Bachelorscriptie Informatica

Radboud Universiteit

Het gebruik van Automated Assessment Tools in programmeercurssusen

Auteur:
Tobias Schröter
s4122283

Inhoudelijk begeleider:
dr. Sjaak Smetsers
s.smetsers@science.ru.nl

Tweede lezer:
prof. dr. Erik Barendsen
E.Barendsen@cs.ru.nl

30 juni 2015

Samenvatting

Het gebruik van automatische beoordelingssystemen (automated assessment tools, kortweg AAT's) bij programmeercursussen heeft verschillende voordelen. Naast dat studenten snel feedback op ingeleverd werk krijgen nemen deze tools (deel van) het nakijkwerk uit handen van de assistenten en/of docenten.
In dit onderzoek richten we ons op de vraag welke factoren bepalend zijn voor succesvolle inzet van AAT's bij programmeeronderwijs. Uit het aanbod van de vrij beschikbare tools selecteren we een AAT die we vervolgens onderwerpen aan een gebruikersonderzoek. Voor deze selectie wordt een lijst van criteria opgesteld op basis waarvan enkele tools worden vergeleken. Het gebruikersonderzoek zelf vindt plaats binnen een bestaande programmeercursus. Hiervoor leveren studenten uitwerkingen van enkele programmeeropdrachten in via de AAT. Gegevens over het gebruik worden verzameld met behulp van vragenlijsten, interviews en automatische logging van de AAT.
Uit de analyse van de resultaten is gebleken dat de studenten de snelle feedback van de AAT inderdaad als een voordeel zien. Een aantal gebruikers geeft zelfs aan dat AAT's van meet af aan bij programmeeronderwijs ingezet zouden moeten worden. Echter, het succes van de tool blijkt ook sterk afhankelijk te zijn van het gebruiksgemak: men is niet of nauwelijks bereid om tijd te investeren in het leren omgaan met de tool. Meer algemeen hebben we geconstateerd dat de motivatie om de tool ook daadwerkelijk te gebruiken laag is ondanks de aantoonbare voordelen.

Inhoudsopgave

Woord vooraf

Nadat ik drie jaar lang op de Radboud Universiteit cursussen gevolgd heb, vooral ook programmeercursussen, had ik het idee het leerproces te verbeteren. Met ondersteuning van mijn begeleider Sjaak Smetsers heb ik de mogelijkheid gezien dit met een Automated Assessment Tool te bereiken. Ik had nooit een experiment gedaan maar vond het wel heel leuk waardoor ik heel positief gestemd was dat het experiment door vele studenten gevolgd wordt. Na de eerste aankonding van het experiment was dit al verdwenen omdat heel weinig studenten zich aangemeld hadden om dit experiment te volgen. Om wel een goede uitvoering te garanderen heb ik zo snel mogelijk alle opdrachten op het systeem gezet en de accounts voor iedere student aangemaakt. Maar tijdens het draaien viel op dat veel studenten niet echt meegewerkt hebben of zelfs nooit op het systeem ingelogd hebben. Aan het einde van mijn onderzoek kan ik zeggen, dat een experiment een leuke manier is om nieuwe strategiën te onderzoeken maar er wel de mogelijkheid moet bestaan de deelnemers van het experiment te motiveren. Een voorbeeld zou zijn dat de studenten alleen het Automated Assessment Tool mogen gebruiken om hun werk in te leveren.
Het schrijven van deze scriptie was voor mij moeilijk omdat ik problemen heb mijn gedachten gestructureerd op te schrijven. Verder is Nederlands ook niet mijn eerste taal.
Mijn dank gaat uit naar mijn begeleiders Sjaak Smetsers en Erik Barendsen voorhun begeleiding en ondersteuning tijdens het maken van mijn onderzoek. Verder gaat mijn dank uit naar Bernadette Smelik, Rick Erkens en Max Tijssen voor het nakijken van taalfouten in mijn scriptie. Tevens gaat mijn dank uit naar mijn familie en vrienden die mij ondersteund hebben bij mijn scriptie.

