF ATTENDANCE

This letter certifies that you have attended the 2 day ethics training, entitled:

The Basics of Health Research Ethics

(Accreditation number: UP1163 from University of Pretoria CPD accreditation department)

presented by Prof Minrie Greeff (Head of the Health Sciences Ethics Office for Research, Training and Support) on **23 and 24 January 2017**.

This proof of attendance, as recognised by HREC and the Ethics Office, NWU, is valid for 3 years and expires on the 24th of January 2020. Where applicable, Ethics CEUs awarded: ***27 Ethics CEUs***

Yours sincerely

Prof Minrie Greeff
Head of Health Sciences Ethics
Office for Research, Training and Support

Prof Awie Kotzé
Dean of Faculty of Health Sciences

BYLAAG M

ETIEK GOEDKEURINGSERTIFIKAAT VAN STUDIE

NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT

Private Bag X6001, Potchefstroom,
South Africa, 2520

Tel: (018) 299-4900
Faks: (018) 299-4910
Web: http://www.nwu.ac.za

Institutional Research Ethics Regulatory Committee
Tel: +27 18 299 4849
Email : Ethics@nwu.ac.za

ETHICS APPROVAL CERTIFICATE OF STUDY

Based on approval by the **Ethics Committee of the Faculty of Education Sciences (ESREC) on 03/08/2017** after being reviewed at the meeting held on **25/05/2017**, the North-West University Institutional Research Ethics Regulatory Committee (NWU-IRERC) hereby **approves** your study as indicated below. This implies that the NWU-IRERC grants its permission that, provided the special conditions specified below are met and pending any other authorisation that may be necessary, the study may be initiated, using the ethics number below.

Study title: Die opsporing van plagiaat in grafiese koppelvlakprogrammeringsopdragte.

Study Leader/Supervisor: Dr JA Liebenberg
Research Team: P Botes & Prof G Drevin

Ethics number: N W U - HS - 2 0 1 7 - 0 0 7 4
(Institution - Study Number - Year - Status)

Status: S = Submission; R = Re-Submission; P = Provisional Authorisation; A = Authorisation

Application Type: N/A

Commencement date: 2017-05-25 **Expiry date:** 2017-11-30 **Risk:** N/A

Special conditions of the approval (if applicable):

- Translation of the informed consent document to the languages applicable to the study participants should be submitted to the ESREC (if applicable).
- Any research at governmental or private institutions, permission must still be obtained from relevant authorities and provided to the ESREC. Ethics approval is required BEFORE approval can be obtained from these authorities.

General conditions:

While this ethics approval is subject to all declarations, undertakings and agreements incorporated and signed in the application form, please note the following:

- *The study leader (principle investigator) must report in the prescribed format to the NWU-IRERC via ESREC:*
  - *annually (or as otherwise requested) on the progress of the study, and upon completion of the project*
  - *without any delay in case of any adverse event (or any matter that interrupts sound ethical principles) during the course of the project.*
  - *Annually a number of projects may be randomly selected for an external audit.*
- *The approval applies strictly to the proposal as stipulated in the application form. Would any changes to the proposal be deemed necessary during the course of the study, the study leader must apply for approval of these changes at the ESREC. Would there be deviated from the study proposal without the necessary approval of such changes, the ethics approval is immediately and automatically forfeited.*
- *The date of approval indicates the first date that the project may be started. Would the project have to continue after the expiry date, a new application must be made to the NWU-IRERC via ESREC and new approval received before or on the expiry date.*
- *In the interest of ethical responsibility the NWU-IRERC and ESREC retains the right to:*
  - *request access to any information or data at any time during the course or after completion of the study;*
  - *to ask further questions, seek additional information, require further modification or monitor the conduct of your research or the informed consent process*
  - *withdraw or postpone approval if:*
    - *any unethical principles or practices of the project are revealed or suspected,*
    - *it becomes apparent that any relevant information was withheld from the ESREC or that information has been false or misrepresented,*
    - *the required annual report and reporting of adverse events was not done timely and accurately,*
    - *new institutional rules, national legislation or international conventions deem it necessary.*
- *ESREC can be contacted for further information or any report templates via Erna.Greyling@nwu.ac.za or 018 299 4656*

The IRERC would like to remain at your service as scientist and researcher, and wishes you well with your project. Please do not hesitate to contact the IRERC or ESREC for any further enquiries or requests for assistance.

Yours sincerely

Prof LA Du Plessis
Digitally signed by Prof LA Du Plessis
Date: 2017.08.05 11:29:43 +02'00'

Prof Linda du Plessis
Chair NWU Institutional Research Ethics Regulatory Committee (IRERC)

BYLAAG N

STATISTIESE KONSULTASIEDIENS

NWU Ethics Application

Project Leader (Title, Initials & Surname)	Project Title (see § 3.1)
Dr JA Liebenberg	Die opsporing van plagiaat in grafiese koppelvlakprogrammeringsopdragte

NWU Ethics Number

N	W	U	-						-			-		

Institution | Project Number | Year | Status

Status: S = Submission; R = Re-Submission; P = Provisional Authorisation; A = Authorisation

(for office use only)

Sec 8e: Statistical Consultation Service

The statistician of the Statistical Consultation Service of the North-West University completes this section (where applicable).

More information

Prior consultation with Statistical Consultation Service can eliminate many problems, simplify and expedite the evaluation and also prevent applications from being returned due to poor project planning and/or statistical justifiability. Where the project leader has sufficient statistical expertise at his disposal, this is, however, not compulsory.

The Ethics Committee relies completely on the professional judgement of the statistician.

Have you ascertained the experimental design of the study and is it statistically justifiable according to your judgement?
(Please mark with **X** *in the appropriate box and provide details)*

Yes	No	Remarks
☒x	☐	.

Name (Title, Full Names & Surname)	Qualifications
Prof Susanna Maria Ellis (Pr Sci Nat)	PhD (Statistics)

Signature	Date
SM Ellis	2017-06-29 (ccyy-mm-dd)

Remember to save your document regularly as you complete it!

BYLAAG O

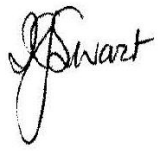
TAALKUNDIGE VERSORGING

Hiermee bevestig ek, Isabella Johanna Swart, geregistreer by en geakkrediteer as professionele vertaler deur die Suid-Afrikaanse Vertalersinstituut, registrasienommer 1001128, dat ek die volgende verhandeling taalversorg het.

Die opsporing van plagiaat in grafiese gebruikerskoppelvlakprogrammeringsopdragte

deur

Philip Botes



28 Februarie 2018
Datum

Dr Isabel J Swart

Poinsettiaslot 23
Van der Stel Park
Dormehlsdrift
GEORGE
6529
Tel: (044) 873 0111
Sel: 082 718 4210
e-pos: isaswart@telkomsa.net

BYLAAG P

RESULTATE VAN DIE PROGRAMKODEPLAGIAATHERKENNINGSTELSELS

Die resultate van ACNP word as volg weergegee. Die aantal karakters wat ooreenstem met die opstelling 1 tot 3 asook die ooreenkomspersentasie van al twaalf programmeringsopdragte met tipe veranderingss 1 tot 10 word in die onderstaande tabelle weergegee. Toetsgeval 1 wat bestaan uit die vergelyking van die programkodelêers word eers weergegee, daarna toetsgeval 2 wat vergelyking van die programkodelêers en die ontwerpkodelêers bevat.

Toetsgeval 1 - Tipe verandering 1 (Kommentaar & indentering)							
Programmerings opdrag	**Totale aantal karakters**	**Opstelling 1 se aantal karakters**	**Opstelling 2 se aantal karakters**	**Opstelling 3 se aantal karakters**	**Opstelling 1 se ooreenkoms persentasie**	**Opstelling 2 se ooreenkoms persentasie**	**Opstelling 3 se ooreenkoms persentasie**
1	2221	537	806	2221	24.18%	36.29%	100.00%
2	1894	225	378	1894	11.88%	19.96%	100.00%
3	2078	423	717	2078	20.36%	34.50%	100.00%
4	2369	860	694	2369	36.30%	29.30%	100.00%
5	1875	975	706	1875	52.00%	37.65%	100.00%
6	1777	401	736	1777	22.57%	41.42%	100.00%
7	1926	411	639	1926	21.34%	33.18%	100.00%
8	1894	820	1176	1894	43.29%	62.09%	100.00%
9	1889	275	538	1889	14.56%	28.48%	100.00%
10	2187	290	715	2187	13.26%	32.69%	100.00%
11	1866	161	512	1866	8.63%	27.44%	100.00%
12	2250	337	594	2250	14.98%	26.40%	100.00%
Gemiddeld:	**2018.83**	**476.25**	**684.25**	**2018.83**	**23.61%**	**34.12%**	**100.00%**

Toetsgeval 1 - Tipe verandering 2 (Veranderlikes)							
Programmerings opdrag	**Totale aantal karakters**	**Opstelling 1 se aantal karakters**	**Opstelling 2 se aantal karakters**	**Opstelling 3 se aantal karakters**	**Opstelling 1 se ooreenkoms persentasie**	**Opstelling 2 se ooreenkoms persentasie**	**Opstelling 3 se ooreenkoms persentasie**
1	2221	653	363	363	29.40%	16.34%	16.34%
2	1894	462	203	334	24.39%	10.72%	17.63%
3	2078	576	320	319	27.72%	15.40%	15.35%
4	2369	433	321	320	18.28%	13.55%	13.51%
5	1875	642	426	406	34.24%	22.72%	21.65%
6	1777	413	310	309	23.24%	17.45%	17.39%
7	1926	562	319	319	29.18%	16.56%	16.56%
8	1894	551	318	636	29.09%	16.79%	33.58%
9	1889	526	323	309	27.85%	17.10%	16.36%
10	2187	541	321	320	24.74%	14.68%	14.63%
11	1866	255	137	136	13.67%	7.34%	7.29%
12	2250	457	337	327	20.31%	14.98%	14.53%
Gemiddeld:	**2018.83**	**505.92**	**308.17**	**341.50**	**25.18%**	**15.30%**	**17.07%**

Toetsgeval 1 - Tipe verandering 3 (Verklarende stellings)							
Programmerings opdrag	**Totale aantal karakters**	**Opstelling 1 se aantal karakters**	**Opstelling 2 se aantal karakters**	**Opstelling 3 se aantal karakters**	**Opstelling 1 se ooreenkoms persentasie**	**Opstelling 2 se ooreenkoms persentasie**	**Opstelling 3 se ooreenkoms persentasie**
1	2221	1102	900	571	49.62%	40.52%	25.71%
2	1894	563	590	302	29.73%	31.15%	15.95%
3	2078	1392	1016	831	66.99%	48.89%	39.99%
4	2369	1388	827	553	58.59%	34.91%	23.34%
5	1875	946	674	516	50.45%	35.95%	27.52%
6	1777	1392	822	613	78.33%	46.26%	34.50%
7	1926	1372	1033	809	71.24%	53.63%	42.00%
8	1894	547	912	1398	28.88%	48.15%	73.81%
9	1889	963	726	500	50.98%	38.43%	26.47%
10	2187	1318	811	617	60.27%	37.08%	28.21%
11	1866	580	311	250	31.08%	16.67%	13.40%
12	2250	1191	652	605	52.93%	28.98%	26.89%
Gemiddeld:	**2018.83**	**1062.83**	**772.83**	**630.42**	**52.42%**	**38.39%**	**31.48%**

Toetsgeval 1 - Tipe verandering 4 (Programmodules)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	2221	1915	989	721	86.22%	44.53%	32.46%
2	1894	1699	807	662	89.70%	42.61%	34.95%
3	2078	1593	1090	768	76.66%	52.45%	36.96%
4	2369	2118	1376	894	89.40%	58.08%	37.74%
5	1875	1615	1034	716	86.13%	55.15%	38.19%
6	1777	1410	729	582	79.35%	41.02%	32.75%
7	1926	922	644	571	47.87%	33.44%	29.65%
8	1894	815	638	499	43.03%	33.69%	26.35%
9	1889	1804	1103	903	95.50%	58.39%	47.80%
10	2187	1634	950	820	74.71%	43.44%	37.49%
11	1866	1535	843	624	82.26%	45.18%	33.44%
12	2250	2155	1210	950	95.78%	53.78%	42.22%
Gemiddeld:	**2018.83**	**1601.25**	**951.08**	**725.83**	**78.89%**	**46.81%**	**35.83%**

Toetsgeval 1 - Tipe verandering 5 (Stellings)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	2221	1686	1062	898	75.91%	47.82%	40.43%
2	1894	1565	932	757	82.63%	49.21%	39.97%
3	2078	1862	1263	1002	89.61%	60.78%	48.22%
4	2369	2269	1494	1053	95.78%	63.06%	44.45%
5	1875	1443	898	853	76.96%	47.89%	45.49%
6	1777	1400	843	739	78.78%	47.44%	41.59%
7	1926	1540	1000	894	79.96%	51.92%	46.42%
8	1894	1065	769	824	56.23%	40.60%	43.51%
9	1889	1630	1057	926	86.29%	55.96%	49.02%
10	2187	1657	962	849	75.77%	43.99%	38.82%
11	1866	1588	951	781	85.10%	50.96%	41.85%
12	2250	2043	1228	969	90.80%	54.58%	43.07%
Gemiddeld:	**2018.83**	**1645.67**	**1038.25**	**878.75**	**81.15%**	**51.18%**	**43.57%**

Toetsgeval 1 - Tipe verandering 6 (Besluit logika)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	2221	1913	1207	982	86.13%	54.34%	44.21%
2	1894	1552	1067	856	81.94%	56.34%	45.20%
3	2078	1646	1291	1031	79.21%	62.13%	49.62%
4	2369	2243	1566	988	94.68%	66.10%	41.71%
5	1875	1711	1184	921	91.25%	63.15%	49.12%
6	1777	1413	988	767	79.52%	55.60%	43.16%
7	1926	1590	1268	987	82.55%	65.84%	51.25%
8	1894	1598	1255	1060	84.37%	66.26%	55.97%
9	1889	1552	1225	958	82.16%	64.85%	50.71%
10	2187	2003	1194	978	91.59%	54.60%	44.72%
11	1866	1636	1082	889	87.67%	57.98%	47.64%
12	2250	2231	1372	1041	99.16%	60.98%	46.27%
Gemiddeld:	**2018.83**	**1757.33**	**1224.92**	**954.83**	**86.69%**	**60.68%**	**47.46%**

Toetsgeval 1 - Tipe verandering 7 (Skuif van kontroles)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	3891	3346	2102	1497	85.99%	54.02%	38.47%
2	4434	4294	2760	1579	96.84%	62.25%	35.61%
3	4364	3597	2298	1446	82.42%	52.66%	33.13%
4	4498	3873	2474	1231	86.10%	55.00%	27.37%
5	4461	3803	2405	1541	85.25%	53.91%	34.54%
6	4818	3715	2369	1650	77.11%	49.17%	34.25%
7	4816	4467	2772	1566	92.75%	57.56%	32.52%
8	4321	3567	2268	1427	82.55%	52.49%	33.02%
9	4848	4030	2555	1562	83.13%	52.70%	32.22%
10	4720	3892	2450	1324	82.46%	51.91%	28.05%
11	4434	3617	2351	1253	81.57%	53.02%	28.26%
12	4330	3566	2267	1486	82.36%	52.36%	34.32%
Gemiddeld:	**4494.58**	**3813.92**	**2422.58**	**1463.50**	**84.88%**	**53.92%**	**32.65%**

Toetsgeval 1 - Tipe verandering 8 (Herbenaming van GGK kontroles)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	3891	1478	944	427	37.99%	24.26%	10.97%
2	4434	1487	953	436	33.54%	21.49%	9.83%
3	4364	1488	954	437	34.10%	21.86%	10.01%
4	4498	1499	965	437	33.33%	21.45%	9.72%
5	4461	1499	965	448	33.60%	21.63%	10.04%
6	4818	1478	944	427	30.68%	19.59%	8.86%
7	4816	1496	962	437	31.06%	19.98%	9.07%
8	4321	1487	953	436	34.41%	22.06%	10.09%
9	4848	1478	944	427	30.49%	19.47%	8.81%
10	4720	1499	964	438	31.76%	20.42%	9.28%
11	4434	1487	953	436	33.54%	21.49%	9.83%
12	4330	1494	960	445	34.50%	22.17%	10.28%
Gemiddeld:	**4494.58**	**1489.17**	**955.08**	**435.92**	**33.25%**	**21.32%**	**9.73%**

Toetsgeval 1 - Tipe verandering 9 (Verandering van teks in GGK)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	3891	3964	2637	1701	101.88%	67.77%	43.72%
2	4434	4462	2912	2308	100.63%	65.67%	52.05%
3	4364	4283	2831	1850	98.14%	64.87%	42.39%
4	4498	4302	2846	2165	95.64%	63.27%	48.13%
5	4461	4334	2877	2337	97.15%	64.49%	52.39%
6	4818	3994	2614	2237	82.90%	54.25%	46.43%
7	4816	4627	3027	2212	96.08%	62.85%	45.93%
8	4321	4200	2745	1827	97.20%	63.53%	42.28%
9	4848	4594	1669	2365	94.76%	34.43%	48.78%
10	4720	4399	2022	1909	93.20%	42.84%	40.44%
11	4434	4357	2904	2467	98.26%	65.49%	55.64%
12	4330	4250	2784	2102	98.15%	64.30%	48.55%
Gemiddeld:	**4494.58**	**4313.83**	**2655.67**	**2123.33**	**96.17%**	**59.48%**	**47.23%**

Toetsgeval 1 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)							
Programmerings opdrag	**Totale aantal karakters**	**Opstelling 1 se aantal karakters**	**Opstelling 2 se aantal karakters**	**Opstelling 3 se aantal karakters**	**Opstelling 1 se ooreenkoms persentasie**	**Opstelling 2 se ooreenkoms persentasie**	**Opstelling 3 se ooreenkoms persentasie**
1	3891	1200	767	427	30.84%	19.71%	10.97%
2	4434	1209	776	436	27.27%	17.50%	9.83%
3	4364	1210	777	437	27.73%	17.80%	10.01%
4	4498	1220	787	437	27.12%	17.50%	9.72%
5	4461	1499	965	448	33.60%	21.63%	10.04%
6	4818	1200	767	427	24.91%	15.92%	8.86%
7	4816	1218	785	437	25.29%	16.30%	9.07%
8	4321	1209	776	436	27.98%	17.96%	10.09%
9	4848	1200	767	427	24.75%	15.82%	8.81%
10	4720	1221	787	438	25.87%	16.67%	9.28%
11	4434	1209	776	436	27.27%	17.50%	9.83%
12	4330	1216	783	445	28.08%	18.08%	10.28%
Gemiddeld:	**4494.58**	**1234.25**	**792.75**	**435.92**	**27.56%**	**17.70%**	**9.73%**

Toetsgeval 2 - Tipe verandering 1 (Kommentaar & indentering)							
Programmerings opdrag	**Totale aantal karakters**	**Opstelling 1 se aantal karakters**	**Opstelling 2 se aantal karakters**	**Opstelling 3 se aantal karakters**	**Opstelling 1 se ooreenkoms persentasie**	**Opstelling 2 se ooreenkoms persentasie**	**Opstelling 3 se ooreenkoms persentasie**
1	6112	4480	5031	6112	73.30%	82.31%	100.00%
2	6328	4652	5492	6328	73.51%	86.79%	100.00%
3	6258	4797	5436	6258	76.65%	86.86%	100.00%
4	6392	4938	5348	6392	77.25%	83.67%	100.00%
5	6355	5718	5459	6355	89.98%	85.90%	100.00%
6	6712	5229	5889	6712	77.91%	87.74%	100.00%
7	6710	5237	5649	6710	78.05%	84.19%	100.00%
8	6215	4756	5526	6215	76.52%	88.91%	100.00%
9	6742	5133	5816	6742	76.13%	86.27%	100.00%
10	6614	5233	5713	6614	79.12%	86.38%	100.00%
11	6328	4558	5235	6328	72.03%	82.73%	100.00%
12	6224	4677	5031	6224	75.14%	80.83%	100.00%
Gemiddeld:	**6415.83**	**4950.67**	**5468.75**	**6415.83**	**77.13%**	**85.21%**	**100.00%**

Toetsgeval 2 - Tipe verandering 2 (Veranderlikes)							
Programmerings opdrag	**Totale aantal karakters**	**Opstelling 1 se aantal karakters**	**Opstelling 2 se aantal karakters**	**Opstelling 3 se aantal karakters**	**Opstelling 1 se ooreenkoms persentasie**	**Opstelling 2 se ooreenkoms persentasie**	**Opstelling 3 se ooreenkoms persentasie**
1	6112	4934	4077	4357	80.73%	66.70%	71.29%
2	6328	5312	4836	4767	83.94%	76.42%	75.33%
3	6258	5077	4875	4751	81.13%	77.90%	75.92%
4	6392	5371	4886	4886	84.03%	76.44%	76.44%
5	6355	5254	4925	4904	82.68%	77.50%	77.17%
6	6712	5502	5297	5297	81.97%	78.92%	78.92%
7	6710	5441	5210	5209	81.09%	77.65%	77.63%
8	6215	4956	4708	4707	79.74%	75.75%	75.74%
9	6742	5408	5238	5238	80.21%	77.69%	77.69%
10	6614	5513	5116	5114	83.35%	77.35%	77.32%
11	6328	5042	4661	4660	79.68%	73.66%	73.64%
12	6224	5259	4795	4776	84.50%	77.04%	76.74%
Gemiddeld:	**6415.83**	**5255.75**	**4885.33**	**4888.83**	**81.92%**	**76.09%**	**76.15%**

Toetsgeval 2 - Tipe verandering 3 (Verklarende stellings)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	6112	5514	5125	4829	90.22%	83.85%	79.01%
2	6328	5966	5426	5151	94.28%	85.75%	81.40%
3	6258	5977	5109	5312	95.51%	81.64%	84.88%
4	6392	6371	4900	5449	99.67%	76.66%	85.25%
5	6355	6308	4550	5323	99.26%	71.60%	83.76%
6	6712	6523	4726	5619	97.18%	70.41%	83.72%
7	6710	6646	5276	5813	99.05%	78.63%	86.63%
8	6215	6136	4379	5220	98.73%	70.46%	83.99%
9	6742	6338	5678	5777	94.01%	84.22%	85.69%
10	6614	6398	5281	5553	96.73%	79.85%	83.96%
11	6328	5453	4520	5093	86.17%	71.43%	80.48%
12	6224	5921	4372	5194	95.13%	70.24%	83.45%
Gemiddeld:	**6415.83**	**6129.25**	**4945.17**	**5361.08**	**95.50%**	**77.06%**	**83.52%**

Toetsgeval 2 - Tipe verandering 4 (Programmodules)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	6112	5931	5039	4714	97.04%	82.44%	77.13%
2	6328	6183	5389	5239	97.71%	85.16%	82.79%
3	6258	6238	5598	5716	99.68%	89.45%	91.34%
4	6392	6380	6015	5384	99.81%	94.10%	84.23%
5	6355	6299	5522	5242	99.12%	86.89%	82.49%
6	6712	6387	5690	5510	95.16%	84.77%	82.09%
7	6710	6288	5721	5524	93.71%	85.26%	82.32%
8	6215	5376	5376	4903	86.50%	86.50%	78.89%
9	6742	6662	6026	5772	98.81%	89.38%	85.61%
10	6614	6541	5876	5746	98.90%	88.84%	86.88%
11	6328	6219	5458	5234	98.28%	86.25%	82.71%
12	6224	5593	5608	5298	89.86%	90.10%	85.12%
Gemiddeld:	**6415.83**	**6174.75**	**5609.83**	**5356.83**	**96.21%**	**87.43%**	**83.47%**

Toetsgeval 2 - Tipe verandering 5 (Stellings)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	6112	4777	5073	4777	78.16%	83.00%	78.16%
2	6328	5295	5531	5295	83.68%	87.41%	83.68%
3	6258	5378	5691	5378	85.94%	90.94%	85.94%
4	6392	5533	6012	5533	86.56%	94.06%	86.56%
5	6355	5346	5419	5346	84.12%	85.27%	84.12%
6	6712	5662	5729	5662	84.36%	85.35%	84.36%
7	6710	5810	5873	5810	86.59%	87.53%	86.59%
8	6215	5217	5187	5217	83.94%	83.46%	83.94%
9	6742	5824	5992	5824	86.38%	88.88%	86.38%
10	6614	5712	5925	5712	86.36%	89.58%	86.36%
11	6328	5312	5312	5312	83.94%	83.94%	83.94%
12	6224	5411	5658	5411	86.94%	90.91%	86.94%
Gemiddeld:	**6415.83**	**5439.75**	**5616.83**	**5439.75**	**84.75%**	**87.53%**	**84.75%**

Toetsgeval 2 - Tipe verandering 6 (Besluit logika)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	6112	5814	5193	4856	95.12%	84.96%	79.45%
2	6328	6301	5611	5401	99.57%	88.67%	85.35%
3	6258	6003	5715	5665	95.93%	91.32%	90.52%
4	6392	6042	6141	5593	94.52%	96.07%	87.50%
5	6355	6192	5644	5060	97.44%	88.81%	79.62%
6	6712	6453	5840	5731	96.14%	87.01%	85.38%
7	6710	6617	5910	5880	98.61%	88.08%	87.63%
8	6215	6157	5607	5420	99.07%	90.22%	87.21%
9	6742	6672	6063	5849	98.96%	89.93%	86.75%
10	6614	5951	6032	5794	89.98%	91.20%	87.60%
11	6328	6218	5636	5414	98.26%	89.06%	85.56%
12	6224	5721	5717	5482	91.92%	91.85%	88.08%
Gemiddeld:	**6415.83**	**6178.42**	**5759.08**	**5512.08**	**96.29%**	**89.77%**	**85.89%**

Toetsgeval 2 - Tipe verandering 7 (Skuif van kontroles)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	6112	5995	4620	4061	98.09%	75.59%	66.44%
2	6328	6231	4697	3830	98.47%	74.23%	60.52%
3	6258	5990	4631	3734	95.72%	74.00%	59.67%
4	6392	6098	4333	4333	95.40%	67.79%	67.79%
5	6355	5864	4220	3691	92.27%	66.40%	58.08%
6	6712	5922	4171	3816	88.23%	62.14%	56.85%
7	6710	6524	4827	4054	97.23%	71.94%	60.42%
8	6215	5766	4322	3691	92.78%	69.54%	59.39%
9	6742	6241	4620	3946	92.57%	68.53%	58.53%
10	6614	6408	4820	4148	96.89%	72.88%	62.72%
11	6328	5824	4397	3741	92.04%	69.48%	59.12%
12	6224	6122	4678	4060	98.36%	75.16%	65.23%
Gemiddeld:	**6415.83**	**6082.08**	**4528.00**	**3925.42**	**94.84%**	**70.64%**	**61.23%**

Toetsgeval 2 - Tipe verandering 8 (Herbenaming van GGK kontroles)							
Programmerings opdrag	Totale aantal karakters	Opstelling 1 se aantal karakters	Opstelling 2 se aantal karakters	Opstelling 3 se aantal karakters	Opstelling 1 se ooreenkoms persentasie	Opstelling 2 se ooreenkoms persentasie	Opstelling 3 se ooreenkoms persentasie
1	6112	4664	3122	2801	76.31%	51.08%	45.83%
2	6328	3830	2804	3359	60.52%	44.31%	53.08%
3	6258	3544	2989	2668	56.63%	47.76%	42.63%
4	6392	3847	3292	2960	60.18%	51.50%	46.31%
5	6355	3352	2797	2476	52.75%	44.01%	38.96%
6	6712	3233	2678	2357	48.17%	39.90%	35.12%
7	6710	3400	2845	2466	50.67%	42.40%	36.75%
8	6215	3359	2804	2483	54.05%	45.12%	39.95%
9	6742	3345	2790	2469	49.61%	41.38%	36.62%
10	6614	3664	3108	2778	55.40%	46.99%	42.00%
11	6328	3331	2776	2455	52.64%	43.87%	38.80%
12	6224	3722	3167	2848	59.80%	50.88%	45.76%
Gemiddeld:	**6415.83**	**3607.58**	**2931.00**	**2676.67**	**56.39%**	**45.77%**	**41.82%**

Toetsgeval 2 - Tipe verandering 9 (Verandering van teks in GGK)							
Programmerings opdrag	**Totale aantal karakters**	**Verstelling 1 se aantal karakters**	**Verstelling 2 se aantal karakters**	**Verstelling 3 se aantal karakters**	**Verstelling 1 se ooreenkoms persentasie**	**Verstelling 2 se ooreenkoms persentasie**	**Verstelling 3 se ooreenkoms persentasie**
1	6112	5924	4584	4454	96.92%	75.00%	72.87%
2	6328	6126	4625	4447	96.81%	73.09%	70.27%
3	6258	6039	4517	4525	96.50%	72.18%	72.31%
4	6392	6122	4668	4857	95.78%	73.03%	75.99%
5	6355	5910	4292	4392	93.00%	67.54%	69.11%
6	6712	5730	4271	4205	85.37%	63.63%	62.65%
7	6710	6244	4485	4671	93.06%	66.84%	69.61%
8	6215	5910	4443	4302	95.09%	71.49%	69.22%
9	6742	6274	4665	4574	93.06%	69.19%	67.84%
10	6614	6514	4907	4656	98.49%	74.19%	70.40%
11	6328	6006	4539	4381	94.91%	71.73%	69.23%
12	6224	6367	4582	4649	102.30%	73.62%	74.69%
Gemiddeld:	**6415.83**	**6097.17**	**4548.17**	**4509.42**	**95.11%**	**70.96%**	**70.35%**

Toetsgeval 2 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)							
Programmerings opdrag	**Totale aantal karakters**	**Verstelling 1 se aantal karakters**	**Verstelling 2 se aantal karakters**	**Verstelling 3 se aantal karakters**	**Verstelling 1 se ooreenkoms persentasie**	**Verstelling 2 se ooreenkoms persentasie**	**Verstelling 3 se ooreenkoms persentasie**
1	6112	3601	3063	2635	58.92%	50.11%	43.11%
2	6328	3283	2745	2317	51.88%	43.38%	36.62%
3	6258	3468	2930	2502	55.42%	46.82%	39.98%
4	6392	4108	3420	2918	64.27%	53.50%	45.65%
5	6355	3352	2797	2476	52.75%	44.01%	38.96%
6	6712	3157	2619	2191	47.04%	39.02%	32.64%
7	6710	3324	2786	2350	49.54%	41.52%	35.02%
8	6215	3283	3283	2317	52.82%	52.82%	37.28%
9	6742	3269	2731	2303	48.49%	40.51%	34.16%
10	6614	3588	3049	2612	54.25%	46.10%	39.49%
11	6328	3318	2732	2289	52.43%	43.17%	36.17%
12	6224	3646	3108	2682	58.58%	49.94%	43.09%
Gemiddeld:	**6415.83**	**3449.75**	**2938.58**	**2466.00**	**53.86%**	**45.91%**	**38.52%**

Die resultate van *CodeMatch*® word as volg weergegee. Die ooreenkomspersentasie van die verklaring-, kommentaar/string- en instruksievolgorde algoritme van al twaalf programmeringsopdragte vir tipe veranderings 1 tot 10 word in die onderstaande tabelle weergegee. Toetsgeval 1 wat bestaan uit die vergelyking van die programkodelêers word eers weergegee, daarna toetsgeval 2 wat vergelyking van die programkodelêers en die ontwerpkodelêers bevat.

Toetsgeval 1 - Tipe verandering 1 (Kommentaar & indentering)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	100.00%	16.00%	100.00%	100.00%	87.00%
2	100.00%	18.00%	100.00%	100.00%	87.00%
3	100.00%	26.00%	100.00%	100.00%	88.00%
4	100.00%	27.00%	100.00%	100.00%	88.00%
5	100.00%	45.00%	100.00%	100.00%	89.00%
6	100.00%	31.00%	100.00%	100.00%	88.00%
7	100.00%	38.00%	100.00%	100.00%	89.00%
8	100.00%	66.00%	100.00%	100.00%	93.00%
9	100.00%	28.00%	100.00%	100.00%	88.00%
10	100.00%	48.00%	100.00%	100.00%	90.00%
11	100.00%	18.00%	100.00%	100.00%	87.00%
12	100.00%	20.00%	100.00%	100.00%	87.00%
Gemiddeld:	**100.00%**	**31.75%**	**100.00%**	**100.00%**	**88.42%**

Toetsgeval 1 - Tipe verandering 2 (Veranderlikes)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	46.00%	100.00%	90.00%	38.00%	74.00%
2	29.00%	100.00%	86.00%	43.00%	71.00%
3	46.00%	100.00%	92.00%	32.00%	72.00%
4	46.00%	100.00%	90.00%	30.00%	71.00%
5	49.00%	100.00%	94.00%	33.00%	74.00%
6	47.00%	100.00%	81.00%	33.00%	71.00%
7	47.00%	100.00%	84.00%	35.00%	70.00%
8	46.00%	100.00%	88.00%	34.00%	71.00%
9	45.00%	100.00%	91.00%	32.00%	72.00%
10	47.00%	100.00%	84.00%	30.00%	70.00%
11	24.00%	100.00%	86.00%	0.00%	67.00%
12	47.00%	100.00%	84.00%	27.00%	69.00%
Gemiddeld:	**43.25%**	**100.00%**	**87.50%**	**30.58%**	**71.00%**

Toetsgeval 1 - Tipe verandering 3 (Verklarende stellings					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	91.00%	100.00%	100.00%	32.00%	86.00%
2	76.00%	100.00%	100.00%	40.00%	83.00%
3	94.00%	100.00%	100.00%	32.00%	86.00%
4	100.00%	100.00%	100.00%	30.00%	88.00%
5	98.00%	100.00%	100.00%	30.00%	87.00%
6	100.00%	100.00%	100.00%	31.00%	88.00%
7	100.00%	100.00%	100.00%	30.00%	88.00%
8	92.00%	100.00%	100.00%	34.00%	86.00%
9	100.00%	100.00%	100.00%	32.00%	88.00%
10	81.00%	100.00%	100.00%	30.00%	83.00%
11	81.00%	100.00%	100.00%	35.00%	83.00%
12	91.00%	100.00%	100.00%	35.00%	86.00%
Gemiddeld:	**92.00%**	**100.00%**	**100.00%**	**32.58%**	**86.00%**

Toetsgeval 1 - Tipe verandering 4 (Programmodules)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	88.00%	100.00%	98.00%	43.00%	86.00%
2	95.00%	100.00%	100.00%	42.00%	88.00%
3	99.00%	100.00%	100.00%	73.00%	94.00%
4	96.00%	100.00%	100.00%	48.00%	89.00%
5	88.00%	100.00%	98.00%	46.00%	86.00%
6	85.00%	100.00%	99.00%	70.00%	89.00%
7	86.00%	100.00%	98.00%	71.00%	90.00%
8	70.00%	100.00%	98.00%	72.00%	86.00%
9	94.00%	100.00%	100.00%	73.00%	92.00%
10	94.00%	100.00%	100.00%	68.00%	91.00%
11	98.00%	100.00%	100.00%	42.00%	89.00%
12	97.00%	100.00%	100.00%	47.00%	89.00%
Gemiddeld:	**90.83%**	**100.00%**	**99.25%**	**57.92%**	**89.08%**

Toetsgeval 1 - Tipe verandering 5 (Stellings)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	96.00%	100.00%	100.00%	51.00%	89.00%
2	93.00%	100.00%	100.00%	46.00%	88.00%
3	96.00%	100.00%	100.00%	73.00%	93.00%
4	95.00%	100.00%	100.00%	48.00%	89.00%
5	95.00%	100.00%	100.00%	86.00%	95.00%
6	94.00%	100.00%	100.00%	71.00%	92.00%
7	93.00%	100.00%	100.00%	75.00%	92.00%
8	94.00%	100.00%	100.00%	73.00%	92.00%
9	96.00%	100.00%	100.00%	55.00%	90.00%
10	96.00%	100.00%	100.00%	51.00%	89.00%
11	94.00%	100.00%	100.00%	51.00%	89.00%
12	95.00%	100.00%	100.00%	51.00%	89.00%
Gemiddeld:	**94.75%**	**100.00%**	**100.00%**	**60.92%**	**90.58%**

Toetsgeval 1 - Tipe verandering 6 (Besluit logika)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	97.00%	100.00%	100.00%	74.00%	93.00%
2	94.00%	100.00%	100.00%	100.00%	99.00%
3	95.00%	100.00%	100.00%	60.00%	90.00%
4	94.00%	100.00%	98.00%	100.00%	98.00%
5	95.00%	100.00%	100.00%	100.00%	99.00%
6	96.00%	100.00%	100.00%	100.00%	99.00%
7	94.00%	100.00%	100.00%	100.00%	98.00%
8	97.00%	100.00%	100.00%	100.00%	99.00%
9	96.00%	100.00%	100.00%	100.00%	99.00%
10	95.00%	100.00%	97.00%	100.00%	98.00%
11	95.00%	100.00%	100.00%	100.00%	99.00%
12	98.00%	100.00%	100.00%	100.00%	99.00%
Gemiddeld:	**95.50%**	**100.00%**	**99.58%**	**94.50%**	**97.50%**

Toetsgeval 1 - Tipe verandering 7 (Skuif van kontroles)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Al vier algoritmes tesame**
1	86.00%	97.00%	100.00%	100.00%	96.00%
2	86.00%	100.00%	99.00%	100.00%	97.00%
3	80.00%	100.00%	99.00%	100.00%	95.00%
4	81.00%	100.00%	99.00%	100.00%	95.00%
5	80.00%	100.00%	99.00%	100.00%	94.00%
6	75.00%	100.00%	99.00%	100.00%	96.00%
7	84.00%	100.00%	99.00%	100.00%	95.00%
8	80.00%	100.00%	99.00%	100.00%	95.00%
9	80.00%	100.00%	99.00%	100.00%	95.00%
10	80.00%	100.00%	99.00%	100.00%	95.00%
11	80.00%	100.00%	99.00%	100.00%	95.00%
12	80.00%	100.00%	99.00%	100.00%	95.00%
Gemiddeld:	**81.00%**	**99.75%**	**99.08%**	**100.00%**	**95.25%**

Toetsgeval 1 - Tipe verandering 8 (Herbenaming van GGK kontroles)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Al vier algoritmes tesame**
1	27.00%	89.00%	94.00%	100.00%	83.00%
2	24.00%	85.00%	93.00%	100.00%	81.00%
3	25.00%	88.00%	93.00%	100.00%	82.00%
4	25.00%	87.00%	94.00%	100.00%	82.00%
5	25.00%	87.00%	93.00%	100.00%	82.00%
6	21.00%	87.00%	92.00%	100.00%	81.00%
7	23.00%	88.00%	92.00%	100.00%	82.00%
8	24.00%	83.00%	91.00%	100.00%	80.00%
9	22.00%	85.00%	93.00%	100.00%	81.00%
10	22.00%	85.00%	91.00%	100.00%	81.00%
11	25.00%	86.00%	95.00%	100.00%	82.00%
12	25.00%	87.00%	95.00%	100.00%	82.00%
Gemiddeld:	**24.00%**	**86.42%**	**93.00%**	**100.00%**	**81.58%**

Toetsgeval 1 - Tipe verandering 9 (Verandering van teks in GGK)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Al vier algoritmes tesame**
1	96.00%	93.00%	100.00%	90.00%	95.00%
2	96.00%	94.00%	100.00%	90.00%	95.00%
3	94.00%	93.00%	100.00%	90.00%	94.00%
4	95.00%	90.00%	100.00%	100.00%	96.00%
5	95.00%	90.00%	100.00%	100.00%	96.00%
6	86.00%	94.00%	100.00%	100.00%	95.00%
7	94.00%	89.00%	100.00%	90.00%	93.00%
8	95.00%	89.00%	100.00%	100.00%	96.00%
9	94.00%	95.00%	100.00%	100.00%	97.00%
10	94.00%	93.00%	100.00%	89.00%	94.00%
11	95.00%	94.00%	100.00%	90.00%	95.00%
12	95.00%	94.00%	100.00%	100.00%	97.00%
Gemiddeld:	**94.08%**	**92.33%**	**100.00%**	**94.92%**	**95.25%**

Toetsgeval 1 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	24.00%	86.00%	92.00%	100.00%	81.00%
2	23.00%	85.00%	93.00%	100.00%	81.00%
3	22.00%	83.00%	91.00%	100.00%	80.00%
4	34.00%	90.00%	94.00%	100.00%	84.00%
5	25.00%	87.00%	93.00%	100.00%	82.00%
6	19.00%	83.00%	90.00%	100.00%	80.00%
7	20.00%	83.00%	89.00%	100.00%	79.00%
8	22.00%	84.00%	90.00%	100.00%	80.00%
9	20.00%	85.00%	92.00%	100.00%	81.00%
10	20.00%	81.00%	89.00%	100.00%	79.00%
11	30.00%	87.00%	94.00%	100.00%	83.00%
12	22.00%	82.00%	92.00%	100.00%	80.00%
Gemiddeld:	**23.42%**	**84.67%**	**91.58%**	**100.00%**	**80.83%**

Toetsgeval 2 - Tipe verandering 1 (Kommentaar & indentering)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	100.00%	58.00%	100.00%	100.00%	93.50%
2	100.00%	59.00%	100.00%	100.00%	93.50%
3	100.00%	63.00%	100.00%	100.00%	94.00%
4	100.00%	63.50%	100.00%	100.00%	94.00%
5	100.00%	72.50%	100.00%	100.00%	94.50%
6	100.00%	65.50%	100.00%	100.00%	94.00%
7	100.00%	69.00%	100.00%	100.00%	94.50%
8	100.00%	83.00%	100.00%	100.00%	96.50%
9	100.00%	64.00%	100.00%	100.00%	94.00%
10	100.00%	74.00%	100.00%	100.00%	95.00%
11	100.00%	59.00%	100.00%	100.00%	93.50%
12	100.00%	60.00%	100.00%	100.00%	93.50%
Gemiddeld:	**100.00%**	**65.88%**	**100.00%**	**100.00%**	**94.21%**

Toetsgeval 2 - Tipe verandering 2 (Veranderlikes)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	73.00%	100.00%	95.00%	69.00%	87.00%
2	59.00%	98.50%	92.50%	71.50%	85.50%
3	67.00%	98.50%	95.50%	66.00%	86.00%
4	73.00%	100.00%	95.00%	65.00%	85.50%
5	74.50%	100.00%	97.00%	66.50%	87.00%
6	54.00%	97.00%	89.00%	66.50%	85.50%
7	52.50%	95.00%	89.00%	67.50%	85.00%
8	56.00%	96.50%	92.00%	67.00%	85.50%
9	57.00%	96.50%	94.00%	66.00%	86.00%
10	54.00%	96.50%	90.00%	65.00%	85.00%
11	46.50%	97.00%	91.50%	50.00%	83.50%
12	56.50%	96.50%	91.00%	63.50%	84.50%
Gemiddeld:	**60.25%**	**97.67%**	**92.63%**	**65.29%**	**85.50%**

Toetsgeval 2 - Tipe verandering 3 (Verklarende stellings)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	95.50%	100.00%	100.00%	61.00%	93.00%
2	88.00%	100.00%	100.00%	65.00%	91.50%
3	96.00%	100.00%	100.00%	61.00%	93.00%
4	100.00%	100.00%	100.00%	60.00%	94.00%
5	99.00%	100.00%	100.00%	65.00%	93.50%
6	96.00%	100.00%	100.00%	61.00%	94.00%
7	100.00%	100.00%	100.00%	60.00%	94.00%
8	96.00%	100.00%	100.00%	62.00%	93.00%
9	100.00%	100.00%	100.00%	61.00%	94.00%
10	90.50%	100.00%	100.00%	59.50%	91.50%
11	90.50%	100.00%	100.00%	62.50%	91.50%
12	95.50%	100.00%	100.00%	62.50%	93.00%
Gemiddeld:	**95.58%**	**100.00%**	**100.00%**	**61.71%**	**93.00%**

Toetsgeval 2 - Tipe verandering 4 (Programmodules)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	94.00%	100.00%	99.00%	71.50%	93.00%
2	97.50%	100.00%	100.00%	71.00%	94.00%
3	99.50%	100.00%	100.00%	86.50%	97.00%
4	98.00%	100.00%	100.00%	74.00%	94.50%
5	94.00%	100.00%	99.00%	73.00%	93.00%
6	92.50%	100.00%	99.50%	85.00%	94.50%
7	93.00%	100.00%	99.00%	85.50%	95.00%
8	85.00%	100.00%	99.00%	86.00%	93.00%
9	97.00%	100.00%	100.00%	86.50%	96.00%
10	97.00%	100.00%	100.00%	84.00%	95.50%
11	99.00%	100.00%	100.00%	71.00%	94.50%
12	98.50%	100.00%	100.00%	73.50%	94.50%
Gemiddeld:	**95.42%**	**100.00%**	**99.63%**	**78.96%**	**94.54%**

Toetsgeval 2 - Tipe verandering 5 (Stellings)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	98.00%	100.00%	100.00%	75.50%	94.50%
2	96.50%	100.00%	100.00%	73.00%	94.00%
3	98.00%	100.00%	100.00%	86.50%	96.50%
4	97.50%	100.00%	100.00%	74.00%	94.50%
5	97.50%	100.00%	100.00%	93.00%	97.50%
6	97.00%	100.00%	100.00%	85.50%	96.00%
7	96.50%	100.00%	100.00%	87.50%	96.00%
8	97.00%	100.00%	100.00%	86.50%	96.00%
9	98.00%	100.00%	100.00%	77.50%	95.00%
10	98.00%	100.00%	100.00%	75.50%	94.50%
11	97.00%	100.00%	100.00%	75.50%	94.50%
12	97.50%	100.00%	100.00%	75.50%	94.50%
Gemiddeld:	**97.38%**	**100.00%**	**100.00%**	**80.46%**	**95.29%**

Toetsgeval 2 - Tipe verandering 6 (Besluit logika)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Al vier algoritmes tesame**
1	98.50%	100.00%	100.00%	87.00%	96.50%
2	97.00%	100.00%	100.00%	100.00%	99.50%
3	97.50%	100.00%	100.00%	80.00%	95.00%
4	97.00%	100.00%	99.00%	100.00%	99.00%
5	97.50%	100.00%	100.00%	100.00%	99.50%
6	98.00%	100.00%	100.00%	100.00%	99.50%
7	97.00%	100.00%	100.00%	100.00%	99.00%
8	98.50%	100.00%	100.00%	100.00%	99.50%
9	98.00%	100.00%	100.00%	100.00%	99.50%
10	97.50%	100.00%	98.50%	100.00%	99.00%
11	97.50%	100.00%	100.00%	100.00%	99.50%
12	99.00%	100.00%	100.00%	100.00%	99.50%
Gemiddeld:	**97.75%**	**100.00%**	**99.79%**	**97.25%**	**98.75%**

Toetsgeval 2 - Tipe verandering 7 (Skuif van kontroles)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Al vier algoritmes tesame**
1	93.00%	98.50%	100.00%	100.00%	98.00%
2	93.00%	100.00%	99.50%	100.00%	98.50%
3	90.00%	100.00%	99.50%	100.00%	97.50%
4	90.50%	100.00%	99.50%	100.00%	97.50%
5	90.00%	100.00%	99.50%	100.00%	97.00%
6	87.50%	100.00%	99.50%	100.00%	98.00%
7	92.00%	100.00%	99.50%	100.00%	97.50%
8	90.00%	100.00%	99.50%	100.00%	97.50%
9	90.00%	100.00%	99.50%	100.00%	97.50%
10	90.00%	100.00%	99.50%	100.00%	97.50%
11	90.00%	100.00%	99.50%	100.00%	97.50%
12	90.00%	100.00%	99.50%	100.00%	97.50%
Gemiddeld:	**90.50%**	**99.88%**	**99.54%**	**100.00%**	**97.63%**

Toetsgeval 2 - Tipe verandering 8 (Herbenaming van GGK kontroles)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Al vier algoritmes tesame**
1	60.00%	94.50%	95.00%	100.00%	91.50%
2	55.00%	92.50%	93.00%	81.50%	90.50%
3	58.50%	94.00%	94.00%	82.00%	91.00%
4	59.00%	93.50%	94.50%	82.50%	91.00%
5	54.50%	93.50%	93.50%	81.50%	91.00%
6	54.00%	93.50%	91.50%	81.50%	90.50%
7	53.00%	94.00%	92.00%	75.00%	91.00%
8	47.50%	91.50%	92.00%	74.50%	90.00%
9	54.50%	92.50%	93.00%	76.00%	90.50%
10	56.50%	92.50%	92.50%	77.50%	90.50%
11	56.50%	93.00%	95.00%	82.50%	91.00%
12	58.00%	93.50%	95.50%	92.00%	91.00%
Gemiddeld:	**55.58%**	**93.21%**	**93.46%**	**82.21%**	**90.79%**

Toetsgeval 2 - Tipe verandering 9 (Verandering van teks in GGK)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	98.00%	96.50%	100.00%	95.00%	97.50%
2	98.00%	97.00%	100.00%	95.00%	97.50%
3	97.00%	96.50%	100.00%	95.00%	97.00%
4	97.50%	95.00%	100.00%	100.00%	98.00%
5	97.50%	95.00%	100.00%	100.00%	98.00%
6	93.00%	97.00%	100.00%	100.00%	97.50%
7	97.00%	94.50%	100.00%	95.00%	96.50%
8	97.50%	94.50%	100.00%	100.00%	98.00%
9	97.00%	97.50%	100.00%	100.00%	98.50%
10	97.00%	96.50%	100.00%	94.50%	97.00%
11	97.50%	97.00%	100.00%	95.00%	97.50%
12	97.50%	97.00%	100.00%	100.00%	98.50%
Gemiddeld:	**97.04%**	**96.17%**	**100.00%**	**97.46%**	**97.63%**

Toetsgeval 2 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)					
Programmerings opdrag	**Verklaring algoritme**	**Kommentaar/ String algoritme**	**Identifiseerder algoritme**	**Instruksie volgorde algoritme**	**Vier algoritmes gelyktydig**
1	58.50%	93.00%	93.50%	100.00%	90.50%
2	54.50%	92.50%	93.50%	81.50%	90.50%
3	56.50%	91.50%	92.00%	82.00%	90.00%
4	63.50%	95.00%	94.50%	82.50%	92.00%
5	54.00%	93.50%	93.50%	81.50%	91.00%
6	53.00%	91.50%	90.50%	81.00%	90.00%
7	51.50%	91.50%	90.00%	75.00%	89.50%
8	46.50%	92.00%	91.50%	74.50%	90.00%
9	53.50%	92.50%	93.00%	76.00%	90.50%
10	55.00%	90.50%	90.00%	77.50%	89.50%
11	60.50%	93.50%	95.00%	82.50%	91.50%
12	56.50%	91.00%	94.00%	92.00%	90.00%
Gemiddeld:	**55.29%**	**92.33%**	**92.58%**	**82.17%**	**90.42%**

Die resultate van CPD word as volg weergegee. Die aantal lyne wat ooreenstem met die drumpelwaardes 3, 6, 9 en 12 asook die ooreenkomspersentasies van die drumpelwaardes van al twaalf programmeringsopdragte vir tipe verandering 1 tot 10 word in die onderstaande tabelle weergegee. Toetsgeval 1 wat bestaan uit die vergelyking van die programkodelêers word eers weergegee, daarna toetsgeval 2 wat die vergelyking van die programkodelêers en die ontwerpkodelêers bevat.

Toetsgeval 1 - Tipe verandering 1 (Kommentaar & indentering)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	90	90	90	90	90	100.00%	100.00%	100.00%	100.00%
2	74	73	73	73	73	98.65%	98.65%	98.65%	98.65%
3	82	80	80	80	80	97.56%	97.56%	97.56%	97.56%
4	85	85	85	85	85	100.00%	100.00%	100.00%	100.00%
5	71	71	71	71	71	100.00%	100.00%	100.00%	100.00%
6	76	76	76	76	76	100.00%	100.00%	100.00%	100.00%
7	71	71	71	71	71	100.00%	100.00%	100.00%	100.00%
8	70	70	70	70	70	100.00%	100.00%	100.00%	100.00%
9	73	73	73	73	73	100.00%	100.00%	100.00%	100.00%
10	87	87	87	87	87	100.00%	100.00%	100.00%	100.00%
11	70	70	70	70	70	100.00%	100.00%	100.00%	100.00%
12	93	93	93	93	93	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**78.50**	**78.25**	**78.25**	**78.25**	**78.25**	**99.68%**	**99.68%**	**99.68%**	**99.68%**

Toetsgeval 1 - Tipe verandering 2 (Veranderlikes)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	90	103	67	39	38	114.44%	74.44%	43.33%	42.22%
2	73	89	60	47	45	121.92%	82.19%	64.38%	61.64%
3	80	83	54	42	35	103.75%	67.50%	52.50%	43.75%
4	85	89	53	37	36	104.71%	62.35%	43.53%	42.35%
5	71	69	53	48	34	97.18%	74.65%	67.61%	47.89%
6	76	82	43	26	26	107.89%	56.58%	34.21%	34.21%
7	71	83	40	38	26	116.90%	56.34%	53.52%	36.62%
8	70	76	44	34	26	108.57%	62.86%	48.57%	37.14%
9	73	86	40	35	27	117.81%	54.79%	47.95%	36.99%
10	87	95	52	31	25	109.20%	59.77%	35.63%	28.74%
11	70	93	37	23	17	132.86%	52.86%	32.86%	24.29%
12	93	75	48	45	33	80.65%	51.61%	48.39%	35.48%
Gemiddeld:	**78.25**	**85.25**	**49.25**	**37.08**	**30.67**	**109.66%**	**63.00%**	**47.71%**	**39.28%**

Toetsgeval 1 - Tipe verandering 3 (Verklarende stellings)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	91	100	83	80	80	109.89%	91.21%	87.91%	87.91%
2	77	89	77	76	74	115.58%	100.00%	98.70%	96.10%
3	80	87	80	80	80	108.75%	100.00%	100.00%	100.00%
4	85	79	79	77	71	92.94%	92.94%	90.59%	83.53%
5	76	71	68	63	54	93.42%	89.47%	82.89%	71.05%
6	80	89	79	74	61	111.25%	98.75%	92.50%	76.25%
7	71	73	67	67	59	102.82%	94.37%	94.37%	83.10%
8	71	78	62	58	55	109.86%	87.32%	81.69%	77.46%
9	79	84	73	66	61	106.33%	92.41%	83.54%	77.22%
10	93	94	83	81	74	101.08%	89.25%	87.10%	79.57%
11	74	99	67	65	63	133.78%	90.54%	87.84%	85.14%
12	95	100	89	85	80	105.26%	93.68%	89.47%	84.21%
Gemiddeld:	**81.00**	**86.92**	**75.58**	**72.67**	**67.67**	**107.58%**	**93.33%**	**89.72%**	**83.46%**

Toetsgeval 1 - Tipe verandering 4 (Programmodules)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	85	83	82	82	82	97.65%	96.47%	96.47%	96.47%
2	73	76	69	66	64	104.11%	94.52%	90.41%	87.67%
3	74	69	72	72	69	93.24%	97.30%	97.30%	93.24%
4	84	78	77	74	74	92.86%	91.67%	88.10%	88.10%
5	69	70	69	65	63	101.45%	100.00%	94.20%	91.30%
6	74	70	68	67	67	94.59%	91.89%	90.54%	90.54%
7	67	59	58	58	58	88.06%	86.57%	86.57%	86.57%
8	67	64	58	57	57	95.52%	86.57%	85.07%	85.07%
9	75	61	66	66	66	81.33%	88.00%	88.00%	88.00%
10	91	77	76	75	72	84.62%	83.52%	82.42%	79.12%
11	66	76	63	62	62	115.15%	95.45%	93.94%	93.94%
12	88	84	84	84	79	95.45%	95.45%	95.45%	89.77%
Gemiddeld:	**76.08**	**72.25**	**70.17**	**69.00**	**67.75**	**95.34%**	**92.28%**	**90.71%**	**89.15%**

Toetsgeval 1 - Tipe verandering 5 (Stellings)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	95	95	71	67	66	100.00%	74.74%	70.53%	69.47%
2	78	80	64	57	57	102.56%	82.05%	73.08%	73.08%
3	84	82	64	64	64	97.62%	76.19%	76.19%	76.19%
4	89	79	74	74	74	88.76%	83.15%	83.15%	83.15%
5	75	70	69	70	68	93.33%	92.00%	93.33%	90.67%
6	80	76	67	61	57	95.00%	83.75%	76.25%	71.25%
7	75	71	68	67	67	94.67%	90.67%	89.33%	89.33%
8	77	71	70	69	69	92.21%	90.91%	89.61%	89.61%
9	77	72	63	63	59	93.51%	81.82%	81.82%	76.62%
10	91	89	85	83	83	97.80%	93.41%	91.21%	91.21%
11	74	80	63	54	54	108.11%	85.14%	72.97%	72.97%
12	97	93	76	70	70	95.88%	78.35%	72.16%	72.16%
Gemiddeld:	**82.67**	**79.83**	**69.50**	**66.58**	**65.67**	**96.62%**	**84.35%**	**80.80%**	**79.64%**

Toetsgeval 1 - Tipe verandering 6 (Besluit logika)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	91	82	80	75	75	90.11%	87.91%	82.42%	82.42%
2	73	73	70	70	70	100.00%	95.89%	95.89%	95.89%
3	80	69	66	66	66	86.25%	82.50%	82.50%	82.50%
4	85	78	76	76	76	91.76%	89.41%	89.41%	89.41%
5	71	64	61	61	61	90.14%	85.92%	85.92%	85.92%
6	76	69	67	67	67	90.79%	88.16%	88.16%	88.16%
7	71	64	62	62	62	90.14%	87.32%	87.32%	87.32%
8	70	63	60	60	60	90.00%	85.71%	85.71%	85.71%
9	73	66	63	63	63	90.41%	86.30%	86.30%	86.30%
10	87	85	77	77	77	97.70%	88.51%	88.51%	88.51%
11	70	70	67	67	67	100.00%	95.71%	95.71%	95.71%
12	93	86	83	83	83	92.47%	89.25%	89.25%	89.25%
Gemiddeld:	**78.33**	**72.42**	**69.33**	**68.92**	**68.92**	**92.48%**	**88.55%**	**88.09%**	**88.09%**

Toetsgeval 1 - Tipe verandering 7 (Skuif van kontroles)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	106	106	106	106	106	100.00%	100.00%	100.00%	100.00%
2	116	116	116	116	116	100.00%	100.00%	100.00%	100.00%
3	117	117	117	116	116	100.00%	100.00%	99.15%	99.15%
4	119	119	119	119	119	100.00%	100.00%	100.00%	100.00%
5	119	119	119	119	119	100.00%	100.00%	100.00%	100.00%
6	124	124	124	124	124	100.00%	100.00%	100.00%	100.00%
7	129	129	129	129	129	100.00%	100.00%	100.00%	100.00%
8	117	117	117	117	117	100.00%	100.00%	100.00%	100.00%
9	129	129	129	129	129	100.00%	100.00%	100.00%	100.00%
10	127	127	127	127	127	100.00%	100.00%	100.00%	100.00%
11	116	116	116	116	116	100.00%	100.00%	100.00%	100.00%
12	117	117	117	117	117	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**119.67**	**119.67**	**119.67**	**119.58**	**119.58**	**100.00%**	**100.00%**	**99.93%**	**99.93%**

Toetsgeval 1 - Tipe verandering 8 (Herbenaming van GGK kontroles)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	106	112	98	90	90	105.66%	92.45%	84.91%	84.91%
2	116	109	90	83	83	93.97%	77.59%	71.55%	71.55%
3	117	115	91	83	83	98.29%	77.78%	70.94%	70.94%
4	119	119	94	83	83	100.00%	78.99%	69.75%	69.75%
5	119	124	107	82	82	104.20%	89.92%	68.91%	68.91%
6	124	121	103	89	89	97.58%	83.06%	71.77%	71.77%
7	129	123	104	83	83	95.35%	80.62%	64.34%	64.34%
8	117	105	96	91	91	89.74%	82.05%	77.78%	77.78%
9	129	118	98	89	89	91.47%	75.97%	68.99%	68.99%
10	127	115	95	80	80	90.55%	74.80%	62.99%	62.99%
11	116	114	96	83	83	98.28%	82.76%	71.55%	71.55%
12	117	124	107	88	88	105.98%	91.45%	75.21%	75.21%
Gemiddeld:	**119.67**	**116.58**	**98.25**	**85.33**	**85.33**	**97.59%**	**82.29%**	**71.56%**	**71.56%**

Toetsgeval 1 - Tipe verandering 9 (Verandering van teks in GGK)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	108	101	99	99	99	93.52%	91.67%	91.67%	91.67%
2	118	108	106	106	106	91.53%	89.83%	89.83%	89.83%
3	119	107	104	104	104	89.92%	87.39%	87.39%	87.39%
4	119	107	104	104	104	89.92%	87.39%	87.39%	87.39%
5	119	107	104	104	104	89.92%	87.39%	87.39%	87.39%
6	124	123	116	116	116	99.19%	93.55%	93.55%	93.55%
7	131	119	114	114	114	90.84%	87.02%	87.02%	87.02%
8	117	111	109	109	109	94.87%	93.16%	93.16%	93.16%
9	129	121	117	117	117	93.80%	90.70%	90.70%	90.70%
10	131	117	113	113	113	89.31%	86.26%	86.26%	86.26%
11	117	111	109	109	109	94.87%	93.16%	93.16%	93.16%
12	117	111	109	109	109	94.87%	93.16%	93.16%	93.16%
Gemiddeld:	**120.75**	**111.92**	**108.67**	**108.67**	**108.67**	**92.71%**	**90.06%**	**90.06%**	**90.06%**

Toetsgeval 1 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	106	99	85	77	77	93.40%	80.19%	72.64%	72.64%
2	116	99	81	68	68	85.34%	69.83%	58.62%	58.62%
3	117	105	94	73	73	89.74%	80.34%	62.39%	62.39%
4	119	114	95	90	90	95.80%	79.83%	75.63%	75.63%
5	119	110	99	68	68	92.44%	83.19%	57.14%	57.14%
6	124	110	97	78	78	88.71%	78.23%	62.90%	62.90%
7	129	108	97	68	68	83.72%	75.19%	52.71%	52.71%
8	117	108	94	76	76	92.31%	80.34%	64.96%	64.96%
9	129	110	98	75	75	85.27%	75.97%	58.14%	58.14%
10	127	105	95	70	70	82.68%	74.80%	55.12%	55.12%
11	116	113	98	90	90	97.41%	84.48%	77.59%	77.59%
12	117	107	95	69	69	91.45%	81.20%	58.97%	58.97%
Gemiddeld:	**119.67**	**107.33**	**94.00**	**75.17**	**75.17**	**89.86%**	**78.63%**	**63.07%**	**63.07%**

Toetsgeval 2 - Tipe verandering 1 (Kommentaar & indentering)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	196	186	186	186	186	94.90%	94.90%	94.90%	94.90%
2	190	186	186	186	186	97.89%	97.89%	97.89%	97.89%
3	199	187	187	187	187	93.97%	93.97%	93.97%	93.97%
4	204	194	194	194	194	95.10%	95.10%	95.10%	95.10%
5	190	180	180	180	180	94.74%	94.74%	94.74%	94.74%
6	200	190	190	190	190	95.00%	95.00%	95.00%	95.00%
7	200	190	190	190	190	95.00%	95.00%	95.00%	95.00%
8	187	177	177	177	177	94.65%	94.65%	94.65%	94.65%
9	202	192	192	192	192	95.05%	95.05%	95.05%	95.05%
10	214	204	204	204	204	95.33%	95.33%	95.33%	95.33%
11	186	183	183	183	183	98.39%	98.39%	98.39%	98.39%
12	210	200	200	200	200	95.24%	95.24%	95.24%	95.24%
Gemiddeld:	**198.17**	**189.08**	**189.08**	**189.08**	**189.08**	**95.44%**	**95.44%**	**95.44%**	**95.44%**

Toetsgeval 2 - Tipe verandering 2 (Veranderlikes)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	196	200	163	135	129	102.04%	83.16%	68.88%	65.82%
2	190	215	173	160	158	113.16%	91.05%	84.21%	83.16%
3	199	195	161	149	142	97.99%	80.90%	74.87%	71.36%
4	204	198	163	146	145	97.06%	79.90%	71.57%	71.08%
5	190	178	162	157	143	93.68%	85.26%	82.63%	75.26%
6	200	196	159	140	140	98.00%	79.50%	70.00%	70.00%
7	200	161	161	157	145	80.50%	80.50%	78.50%	72.50%
8	187	203	161	141	133	108.56%	86.10%	75.40%	71.12%
9	202	206	159	154	146	101.98%	78.71%	76.24%	72.28%
10	214	211	164	148	142	98.60%	76.64%	69.16%	66.36%
11	186	204	150	136	130	109.68%	80.65%	73.12%	69.89%
12	210	191	155	152	140	90.95%	73.81%	72.38%	66.67%
Gemiddeld:	**198.17**	**196.50**	**160.92**	**147.92**	**141.08**	**99.35%**	**81.35%**	**74.75%**	**71.29%**

Toetsgeval 2 - Tipe verandering 3 (Verklarende stellings)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	196	206	199	193	192	105.10%	101.53%	98.47%	97.96%
2	190	220	208	193	187	115.79%	109.47%	101.58%	98.42%
3	199	204	194	194	190	102.51%	97.49%	97.49%	95.48%
4	204	198	188	188	180	97.06%	92.16%	92.16%	88.24%
5	190	189	180	172	163	99.47%	94.74%	90.53%	85.79%
6	200	213	200	198	183	106.50%	100.00%	99.00%	91.50%
7	200	202	189	189	178	101.00%	94.50%	94.50%	89.00%
8	187	195	179	172	162	104.28%	95.72%	91.98%	86.63%
9	202	213	195	194	186	105.45%	96.53%	96.04%	92.08%
10	214	221	209	202	195	103.27%	97.66%	94.39%	91.12%
11	186	210	189	187	184	112.90%	101.61%	100.54%	98.92%
12	210	217	203	197	187	103.33%	96.67%	93.81%	89.05%
Gemiddeld:	**198.17**	**207.33**	**194.42**	**189.92**	**182.25**	**104.72%**	**98.17%**	**95.87%**	**92.02%**

Toetsgeval 2 - Tipe verandering 4 (Programmodules)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	191	183	183	183	182	95.81%	95.81%	95.81%	95.29%
2	189	189	187	186	184	100.00%	98.94%	98.41%	97.35%
3	191	186	185	185	181	97.38%	96.86%	96.86%	94.76%
4	203	196	190	189	189	96.55%	93.60%	93.10%	93.10%
5	188	188	188	181	176	100.00%	100.00%	96.28%	93.62%
6	198	190	183	182	182	95.96%	92.42%	91.92%	91.92%
7	196	187	180	179	179	95.41%	91.84%	91.33%	91.33%
8	184	181	174	169	169	98.37%	94.57%	91.85%	91.85%
9	204	190	185	185	185	93.14%	90.69%	90.69%	90.69%
10	218	203	203	203	200	93.12%	93.12%	93.12%	91.74%
11	182	179	179	179	179	98.35%	98.35%	98.35%	98.35%
12	205	197	197	196	191	96.10%	96.10%	95.61%	93.17%
Gemiddeld:	**195.75**	**189.08**	**186.17**	**184.75**	**183.08**	**96.68%**	**95.19%**	**94.44%**	**93.60%**

Toetsgeval 2 - Tipe verandering 5 (Stellings)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	201	201	184	163	162	100.00%	91.54%	81.09%	80.60%
2	194	194	180	170	170	100.00%	92.78%	87.63%	87.63%
3	201	199	181	171	171	99.00%	90.05%	85.07%	85.07%
4	208	200	183	183	183	96.15%	87.98%	87.98%	87.98%
5	194	189	186	186	177	97.42%	95.88%	95.88%	91.24%
6	204	195	183	175	171	95.59%	89.71%	85.78%	83.82%
7	204	200	195	187	187	98.04%	95.59%	91.67%	91.67%
8	194	186	183	183	183	95.88%	94.33%	94.33%	94.33%
9	206	201	182	182	178	97.57%	88.35%	88.35%	86.41%
10	218	216	205	205	205	99.08%	94.04%	94.04%	94.04%
11	190	190	180	167	167	100.00%	94.74%	87.89%	87.89%
12	214	210	193	187	187	98.13%	90.19%	87.38%	87.38%
Gemiddeld:	**202.33**	**198.42**	**186.25**	**179.92**	**178.42**	**98.07%**	**92.10%**	**88.93%**	**88.17%**

Toetsgeval 2 - Tipe verandering 6 (Besluit logika)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	197	188	187	181	181	95.43%	94.92%	91.88%	91.88%
2	189	189	189	189	189	100.00%	100.00%	100.00%	100.00%
3	197	186	185	185	185	94.42%	93.91%	93.91%	93.91%
4	204	197	197	197	197	96.57%	96.57%	96.57%	96.57%
5	190	183	183	183	183	96.32%	96.32%	96.32%	96.32%
6	200	193	193	193	193	96.50%	96.50%	96.50%	96.50%
7	200	193	193	193	193	96.50%	96.50%	96.50%	96.50%
8	187	180	180	180	180	96.26%	96.26%	96.26%	96.26%
9	202	195	195	195	195	96.53%	96.53%	96.53%	96.53%
10	214	207	207	207	207	96.73%	96.73%	96.73%	96.73%
11	186	186	186	186	186	100.00%	100.00%	100.00%	100.00%
12	210	203	203	203	203	96.67%	96.67%	96.67%	96.67%
Gemiddeld:	**198.00**	**191.67**	**191.50**	**191.00**	**191.00**	**96.83%**	**96.74%**	**96.49%**	**96.49%**

Toetsgeval 2 - Tipe verandering 7 (Skuif van kontroles)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	197	192	192	192	192	97.46%	97.46%	97.46%	97.46%
2	189	184	184	184	184	97.35%	97.35%	97.35%	97.35%
3	197	197	197	197	197	100.00%	100.00%	100.00%	100.00%
4	204	204	204	204	204	100.00%	100.00%	100.00%	100.00%
5	190	190	190	190	190	100.00%	100.00%	100.00%	100.00%
6	200	200	200	200	200	100.00%	100.00%	100.00%	100.00%
7	200	199	199	199	199	99.50%	99.50%	99.50%	99.50%
8	188	188	187	187	187	100.00%	99.47%	99.47%	99.47%
9	202	202	193	193	193	100.00%	95.54%	95.54%	95.54%
10	214	219	214	214	214	102.34%	100.00%	100.00%	100.00%
11	186	186	186	186	186	100.00%	100.00%	100.00%	100.00%
12	210	210	210	210	210	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**198.08**	**197.58**	**196.33**	**196.33**	**196.33**	**99.72%**	**99.11%**	**99.11%**	**99.11%**

Toetsgeval 2 - Tipe verandering 8 (Herbenaming van GGK kontroles)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	196	194	194	172	170	98.98%	98.98%	87.76%	86.73%
2	190	184	172	155	153	96.84%	90.53%	81.58%	80.53%
3	199	187	173	155	153	93.97%	86.93%	77.89%	76.88%
4	204	192	178	158	156	94.12%	87.25%	77.45%	76.47%
5	190	189	187	142	140	99.47%	98.42%	74.74%	73.68%
6	200	190	183	156	154	95.00%	91.50%	78.00%	77.00%
7	200	186	177	146	144	93.00%	88.50%	73.00%	72.00%
8	187	185	185	153	151	98.93%	98.93%	81.82%	80.75%
9	202	188	188	153	150	93.07%	93.07%	75.74%	74.26%
10	214	201	180	158	156	93.93%	84.11%	73.83%	72.90%
11	186	183	169	152	150	98.39%	90.86%	81.72%	80.65%
12	210	210	208	170	167	100.00%	99.05%	80.95%	79.52%
Gemiddeld:	**198.17**	**190.75**	**182.83**	**155.83**	**153.67**	**96.31%**	**92.34%**	**78.71%**	**77.61%**

Toetsgeval 2 - Tipe verandering 9 (Verandering van teks in GGK)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	198	181	181	181	181	91.41%	91.41%	91.41%	91.41%
2	192	178	178	178	178	92.71%	92.71%	92.71%	92.71%
3	201	177	177	177	177	88.06%	88.06%	88.06%	88.06%
4	204	182	182	182	182	89.22%	89.22%	89.22%	89.22%
5	190	168	168	168	168	88.42%	88.42%	88.42%	88.42%
6	200	189	189	189	186	94.50%	94.50%	94.50%	93.00%
7	202	180	180	180	180	89.11%	89.11%	89.11%	89.11%
8	187	171	171	171	171	91.44%	91.44%	91.44%	91.44%
9	202	184	184	184	184	91.09%	91.09%	91.09%	91.09%
10	218	194	194	194	191	88.99%	88.99%	88.99%	87.61%
11	187	178	178	178	175	95.19%	95.19%	95.19%	93.58%
12	210	194	194	194	194	92.38%	92.38%	92.38%	92.38%
Gemiddeld:	**199.25**	**181.33**	**181.33**	**181.33**	**180.58**	**91.04%**	**91.04%**	**91.04%**	**90.67%**

Toetsgeval 2 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	196	181	181	159	157	92.35%	92.35%	81.12%	80.10%
2	190	171	157	140	138	90.00%	82.63%	73.68%	72.63%
3	199	177	177	145	143	88.94%	88.94%	72.86%	71.86%
4	204	192	173	165	163	94.12%	84.80%	80.88%	79.90%
5	190	177	173	131	126	93.16%	91.05%	68.95%	66.32%
6	200	177	177	143	143	88.50%	88.50%	71.50%	71.50%
7	200	171	171	131	129	85.50%	85.50%	65.50%	64.50%
8	187	172	170	138	136	91.98%	90.91%	73.80%	72.73%
9	202	174	174	138	135	86.14%	86.14%	68.32%	66.83%
10	214	187	184	148	146	87.38%	85.98%	69.16%	68.22%
11	186	181	170	158	152	97.31%	91.40%	84.95%	81.72%
12	210	191	191	153	150	90.95%	90.95%	72.86%	71.43%
Gemiddeld:	**198.17**	**179.25**	**174.83**	**145.75**	**143.17**	**90.53%**	**88.26%**	**73.63%**	**72.31%**

Die resultate van JPlag word as volg weergegee. Die aantal lyne wat ooreenstem met die drumpelwaardes 3, 6, 9 en 12 asook die ooreenkomspersentasies van die drumpelwaardes van al twaalf programmeringsopdragte vir tipe verandering 1 tot 10 word in die onderstaande tabelle weergegee. Toetsgeval 1 wat bestaan uit die vergelyking van die programkodelêers word eers weergegee, daarna toetsgeval 2 wat die vergelyking van die programkodelêers en die ontwerpkodelêers bevat

Toetsgeval 1 - Tipe verandering 1 (Kommentaar & indentering)				
Programmerings opdrag	Ooreenkoms persentasie met 3 tokens	Ooreenkoms persentasie met 6 tokens	Ooreenkoms persentasie met 9 tokens	Ooreenkoms persentasie met 12 tokens
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	100.00%	100.00%	100.00%	100.00%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	100.00%	100.00%	100.00%	100.00%
11	100.00%	100.00%	100.00%	100.00%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 1 - Tipe verandering 2 (Veranderlikes)				
Programmerings opdrag	Ooreenkoms persentasie met 3 tokens	Ooreenkoms persentasie met 6 tokens	Ooreenkoms persentasie met 9 tokens	Ooreenkoms persentasie met 12 tokens
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	100.00%	100.00%	100.00%	100.00%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	100.00%	100.00%	100.00%	100.00%
11	100.00%	100.00%	100.00%	100.00%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 1 - Tipe verandering 3 (Verklarende stellings)				
Programmerings opdrag	Ooreenkoms persentasie met 3 tokens	Ooreenkoms persentasie met 6 tokens	Ooreenkoms persentasie met 9 tokens	Ooreenkoms persentasie met 12 tokens
1	96.43%	86.90%	79.76%	66.67%
2	91.97%	86.13%	75.91%	48.18%
3	89.02%	73.55%	73.55%	60.65%
4	93.79%	82.76%	65.83%	64.83%
5	93.85%	72.75%	72.31%	56.92%
6	89.31%	80.50%	80.50%	80.50%
7	89.04%	89.04%	69.86%	54.79%
8	92.99%	86.62%	76.43%	62.42%
9	89.19%	78.38%	70.27%	56.76%
10	91.11%	77.78%	70.00%	58.89%
11	94.96%	83.45%	61.87%	47.48%
12	93.49%	74.56%	65.09%	65.09%
Gemiddeld:	**92.10%**	**81.04%**	**71.78%**	**60.26%**

Toetsgeval 1 - Tipe verandering 4 (Programmodules)				
Programmerings opdrag	Ooreenkoms persentasie met 3 tokens	Ooreenkoms persentasie met 6 tokens	Ooreenkoms persentasie met 9 tokens	Ooreenkoms persentasie met 12 tokens
1	93.83%	87.65%	79.76%	80.25%
2	94.65%	82.44%	75.91%	82.44%
3	96.64%	91.28%	73.55%	73.83%
4	93.53%	77.70%	65.83%	77.70%
5	90.32%	83.87%	72.31%	69.35%
6	95.42%	90.20%	80.50%	64.05%
7	92.65%	60.29%	69.86%	48.53%
8	93.24%	67.57%	76.43%	43.24%
9	94.67%	94.67%	70.27%	94.67%
10	93.92%	86.19%	70.00%	66.30%
11	93.94%	80.30%	61.87%	80.30%
12	94.41%	84.47%	65.09%	84.47%
Gemiddeld:	**93.94%**	**82.22%**	**71.78%**	**72.09%**

Toetsgeval 1 - Tipe verandering 5 (Stellings)				
Programmerings opdrag	Ooreenkoms persentasie met 3 tokens	Ooreenkoms persentasie met 6 tokens	Ooreenkoms persentasie met 9 tokens	Ooreenkoms persentasie met 12 tokens
1	90.36%	79.52%	79.52%	68.67%
2	92.75%	92.75%	92.75%	92.75%
3	93.59%	93.59%	93.59%	93.59%
4	93.15%	93.15%	93.15%	93.15%
5	92.31%	92.31%	81.54%	81.54%
6	92.50%	92.50%	92.50%	92.50%
7	93.15%	93.15%	93.15%	93.15%
8	92.31%	87.18%	76.92%	76.92%
9	93.24%	93.24%	93.24%	93.24%
10	94.44%	94.44%	94.44%	94.44%
11	92.86%	92.86%	92.86%	92.86%
12	94.12%	94.12%	94.12%	94.12%
Gemiddeld:	**92.90%**	**91.57%**	**89.82%**	**88.91%**

Toetsgeval 1 - Tipe verandering 6 (Besluit logika)				
Programmerings opdrag	Ooreenkoms persentasie met 3 tokens	Ooreenkoms persentasie met 6 tokens	Ooreenkoms persentasie met 9 tokens	Ooreenkoms persentasie met 12 tokens
1	98.24%	98.25%	88.89%	88.89%
2	98.55%	98.55%	98.55%	84.06%
3	98.01%	98.01%	98.01%	98.01%
4	98.61%	98.61%	90.28%	90.28%
5	98.46%	98.46%	98.46%	81.54%
6	98.72%	98.72%	98.72%	98.72%
7	98.63%	98.63%	98.63%	98.63%
8	98.73%	98.73%	98.73%	98.73%
9	98.61%	98.61%	98.61%	83.33%
10	98.88%	98.88%	98.88%	87.64%
11	98.57%	98.57%	98.57%	84.29%
12	98.82%	98.82%	91.76%	91.76%
Gemiddeld:	**98.57%**	**98.57%**	**96.51%**	**90.49%**

Toetsgeval 1 - Tipe verandering 7 (Skuif van kontroles)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	100.00%	100.00%	100.00%	100.00%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	100.00%	100.00%	100.00%	100.00%
11	100.00%	100.00%	100.00%	100.00%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 1 - Tipe verandering 8 (Herbenaming van GGK kontroles)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	100.00%	100.00%	100.00%	100.00%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	100.00%	100.00%	100.00%	100.00%
11	100.00%	100.00%	100.00%	100.00%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 1 - Tipe verandering 9 (Verandering van teks in GGK)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	98.86%	94.32%	94.32%	94.32%
2	98.96%	98.97%	98.97%	87.63%
3	98.98%	94.90%	94.90%	94.90%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	99.09%	95.45%	95.45%	95.45%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	98.17%	94.50%	88.99%	88.99%
11	99.48%	99.48%	99.48%	99.48%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**99.46%**	**98.13%**	**97.68%**	**96.73%**

Toetsgeval 1 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)				
Programmerings opdrag	Ooreenkoms persentasie met 3 tokens	Ooreenkoms persentasie met 6 tokens	Ooreenkoms persentasie met 9 tokens	Ooreenkoms persentasie met 12 tokens
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	100.00%	100.00%	100.00%	100.00%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	100.00%	100.00%	100.00%	100.00%
11	100.00%	100.00%	100.00%	100.00%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 2 - Tipe verandering 1 (Kommentaar & indentering)				
Programmerings opdrag	Ooreenkoms persentasie met 3 tokens	Ooreenkoms persentasie met 6 tokens	Ooreenkoms persentasie met 9 tokens	Ooreenkoms persentasie met 12 tokens
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	100.00%	100.00%	100.00%	100.00%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	100.00%	100.00%	100.00%	100.00%
11	100.00%	100.00%	100.00%	100.00%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 2 - Tipe verandering 2 (Veranderlikes)				
Programmerings opdrag	Ooreenkoms persentasie met 3 tokens	Ooreenkoms persentasie met 6 tokens	Ooreenkoms persentasie met 9 tokens	Ooreenkoms persentasie met 12 tokens
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	100.00%	100.00%	100.00%	100.00%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	100.00%	100.00%	100.00%	100.00%
11	100.00%	100.00%	100.00%	100.00%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 2 - Tipe verandering 3 (Verklarende stellings)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	98.21%	93.45%	89.88%	83.33%
2	95.99%	93.07%	87.96%	74.09%
3	94.51%	86.77%	86.77%	80.32%
4	96.90%	91.38%	82.91%	82.41%
5	96.92%	86.38%	86.15%	78.46%
6	94.65%	90.25%	90.25%	90.25%
7	94.52%	94.52%	84.93%	77.40%
8	96.50%	93.31%	88.22%	81.21%
9	94.59%	89.19%	85.14%	78.38%
10	95.56%	88.89%	85.00%	79.44%
11	97.48%	91.73%	80.94%	73.74%
12	96.75%	87.28%	82.54%	82.54%
Gemiddeld:	**96.05%**	**90.52%**	**85.89%**	**80.13%**

Toetsgeval 2 - Tipe verandering 4 (Programmodules)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	96.91%	93.83%	90.12%	90.12%
2	97.33%	91.22%	91.22%	91.22%
3	98.32%	95.64%	86.91%	86.91%
4	96.76%	88.85%	88.85%	88.85%
5	95.16%	91.94%	91.94%	84.68%
6	97.71%	95.10%	82.03%	82.03%
7	96.32%	80.15%	74.26%	74.26%
8	96.62%	83.78%	79.05%	71.62%
9	97.33%	97.33%	97.33%	97.33%
10	96.96%	93.09%	83.15%	83.15%
11	96.97%	90.15%	90.15%	90.15%
12	97.20%	92.24%	92.24%	92.24%
Gemiddeld:	**96.97%**	**91.11%**	**87.27%**	**86.05%**

Toetsgeval 2 - Tipe verandering 5 (Stellings)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	95.18%	89.76%	89.76%	84.34%
2	96.38%	96.38%	96.38%	96.38%
3	96.79%	96.79%	96.79%	96.79%
4	96.58%	96.58%	96.58%	96.58%
5	96.15%	96.15%	90.77%	90.77%
6	96.25%	96.25%	96.25%	96.25%
7	96.58%	96.58%	96.58%	96.58%
8	96.15%	93.59%	88.46%	88.46%
9	96.62%	96.62%	96.62%	96.62%
10	97.22%	97.22%	97.22%	97.22%
11	96.43%	96.43%	96.43%	96.43%
12	97.06%	97.06%	97.06%	97.06%
Gemiddeld:	**96.45%**	**95.78%**	**94.91%**	**94.46%**

Toetsgeval 2 - Tipe verandering 6 (Besluit logika)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	99.12%	99.12%	94.44%	94.44%
2	99.28%	99.28%	99.28%	92.03%
3	99.01%	99.01%	99.01%	99.01%
4	99.31%	99.31%	95.14%	95.14%
5	99.23%	99.23%	99.23%	90.77%
6	99.36%	99.36%	99.36%	99.36%
7	99.32%	99.32%	99.32%	99.32%
8	99.37%	99.37%	99.37%	99.37%
9	99.31%	99.31%	99.31%	91.67%
10	99.44%	99.44%	99.44%	93.82%
11	99.29%	99.29%	99.29%	92.14%
12	99.41%	99.41%	95.88%	95.88%
Gemiddeld:	**99.28%**	**99.29%**	**98.25%**	**95.24%**

Toetsgeval 2 - Tipe verandering 7 (Skuif van kontroles)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	100.00%	100.00%	100.00%	100.00%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	100.00%	100.00%	100.00%	100.00%
11	100.00%	100.00%	100.00%	100.00%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 2 - Tipe verandering 8 (Herbenaming van GGK kontroles)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	100.00%	100.00%	100.00%	100.00%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	100.00%	100.00%	100.00%	100.00%
11	100.00%	100.00%	100.00%	100.00%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 2 - Tipe verandering 9 (Verandering van teks in GGK)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	99.43%	97.16%	97.16%	97.16%
2	99.48%	99.48%	99.48%	93.81%
3	99.49%	97.45%	97.45%	97.45%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	99.55%	97.73%	97.73%	97.73%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	99.08%	97.25%	94.50%	94.50%
11	99.74%	99.74%	99.74%	99.74%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**99.73%**	**99.07%**	**98.84%**	**98.37%**

Toetsgeval 2 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)				
Programmerings opdrag	**Ooreenkoms persentasie met 3 tokens**	**Ooreenkoms persentasie met 6 tokens**	**Ooreenkoms persentasie met 9 tokens**	**Ooreenkoms persentasie met 12 tokens**
1	100.00%	100.00%	100.00%	100.00%
2	100.00%	100.00%	100.00%	100.00%
3	100.00%	100.00%	100.00%	100.00%
4	100.00%	100.00%	100.00%	100.00%
5	100.00%	100.00%	100.00%	100.00%
6	100.00%	100.00%	100.00%	100.00%
7	100.00%	100.00%	100.00%	100.00%
8	100.00%	100.00%	100.00%	100.00%
9	100.00%	100.00%	100.00%	100.00%
10	100.00%	100.00%	100.00%	100.00%
11	100.00%	100.00%	100.00%	100.00%
12	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Moss

A System for Detecting Software Similarity

Die resultate van MOSS word as volg weergegee. Die ooreenkomspersentasie van opsie M = 3, 6, 9 en 12 van al twaalf programmeringsopdragte vir tipe verandering 1 tot 10 word in die onderstande tabelle weergegee. Toetsgeval 1 wat bestaan uit die vergelyking van die programkodelêers word eers weergegee, daarna toetsgeval 2 wat die vergelyking van die programkodelêers en die ontwerpkodelêers bevat.

Toetsgeval 1 - Tipe verandering 1 (Kommentaar & indentering)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	97.00%	97.00%	97.00%	97.00%
2	98.00%	98.00%	98.00%	98.00%
3	98.00%	98.00%	98.00%	98.00%
4	99.00%	99.00%	99.00%	99.00%
5	99.00%	99.00%	99.00%	99.00%
6	98.00%	98.00%	98.00%	98.00%
7	99.00%	99.00%	99.00%	99.00%
8	85.00%	85.00%	85.00%	85.00%
9	99.00%	99.00%	99.00%	99.00%
10	98.00%	98.00%	98.00%	98.00%
11	98.00%	98.00%	98.00%	98.00%
12	98.00%	98.00%	98.00%	98.00%
Gemiddeld:	**97.17%**	**97.17%**	**97.17%**	**97.17%**

Toetsgeval 1 - Tipe verandering 2 (Veranderlikes)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	97.00%	97.00%	97.00%	97.00%
2	98.00%	98.00%	98.00%	98.00%
3	98.00%	98.00%	98.00%	98.00%
4	99.00%	99.00%	99.00%	99.00%
5	99.00%	99.00%	99.00%	99.00%
6	98.00%	98.00%	98.00%	98.00%
7	99.00%	99.00%	99.00%	99.00%
8	85.00%	85.00%	85.00%	85.00%
9	99.00%	99.00%	99.00%	99.00%
10	98.00%	98.00%	98.00%	98.00%
11	98.00%	98.00%	98.00%	98.00%
12	98.00%	98.00%	98.00%	98.00%
Gemiddeld:	**97.17%**	**97.17%**	**97.17%**	**97.17%**

Toetsgeval 1 - Tipe verandering 3 (Verklarende stellings)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	49.00%	49.00%	49.00%	49.00%
2	48.00%	48.00%	48.00%	48.00%
3	58.50%	58.50%	58.50%	58.50%
4	54.00%	54.00%	54.00%	54.00%
5	64.00%	64.00%	64.00%	64.00%
6	74.50%	74.50%	74.50%	74.50%
7	69.50%	69.50%	69.50%	69.50%
8	46.50%	46.50%	46.50%	46.50%
9	44.50%	44.50%	44.50%	44.50%
10	58.50%	58.50%	58.50%	58.50%
11	33.50%	33.50%	33.50%	33.50%
12	42.50%	42.50%	42.50%	42.50%
Gemiddeld:	**53.58%**	**53.58%**	**53.58%**	**53.58%**

Toetsgeval 1 - Tipe verandering 4 (Programmodules)				
Programmerings opdrag	Ooreenkoms persentasie met opsie M = 3	Ooreenkoms persentasie met opsie M = 6	Ooreenkoms persentasie met opsie M = 9	Ooreenkoms persentasie met opsie M = 12
1	81.00%	81.00%	81.00%	81.00%
2	72.50%	72.50%	72.50%	72.50%
3	68.00%	68.00%	68.00%	68.00%
4	77.50%	77.50%	77.50%	77.50%
5	67.50%	67.50%	67.50%	67.50%
6	60.00%	60.00%	60.00%	60.00%
7	45.00%	45.00%	45.00%	45.00%
8	44.50%	44.50%	44.50%	44.50%
9	86.00%	86.00%	86.00%	86.00%
10	63.00%	63.00%	63.00%	63.00%
11	66.00%	66.00%	66.00%	66.00%
12	78.00%	78.00%	78.00%	78.00%
Gemiddeld:	**67.42%**	**67.42%**	**67.42%**	**67.42%**

Toetsgeval 1 - Tipe verandering 5 (Stellings)				
Programmerings opdrag	Ooreenkoms persentasie met opsie M = 3	Ooreenkoms persentasie met opsie M = 6	Ooreenkoms persentasie met opsie M = 9	Ooreenkoms persentasie met opsie M = 12
1	76.50%	76.50%	76.50%	76.50%
2	67.00%	67.00%	67.00%	67.00%
3	79.50%	79.50%	79.50%	79.50%
4	81.00%	81.00%	81.00%	81.00%
5	68.50%	68.50%	68.50%	68.50%
6	76.00%	76.00%	76.00%	76.00%
7	79.50%	79.50%	79.50%	79.50%
8	85.50%	85.50%	85.50%	85.50%
9	84.50%	84.50%	84.50%	84.50%
10	76.50%	76.50%	76.50%	76.50%
11	77.00%	77.00%	77.00%	77.00%
12	81.50%	81.50%	81.50%	81.50%
Gemiddeld:	**77.75%**	**77.75%**	**77.75%**	**77.75%**

Toetsgeval 1 - Tipe verandering 6 (Besluit logika)				
Programmerings opdrag	Ooreenkoms persentasie met opsie M = 3	Ooreenkoms persentasie met opsie M = 6	Ooreenkoms persentasie met opsie M = 9	Ooreenkoms persentasie met opsie M = 12
1	82.00%	82.00%	82.00%	82.00%
2	82.50%	82.50%	82.50%	82.50%
3	84.00%	84.00%	84.00%	84.00%
4	80.50%	80.50%	80.50%	80.50%
5	82.50%	82.50%	82.50%	82.50%
6	82.50%	82.50%	82.50%	82.50%
7	85.00%	85.00%	85.00%	85.00%
8	74.00%	74.00%	74.00%	74.00%
9	84.50%	84.50%	84.50%	84.50%
10	84.00%	84.00%	84.00%	84.00%
11	84.50%	84.50%	84.50%	84.50%
12	81.50%	81.50%	81.50%	81.50%
Gemiddeld:	**82.29%**	**82.29%**	**82.29%**	**82.29%**

Toetsgeval 1 - Tipe verandering 7 (Skuif van kontroles)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	79.00%	79.00%	79.00%	79.00%
2	55.00%	55.00%	55.00%	55.00%
3	63.00%	63.00%	63.00%	63.00%
4	51.00%	51.00%	51.00%	51.00%
5	44.00%	44.00%	44.00%	44.00%
6	65.00%	65.00%	65.00%	65.00%
7	49.00%	49.00%	49.00%	49.00%
8	79.00%	79.00%	79.00%	79.00%
9	64.00%	64.00%	64.00%	64.00%
10	50.00%	50.00%	50.00%	50.00%
11	55.00%	55.00%	55.00%	55.00%
12	66.00%	66.00%	66.00%	66.00%
Gemiddeld:	**60.00%**	**60.00%**	**60.00%**	**60.00%**

Toetsgeval 1 - Tipe verandering 8 (Herbenaming van GGK kontroles)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	79.00%	79.00%	79.00%	79.00%
2	55.00%	55.00%	55.00%	55.00%
3	63.00%	63.00%	63.00%	63.00%
4	51.00%	51.00%	51.00%	51.00%
5	44.00%	44.00%	44.00%	44.00%
6	65.00%	65.00%	65.00%	65.00%
7	49.00%	49.00%	49.00%	49.00%
8	79.00%	79.00%	79.00%	79.00%
9	64.00%	64.00%	64.00%	64.00%
10	50.00%	50.00%	50.00%	50.00%
11	55.00%	55.00%	55.00%	55.00%
12	66.00%	66.00%	66.00%	66.00%
Gemiddeld:	**60.00%**	**60.00%**	**60.00%**	**60.00%**

Toetsgeval 1 - Tipe verandering 9 (Verandering van teks in GGK)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	67.50%	67.50%	67.50%	67.50%
2	53.50%	53.50%	53.50%	53.50%
3	48.50%	48.50%	48.50%	48.50%
4	51.00%	51.00%	51.00%	51.00%
5	44.00%	44.00%	44.00%	44.00%
6	65.50%	65.50%	65.50%	65.50%
7	42.50%	42.50%	42.50%	42.50%
8	79.00%	79.00%	79.00%	79.00%
9	64.00%	64.00%	64.00%	64.00%
10	40.00%	40.00%	40.00%	40.00%
11	50.00%	50.00%	50.00%	50.00%
12	66.00%	66.00%	66.00%	66.00%
Gemiddeld:	**55.96%**	**55.96%**	**55.96%**	**55.96%**

Toetsgeval 1 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	79.00%	79.00%	79.00%	79.00%
2	55.00%	55.00%	55.00%	55.00%
3	63.00%	63.00%	63.00%	63.00%
4	51.00%	51.00%	51.00%	51.00%
5	44.00%	44.00%	44.00%	44.00%
6	65.00%	65.00%	65.00%	65.00%
7	49.00%	49.00%	49.00%	49.00%
8	79.00%	79.00%	79.00%	79.00%
9	64.00%	64.00%	64.00%	64.00%
10	50.00%	50.00%	50.00%	50.00%
11	55.00%	55.00%	55.00%	55.00%
12	66.00%	66.00%	66.00%	66.00%
Gemiddeld:	**60.00%**	**60.00%**	**60.00%**	**60.00%**

Toetsgeval 2 - Tipe verandering 1 (Kommentaar & indentering)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	88.00%	88.00%	88.00%	88.00%
2	76.50%	76.50%	76.50%	76.50%
3	80.50%	80.50%	80.50%	80.50%
4	75.00%	75.00%	75.00%	75.00%
5	71.50%	71.50%	71.50%	71.50%
6	81.50%	81.50%	81.50%	81.50%
7	74.00%	74.00%	74.00%	74.00%
8	82.00%	82.00%	82.00%	82.00%
9	81.50%	81.50%	81.50%	81.50%
10	74.00%	74.00%	74.00%	74.00%
11	76.50%	76.50%	76.50%	76.50%
12	82.00%	82.00%	82.00%	82.00%
Gemiddeld:	**78.58%**	**78.58%**	**78.58%**	**78.58%**

Toetsgeval 2 - Tipe verandering 2 (Veranderlikes)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	88.00%	88.00%	88.00%	88.00%
2	76.50%	76.50%	76.50%	76.50%
3	80.50%	80.50%	80.50%	80.50%
4	75.00%	75.00%	75.00%	75.00%
5	71.50%	71.50%	71.50%	71.50%
6	80.00%	80.00%	80.00%	80.00%
7	74.00%	74.00%	74.00%	74.00%
8	82.00%	82.00%	82.00%	82.00%
9	81.50%	81.50%	81.50%	81.50%
10	74.00%	74.00%	74.00%	74.00%
11	76.50%	76.50%	76.50%	76.50%
12	82.00%	82.00%	82.00%	82.00%
Gemiddeld:	**78.46%**	**78.46%**	**78.46%**	**78.46%**

Toetsgeval 2 - Tipe verandering 3 (Verklarende stellings)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	64.00%	64.00%	64.00%	64.00%
2	51.75%	51.75%	51.75%	51.75%
3	60.75%	60.75%	60.75%	60.75%
4	52.50%	52.50%	52.50%	52.50%
5	54.00%	54.00%	54.00%	54.00%
6	69.75%	69.75%	69.75%	69.75%
7	59.25%	59.25%	59.25%	59.25%
8	62.75%	62.75%	62.75%	62.75%
9	54.25%	54.25%	54.25%	54.25%
10	54.50%	54.50%	54.50%	54.50%
11	44.25%	44.25%	44.25%	44.25%
12	54.25%	54.25%	54.25%	54.25%
Gemiddeld:	**56.83%**	**56.83%**	**56.83%**	**56.83%**

Toetsgeval 2 - Tipe verandering 4 (Programmodules)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	80.00%	80.00%	80.00%	80.00%
2	64.00%	64.00%	64.00%	64.00%
3	65.50%	65.50%	65.50%	65.50%
4	64.25%	64.25%	64.25%	64.25%
5	55.75%	55.75%	55.75%	55.75%
6	62.50%	62.50%	62.50%	62.50%
7	47.00%	47.00%	47.00%	47.00%
8	61.75%	61.75%	61.75%	61.75%
9	75.00%	75.00%	75.00%	75.00%
10	56.50%	56.50%	56.50%	56.50%
11	60.50%	60.50%	60.50%	60.50%
12	72.00%	72.00%	72.00%	72.00%
Gemiddeld:	**63.73%**	**63.73%**	**63.73%**	**63.73%**

Toetsgeval 2 - Tipe verandering 5 (Stellings)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	77.75%	77.75%	77.75%	77.75%
2	61.00%	61.00%	61.00%	61.00%
3	71.25%	71.25%	71.25%	71.25%
4	66.00%	66.00%	66.00%	66.00%
5	56.25%	56.25%	56.25%	56.25%
6	70.50%	70.50%	70.50%	70.50%
7	64.25%	64.25%	64.25%	64.25%
8	82.25%	82.25%	82.25%	82.25%
9	74.25%	74.25%	74.25%	74.25%
10	63.25%	63.25%	63.25%	63.25%
11	66.00%	66.00%	66.00%	66.00%
12	73.75%	73.75%	73.75%	73.75%
Gemiddeld:	**68.88%**	**68.88%**	**68.88%**	**68.88%**

Toetsgeval 2 - Tipe verandering 6 (Besluit logika)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	80.50%	80.50%	80.50%	80.50%
2	68.75%	68.75%	68.75%	68.75%
3	73.50%	73.50%	73.50%	73.50%
4	65.75%	65.75%	65.75%	65.75%
5	63.25%	63.25%	63.25%	63.25%
6	73.75%	73.75%	73.75%	73.75%
7	67.00%	67.00%	67.00%	67.00%
8	76.75%	76.75%	76.75%	76.75%
9	74.25%	74.25%	74.25%	74.25%
10	67.00%	67.00%	67.00%	67.00%
11	69.75%	69.75%	69.75%	69.75%
12	73.50%	73.50%	73.50%	73.50%
Gemiddeld:	**71.15%**	**71.15%**	**71.15%**	**71.15%**

Toetsgeval 2 - Tipe verandering 7 (Skuif van kontroles)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	88.00%	88.00%	88.00%	88.00%
2	76.50%	76.50%	76.50%	76.50%
3	80.50%	80.50%	80.50%	80.50%
4	75.00%	75.00%	75.00%	75.00%
5	71.50%	71.50%	71.50%	71.50%
6	81.50%	81.50%	81.50%	81.50%
7	74.00%	74.00%	74.00%	74.00%
8	82.00%	82.00%	82.00%	82.00%
9	81.50%	81.50%	81.50%	81.50%
10	74.00%	74.00%	74.00%	74.00%
11	76.50%	76.50%	76.50%	76.50%
12	82.00%	82.00%	82.00%	82.00%
Gemiddeld:	**78.58%**	**78.58%**	**78.58%**	**78.58%**

Toetsgeval 2 - Tipe verandering 8 (Herbenaming van GGK kontroles)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	88.00%	88.00%	88.00%	88.00%
2	76.50%	76.50%	76.50%	76.50%
3	80.50%	80.50%	80.50%	80.50%
4	75.00%	75.00%	75.00%	75.00%
5	71.50%	71.50%	71.50%	71.50%
6	81.50%	81.50%	81.50%	81.50%
7	74.00%	74.00%	74.00%	74.00%
8	82.00%	82.00%	82.00%	82.00%
9	81.50%	81.50%	81.50%	81.50%
10	74.00%	74.00%	74.00%	74.00%
11	76.50%	76.50%	76.50%	76.50%
12	82.00%	82.00%	82.00%	82.00%
Gemiddeld:	**78.58%**	**78.58%**	**78.58%**	**78.58%**

Toetsgeval 2 - Tipe verandering 9 (Verandering van teks in GGK)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	82.00%	82.00%	82.00%	82.00%
2	75.75%	75.75%	75.75%	75.75%
3	73.25%	73.25%	73.25%	73.25%
4	76.25%	76.25%	76.25%	76.25%
5	71.50%	71.50%	71.50%	71.50%
6	81.50%	81.50%	81.50%	81.50%
7	70.75%	70.75%	70.75%	70.75%
8	82.00%	82.00%	82.00%	82.00%
9	81.50%	81.50%	81.50%	81.50%
10	69.00%	69.00%	69.00%	69.00%
11	74.00%	74.00%	74.00%	74.00%
12	82.00%	82.00%	82.00%	82.00%
Gemiddeld:	**76.63%**	**76.63%**	**76.63%**	**76.63%**

Toetsgeval 2 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)				
Programmerings opdrag	**Ooreenkoms persentasie met opsie M = 3**	**Ooreenkoms persentasie met opsie M = 6**	**Ooreenkoms persentasie met opsie M = 9**	**Ooreenkoms persentasie met opsie M = 12**
1	88.00%	88.00%	88.00%	88.00%
2	76.50%	76.50%	76.50%	76.50%
3	80.50%	80.50%	80.50%	80.50%
4	75.00%	75.00%	75.00%	75.00%
5	71.50%	71.50%	71.50%	71.50%
6	81.50%	81.50%	81.50%	81.50%
7	74.00%	74.00%	74.00%	74.00%
8	82.00%	82.00%	82.00%	82.00%
9	81.50%	81.50%	81.50%	81.50%
10	74.00%	74.00%	74.00%	74.00%
11	76.50%	76.50%	76.50%	76.50%
12	82.00%	82.00%	82.00%	82.00%
Gemiddeld:	**78.58%**	**78.58%**	**78.58%**	**78.58%**

Simian - Similarity Analyser

Die resultate van Simain word as volg weergegee. Die aantal lyne wat ooreenstem met die drumpelwaardes 3, 6, 9 en 12 asook die ooreenkomspersentasies van die drumpelwaardes van al twaalf programmeringsopdragte vir tipe verandering 1 tot 10 word in die onderstaande tabelle weergegee. Toetsgeval 1 wat bestaan uit die vergelyking van die programkodelêers word eers weergegee, daarna toetsgeval 2 wat die vergelyking van die programkodelêers en die ontwerpkodelêers bevat

Toetsgeval 1 - Tipe verandering 1 (Kommentaar & indentering)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	70	70	70	70	70	100.00%	100.00%	100.00%	100.00%
2	60	60	60	60	60	100.00%	100.00%	100.00%	100.00%
3	64	64	64	64	64	100.00%	100.00%	100.00%	100.00%
4	68	68	68	68	68	100.00%	100.00%	100.00%	100.00%
5	62	62	62	62	62	100.00%	100.00%	100.00%	100.00%
6	66	66	66	66	66	100.00%	100.00%	100.00%	100.00%
7	56	56	56	56	56	100.00%	100.00%	100.00%	100.00%
8	58	58	58	58	58	100.00%	100.00%	100.00%	100.00%
9	64	64	64	64	64	100.00%	100.00%	100.00%	100.00%
10	70	70	70	70	70	100.00%	100.00%	100.00%	100.00%
11	64	64	64	64	64	100.00%	100.00%	100.00%	100.00%
12	78	78	78	78	78	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**65.00**	**65.00**	**65.00**	**65.00**	**65.00**	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 1 - Tipe verandering 2 (Veranderlikes)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	70	6	0	0	0	8.57%	0.00%	0.00%	0.00%
2	60	12	0	0	0	20.00%	0.00%	0.00%	0.00%
3	64	6	0	0	0	9.38%	0.00%	0.00%	0.00%
4	68	6	0	0	0	8.82%	0.00%	0.00%	0.00%
5	62	6	0	0	0	9.68%	0.00%	0.00%	0.00%
6	66	6	0	0	0	9.09%	0.00%	0.00%	0.00%
7	56	6	0	0	0	10.71%	0.00%	0.00%	0.00%
8	58	6	0	0	0	10.34%	0.00%	0.00%	0.00%
9	64	6	0	0	0	9.38%	0.00%	0.00%	0.00%
10	70	6	0	0	0	8.57%	0.00%	0.00%	0.00%
11	64	6	0	0	0	9.38%	0.00%	0.00%	0.00%
12	78	6	0	0	0	7.69%	0.00%	0.00%	0.00%
Gemiddeld:	**65.00**	**6.50**	**0.00**	**0.00**	**0.00**	**10.13%**	**0.00%**	**0.00%**	**0.00%**

Toetsgeval 1 - Tipe verandering 3 (Verklarende stellings)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	70	34	26	26	26	48.57%	37.14%	37.14%	37.14%
2	61	36	18	18	0	59.02%	29.51%	29.51%	0.00%
3	65	46	34	22	0	70.77%	52.31%	33.85%	0.00%
4	69	44	20	20	0	63.77%	28.99%	28.99%	0.00%
5	64	26	20	20	0	40.63%	31.25%	31.25%	0.00%
6	67	46	18	18	0	68.66%	26.87%	26.87%	0.00%
7	58	42	14	0	0	72.41%	24.14%	0.00%	0.00%
8	59	30	18	18	0	50.85%	30.51%	30.51%	0.00%
9	66	36	24	24	24	54.55%	36.36%	36.36%	36.36%
10	72	50	24	24	24	69.44%	33.33%	33.33%	33.33%
11	65	30	18	18	0	46.15%	27.69%	27.69%	0.00%
12	79	44	18	18	0	55.70%	22.78%	22.78%	0.00%
Gemiddeld:	**66.25**	**38.67**	**21.00**	**18.83**	**6.17**	**58.38%**	**31.74%**	**28.19%**	**8.90%**

Toetsgeval 1 - Tipe verandering 4 (Programmodules)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	67	48	38	38	0	71.64%	56.72%	56.72%	0.00%
2	58	46	40	40	0	79.31%	68.97%	68.97%	0.00%
3	61	50	38	38	38	81.97%	62.30%	62.30%	62.30%
4	66	50	50	50	32	75.76%	75.76%	75.76%	48.48%
5	60	34	28	0	0	56.67%	46.67%	0.00%	0.00%
6	64	48	34	20	0	75.00%	53.13%	31.25%	0.00%
7	54	24	18	18	0	44.44%	33.33%	33.33%	0.00%
8	56	22	16	0	0	39.29%	28.57%	0.00%	0.00%
9	67	54	54	38	38	80.60%	80.60%	56.72%	56.72%
10	74	46	40	40	40	62.16%	54.05%	54.05%	54.05%
11	60	44	38	38	0	73.33%	63.33%	63.33%	0.00%
12	74	56	56	56	36	75.68%	75.68%	75.68%	48.65%
Gemiddeld:	**63.42**	**43.50**	**37.50**	**31.33**	**15.33**	**67.99%**	**58.26%**	**48.17%**	**22.52%**

Toetsgeval 1 - Tipe verandering 5 (Stellings)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	70	42	42	42	0	60.00%	60.00%	60.00%	0.00%
2	62	38	38	26	26	61.29%	61.29%	41.94%	41.94%
3	66	44	34	18	0	66.67%	51.52%	27.27%	0.00%
4	70	48	38	24	24	68.57%	54.29%	34.29%	34.29%
5	64	32	32	32	32	50.00%	50.00%	50.00%	50.00%
6	68	44	38	38	0	64.71%	55.88%	55.88%	0.00%
7	58	42	36	22	0	72.41%	62.07%	37.93%	0.00%
8	60	40	34	18	0	66.67%	56.67%	30.00%	0.00%
9	66	44	44	28	28	66.67%	66.67%	42.42%	42.42%
10	72	44	44	28	28	61.11%	61.11%	38.89%	38.89%
11	66	44	44	44	26	66.67%	66.67%	66.67%	39.39%
12	80	60	60	44	26	75.00%	75.00%	55.00%	32.50%
Gemiddeld:	**66.83**	**43.50**	**40.33**	**30.33**	**15.83**	**64.98%**	**60.10%**	**45.02%**	**23.29%**

Toetsgeval 1 - Tipe verandering 6 (Besluit logika)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	71	60	50	50	50	84.51%	70.42%	70.42%	70.42%
2	60	52	44	30	30	86.67%	73.33%	50.00%	50.00%
3	63	54	54	24	24	85.71%	85.71%	38.10%	38.10%
4	68	60	54	40	40	88.24%	79.41%	58.82%	58.82%
5	62	50	50	50	30	80.65%	80.65%	80.65%	48.39%
6	66	58	48	48	30	87.88%	72.73%	72.73%	45.45%
7	56	48	48	22	0	85.71%	85.71%	39.29%	0.00%
8	58	50	50	22	0	86.21%	86.21%	37.93%	0.00%
9	64	56	46	32	32	87.50%	71.88%	50.00%	50.00%
10	70	62	52	38	38	88.57%	74.29%	54.29%	54.29%
11	64	56	48	34	34	87.50%	75.00%	53.13%	53.13%
12	78	70	64	48	48	89.74%	82.05%	61.54%	61.54%
Gemiddeld:	**65.00**	**56.33**	**50.67**	**36.50**	**29.67**	**86.57%**	**78.12%**	**55.57%**	**44.18%**

Toetsgeval 1 - Tipe verandering 7 (Skuif van kontroles)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	116	98	66	54	54	84.48%	56.90%	46.55%	46.55%
2	130	116	74	60	60	89.23%	56.92%	46.15%	46.15%
3	132	100	72	60	60	75.76%	54.55%	45.45%	45.45%
4	136	106	60	60	60	77.94%	44.12%	44.12%	44.12%
5	136	106	74	62	62	77.94%	54.41%	45.59%	45.59%
6	146	106	86	48	28	72.60%	58.90%	32.88%	19.18%
7	150	130	80	68	68	86.67%	53.33%	45.33%	45.33%
8	132	100	72	60	60	75.76%	54.55%	45.45%	45.45%
9	150	116	78	66	66	77.33%	52.00%	44.00%	44.00%
10	146	112	66	66	66	76.71%	45.21%	45.21%	45.21%
11	130	96	60	60	60	73.85%	46.15%	46.15%	46.15%
12	132	100	72	60	60	75.76%	54.55%	45.45%	45.45%
Gemiddeld:	**136.33**	**107.17**	**71.67**	**60.33**	**58.67**	**78.67%**	**52.63%**	**44.36%**	**43.22%**

Toetsgeval 1 - Tipe verandering 8 (Herbenaming van GGK kontroles)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	116	28	14	0	0	24.14%	12.07%	0.00%	0.00%
2	130	28	14	0	0	21.54%	10.77%	0.00%	0.00%
3	132	28	14	0	0	21.21%	10.61%	0.00%	0.00%
4	136	28	14	0	0	20.59%	10.29%	0.00%	0.00%
5	136	28	14	0	0	20.59%	10.29%	0.00%	0.00%
6	146	28	14	0	0	19.18%	9.59%	0.00%	0.00%
7	150	28	14	0	0	18.67%	9.33%	0.00%	0.00%
8	132	28	14	0	0	21.21%	10.61%	0.00%	0.00%
9	150	28	14	0	0	18.67%	9.33%	0.00%	0.00%
10	146	28	14	0	0	19.18%	9.59%	0.00%	0.00%
11	130	28	14	0	0	21.54%	10.77%	0.00%	0.00%
12	132	28	14	0	0	21.21%	10.61%	0.00%	0.00%
Gemiddeld:	**136.33**	**28.00**	**14.00**	**0.00**	**0.00**	**20.64%**	**10.32%**	**0.00%**	**0.00%**

Toetsgeval 1 - Tipe verandering 9 (Verandering van teks in GGK)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	118	100	78	52	34	84.75%	66.10%	44.07%	28.81%
2	132	114	86	70	70	86.36%	65.15%	53.03%	53.03%
3	134	110	84	56	36	82.09%	62.69%	41.79%	26.87%
4	136	110	72	56	36	80.88%	52.94%	41.18%	26.47%
5	136	110	74	58	58	80.88%	54.41%	42.65%	42.65%
6	146	106	90	48	30	72.60%	61.64%	32.88%	20.55%
7	152	122	76	76	36	80.26%	50.00%	50.00%	23.68%
8	132	110	94	78	36	83.33%	71.21%	59.09%	27.27%
9	150	122	92	92	52	81.33%	61.33%	61.33%	34.67%
10	150	122	90	78	38	81.33%	60.00%	52.00%	25.33%
11	131	110	92	62	42	83.97%	70.23%	47.33%	32.06%
12	132	112	100	84	84	84.85%	75.76%	63.64%	63.64%
Gemiddeld:	**137.42**	**112.33**	**85.67**	**67.50**	**46.00**	**81.89%**	**62.62%**	**49.08%**	**33.75%**

Toetsgeval 1 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	116	22	14	0	0	18.97%	12.07%	0.00%	0.00%
2	130	22	14	0	0	16.92%	10.77%	0.00%	0.00%
3	132	22	14	0	0	16.67%	10.61%	0.00%	0.00%
4	136	30	14	0	0	22.06%	10.29%	0.00%	0.00%
5	136	28	14	0	0	20.59%	10.29%	0.00%	0.00%
6	146	22	14	0	0	15.07%	9.59%	0.00%	0.00%
7	150	22	14	0	0	14.67%	9.33%	0.00%	0.00%
8	132	22	14	0	0	16.67%	10.61%	0.00%	0.00%
9	150	22	14	0	0	14.67%	9.33%	0.00%	0.00%
10	146	22	14	0	0	15.07%	9.59%	0.00%	0.00%
11	130	22	14	0	0	16.92%	10.77%	0.00%	0.00%
12	132	22	14	0	0	16.67%	10.61%	0.00%	0.00%
Gemiddeld:	**136.33**	**23.17**	**14.00**	**0.00**	**0.00**	**17.08%**	**10.32%**	**0.00%**	**0.00%**

Toetsgeval 2 - Tipe verandering 1 (Kommentaar & indentering)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	186	186	186	186	186	100.00%	100.00%	100.00%	100.00%
2	190	190	190	190	190	100.00%	100.00%	100.00%	100.00%
3	196	196	196	196	196	100.00%	100.00%	100.00%	100.00%
4	204	204	204	204	204	100.00%	100.00%	100.00%	100.00%
5	198	198	198	198	198	100.00%	100.00%	100.00%	100.00%
6	212	212	212	212	212	100.00%	100.00%	100.00%	100.00%
7	206	206	206	206	206	100.00%	100.00%	100.00%	100.00%
8	190	190	190	190	190	100.00%	100.00%	100.00%	100.00%
9	214	214	214	214	214	100.00%	100.00%	100.00%	100.00%
10	216	216	216	216	216	100.00%	100.00%	100.00%	100.00%
11	194	194	194	194	194	100.00%	100.00%	100.00%	100.00%
12	210	210	210	210	210	100.00%	100.00%	100.00%	100.00%
Gemiddeld:	**201.33**	**201.33**	**201.33**	**201.33**	**201.33**	**100.00%**	**100.00%**	**100.00%**	**100.00%**

Toetsgeval 2 - Tipe verandering 2 (Veranderlikes)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	186	122	116	116	116	65.59%	62.37%	62.37%	62.37%
2	190	142	130	130	130	74.74%	68.42%	68.42%	68.42%
3	196	138	132	132	132	70.41%	67.35%	67.35%	67.35%
4	204	142	136	136	136	69.61%	66.67%	66.67%	66.67%
5	198	142	136	136	136	71.72%	68.69%	68.69%	68.69%
6	212	152	146	146	146	71.70%	68.87%	68.87%	68.87%
7	206	156	150	150	150	75.73%	72.82%	72.82%	72.82%
8	190	138	132	132	132	72.63%	69.47%	69.47%	69.47%
9	214	156	150	150	150	72.90%	70.09%	70.09%	70.09%
10	216	152	146	146	146	70.37%	67.59%	67.59%	67.59%
11	194	136	130	130	130	70.10%	67.01%	67.01%	67.01%
12	210	138	132	132	132	65.71%	62.86%	62.86%	62.86%
Gemiddeld:	**201.33**	**142.83**	**136.33**	**136.33**	**136.33**	**70.93%**	**67.68%**	**67.68%**	**67.68%**

Toetsgeval 2 - Tipe verandering 3 (Verklarende stellings)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	186	150	142	142	142	80.65%	76.34%	76.34%	76.34%
2	191	166	148	148	130	86.91%	77.49%	77.49%	68.06%
3	197	178	166	154	132	90.36%	84.26%	78.17%	67.01%
4	205	180	156	156	136	87.80%	76.10%	76.10%	66.34%
5	200	162	156	164	136	81.00%	78.00%	82.00%	68.00%
6	213	192	164	164	146	90.14%	77.00%	77.00%	68.54%
7	208	192	164	150	150	92.31%	78.85%	72.12%	72.12%
8	191	162	150	150	132	84.82%	78.53%	78.53%	69.11%
9	216	186	174	174	174	86.11%	80.56%	80.56%	80.56%
10	218	196	170	170	170	89.91%	77.98%	77.98%	77.98%
11	195	160	148	148	130	82.05%	75.90%	75.90%	66.67%
12	211	176	150	150	132	83.41%	71.09%	71.09%	62.56%
Gemiddeld:	**202.58**	**175.00**	**157.33**	**155.83**	**142.50**	**86.29%**	**77.67%**	**76.94%**	**70.27%**

Toetsgeval 2 - Tipe verandering 4 (Programmodules)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	183	164	154	154	116	89.62%	84.15%	84.15%	63.39%
2	188	176	170	170	130	93.62%	90.43%	90.43%	69.15%
3	193	182	170	170	170	94.30%	88.08%	88.08%	88.08%
4	202	186	186	186	168	92.08%	92.08%	92.08%	83.17%
5	196	170	164	136	136	86.73%	83.67%	69.39%	69.39%
6	210	194	180	166	146	92.38%	85.71%	79.05%	69.52%
7	204	174	168	168	150	85.29%	82.35%	82.35%	73.53%
8	188	154	148	132	132	81.91%	78.72%	70.21%	70.21%
9	217	204	204	188	188	94.01%	94.01%	86.64%	86.64%
10	220	192	186	186	186	87.27%	84.55%	84.55%	84.55%
11	190	174	168	168	130	91.58%	88.42%	88.42%	68.42%
12	206	188	188	188	168	91.26%	91.26%	91.26%	81.55%
Gemiddeld:	**199.75**	**179.83**	**173.83**	**167.67**	**151.67**	**90.01%**	**86.95%**	**83.88%**	**75.63%**

Toetsgeval 2 - Tipe verandering 5 (Stellings)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	186	158	158	158	116	84.95%	84.95%	84.95%	62.37%
2	192	168	168	156	156	87.50%	87.50%	81.25%	81.25%
3	198	176	166	150	132	88.89%	83.84%	75.76%	66.67%
4	206	184	174	160	160	89.32%	84.47%	77.67%	77.67%
5	200	168	168	168	168	84.00%	84.00%	84.00%	84.00%
6	214	190	184	184	146	88.79%	85.98%	85.98%	68.22%
7	208	192	186	172	150	92.31%	89.42%	82.69%	72.12%
8	192	172	166	150	132	89.58%	86.46%	78.13%	68.75%
9	216	194	194	178	178	89.81%	89.81%	82.41%	82.41%
10	218	190	190	174	174	87.16%	87.16%	79.82%	79.82%
11	196	174	174	174	156	88.78%	88.78%	88.78%	79.59%
12	212	192	192	176	158	90.57%	90.57%	83.02%	74.53%
Gemiddeld:	**203.17**	**179.83**	**176.67**	**166.67**	**152.17**	**88.47%**	**86.91%**	**82.04%**	**74.78%**

Toetsgeval 2 - Tipe verandering 6 (Besluit logika)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	187	176	166	166	166	94.12%	88.77%	88.77%	88.77%
2	190	182	174	160	160	95.79%	91.58%	84.21%	84.21%
3	195	186	186	156	156	95.38%	95.38%	80.00%	80.00%
4	204	196	190	176	176	96.08%	93.14%	86.27%	86.27%
5	198	186	186	186	166	93.94%	93.94%	93.94%	83.84%
6	212	204	194	194	176	96.23%	91.51%	91.51%	83.02%
7	206	198	198	172	150	96.12%	96.12%	83.50%	72.82%
8	190	182	182	154	132	95.79%	95.79%	81.05%	69.47%
9	214	206	196	182	182	96.26%	91.59%	85.05%	85.05%
10	216	208	198	184	184	96.30%	91.67%	85.19%	85.19%
11	194	186	178	164	164	95.88%	91.75%	84.54%	84.54%
12	210	202	196	180	180	96.19%	93.33%	85.71%	85.71%
Gemiddeld:	**201.33**	**192.67**	**187.00**	**172.83**	**166.00**	**95.67%**	**92.88%**	**85.81%**	**82.41%**

Toetsgeval 2 - Tipe verandering 7 (Skuif van kontroles)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	186	168	136	124	124	90.32%	73.12%	66.67%	66.67%
2	190	176	134	120	120	92.63%	70.53%	63.16%	63.16%
3	196	164	136	124	124	83.67%	69.39%	63.27%	63.27%
4	204	174	128	128	128	85.29%	62.75%	62.75%	62.75%
5	198	168	136	124	124	84.85%	68.69%	62.63%	62.63%
6	212	172	152	114	94	81.13%	71.70%	53.77%	44.34%
7	206	186	136	124	124	90.29%	66.02%	60.19%	60.19%
8	190	158	130	118	118	83.16%	68.42%	62.11%	62.11%
9	214	180	142	130	130	84.11%	66.36%	60.75%	60.75%
10	216	182	136	136	136	84.26%	62.96%	62.96%	62.96%
11	194	160	124	124	124	82.47%	63.92%	63.92%	63.92%
12	210	178	150	138	138	84.76%	71.43%	65.71%	65.71%
Gemiddeld:	**201.33**	**172.17**	**136.67**	**125.33**	**123.67**	**85.58%**	**67.94%**	**62.32%**	**61.54%**

Toetsgeval 2 - Tipe verandering 8 (Herbenaming van GGK kontroles)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	186	94	80	66	44	50.54%	43.01%	35.48%	23.66%
2	190	82	54	40	40	43.16%	28.42%	21.05%	21.05%
3	196	86	64	50	50	43.88%	32.65%	25.51%	25.51%
4	204	90	68	54	54	44.12%	33.33%	26.47%	26.47%
5	198	72	50	22	0	36.36%	25.25%	11.11%	0.00%
6	212	84	70	56	36	39.62%	33.02%	26.42%	16.98%
7	206	66	36	22	0	32.04%	17.48%	10.68%	0.00%
8	190	68	38	24	24	35.79%	20.00%	12.63%	12.63%
9	214	84	56	42	42	39.25%	26.17%	19.63%	19.63%
10	216	90	62	48	48	41.67%	28.70%	22.22%	22.22%
11	194	86	58	44	44	44.33%	29.90%	22.68%	22.68%
12	210	100	78	50	50	47.62%	37.14%	23.81%	23.81%
Gemiddeld:	**201.33**	**83.50**	**59.50**	**43.17**	**36.00**	**41.53%**	**29.59%**	**21.47%**	**17.89%**

Toetsgeval 2 - Tipe verandering 9 (Verandering van teks in GGK)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	188	170	148	122	104	90.43%	78.72%	64.89%	55.32%
2	192	174	146	130	130	90.63%	76.04%	67.71%	67.71%
3	198	174	148	120	100	87.88%	74.75%	60.61%	50.51%
4	204	178	140	124	104	87.25%	68.63%	60.78%	50.98%
5	198	172	136	120	120	86.87%	68.69%	60.61%	60.61%
6	212	172	156	114	96	81.13%	73.58%	53.77%	45.28%
7	208	178	132	132	92	85.58%	63.46%	63.46%	44.23%
8	190	168	152	136	94	88.42%	80.00%	71.58%	49.47%
9	214	186	156	156	116	86.92%	72.90%	72.90%	54.21%
10	220	192	160	148	108	87.27%	72.73%	67.27%	49.09%
11	195	174	156	126	106	89.23%	80.00%	64.62%	54.36%
12	210	190	178	162	162	90.48%	84.76%	77.14%	77.14%
Gemiddeld:	**202.42**	**177.33**	**150.67**	**132.50**	**111.00**	**87.67%**	**74.52%**	**65.45%**	**54.91%**

Toetsgeval 2 - Tipe verandering 10 (Skuif en herbenaming van kontroles in GGK)									
Programmerings opdrag	**Totale aantal lyne**	**Aantal lyne van drumpelwaarde 3**	**Aantal lyne van drumpelwaarde 6**	**Aantal lyne van drumpelwaarde 9**	**Aantal lyne van drumpelwaarde 12**	**Ooreenkoms persentasie van drumpelwaarde 3**	**Ooreenkoms persentasie van drumpelwaarde 6**	**Ooreenkoms persentasie van drumpelwaarde 9**	**Ooreenkoms persentasie van drumpelwaarde 12**
1	186	88	80	66	44	47.31%	43.01%	35.48%	23.66%
2	190	76	54	40	40	40.00%	28.42%	21.05%	21.05%
3	196	80	64	50	50	40.82%	32.65%	25.51%	25.51%
4	204	92	68	54	54	45.10%	33.33%	26.47%	26.47%
5	198	72	50	22	0	36.36%	25.25%	11.11%	0.00%
6	212	78	70	56	36	36.79%	33.02%	26.42%	16.98%
7	206	60	36	22	0	29.13%	17.48%	10.68%	0.00%
8	190	62	38	24	24	32.63%	20.00%	12.63%	12.63%
9	214	78	56	42	42	36.45%	26.17%	19.63%	19.63%
10	216	84	62	48	48	38.89%	28.70%	22.22%	22.22%
11	194	82	58	44	44	42.27%	29.90%	22.68%	22.68%
12	210	94	78	50	50	44.76%	37.14%	23.81%	23.81%
Gemiddeld:	**201.33**	**78.83**	**59.50**	**43.17**	**36.00**	**39.21%**	**29.59%**	**21.47%**	**17.89%**