Inleiding

2.1 Het probleem

In veel gevallen vindt de beoordeling van programmeeropdrachten niet automatisch plaats. De programmacode wordt ingeleverd en vervolgens één tot twee weken later nagekeken. Het probleem bij deze situatie is dat de studenten relatief lang moeten wachten voordat zij feedback krijgen.

2.2 Mogelijke oplossing

Programmeerwedstrijden op het internet maken gebruik van Automated Assessment Tools. UVA-onlinejudge is een van de bekendste aanbieders. [1] De deelnemer kiest een opdracht, lost offline het probleem op en levert vervolgens de uitwerking ter beoordeling in. Iedere opgave heeft een aantal bijbehorende testcases. Met behulp van deze testcases wordt vervolgens de ingeleverde programmacode beoordeeld.

2.3 Vraagstelling

Het doel van dit onderzoek is, een Automated Assessment Tool bij een programmeercursus te gebruiken en te testen wat het succes bepaalt bij het gebruiken van Automated Assessment Tools. Dus:
"Welke aspecten zijn bepalend voor de succesvolle inzet van een Automated Assessment Tool?" Deze vraag zal beantwoord worden aan de hand van de volgende deelvragen:

- Zijn Automated Assessment Tools bruikbaar voor programmeercursussen?
- Beïnvloedt de Automated Assessment Tool de werkwijze van een student?
- Werken studenten met een Automated Assessment Tool structureler?

[1] $http://uva.onlinejudge.org/$

Deze deelvragen worden beantwoord met de resultaten van het experiment dat bij dit onderzoek hoort.

2.4 Motivatie

De reden om dit onderzoek uit te voeren is de eigen ervaring met de late feedback op programmeeropdrachten. Verder wordt verwacht dat het automatieseren van de feedback twee voordelen oplevert:

- student komt sneller te weten wat goed/fout ging, snel feedback beklijft beter
- docent wordt werk uit handen genomen

Het is de moeite waard dit onderzoek uit te voeren omdat programmeren een van de grootste onderdelen is van de studie informatica. Met een positief resultaat zou de mogelijkheid bestaan het leerprocess met behulp van Automated Assessment Tools te verbeteren.

Achtergrond

Aan het begin van dit hoofdstuk wordt een aantal onderzoeken opgesomd die al met Automated Assessment Tools gewerkt hebben. Verder wordt een inventarisatie van een aantal Automated Assessment Tools (AATs) gegeven. Daarbij worden alle tools op dezelfde criteria beoordeeld. Met behulp van deze beoordeling wordt de beste tool gekozen. Verder wordt kort uitgelegd wat het experiment inhoudt en bij welke cursus het experiment uitgevoerd wordt.
Binnen het onderzoeksgebied 'Computing Education' wordt veel onderzoek gedaan naar de leerproblemen die studenten ervaren en manieren om die te verbeteren. Kleiner et al.([8]) hebben dit toegepast op de programmeertaal SQL. SQL is een van de belangrijkste programmeertalen om databases op te zetten. Het onderzoek laat zien dat met behulp van het ontwikkelde systeem het mogelijk is om het leerproces van de studenten te verbeteren. Verder laat het zien dat de studenten door het systeem de programmacode meermaals in kunnen leveren en direct kunnen testen of deze goed is. Een interessant resultaat is dat de beoordeling van de ingeleverde SQL-statements efficiënter wordt. Dit betekent dat het mogelijk is automatisch een beoordeling op ingeleverde SQL-statements te geven. Dit resultaat laat zien dat het mogelijk is met een AAT programmacode te beoordelen.
In dit onderzoek wordt met behulp van een vragenlijst de usability van de AAT getest. Dit wordt met de 'System Usability Scale'(SUS) gedaan ([3]). De SUS bestaat uit tien vragen die gericht zijn op de usability van het systeem dat getest wordt. Het is een oudere techniek die nog steeds gebruikt wordt([2]). Aaron Bangor et al. hebben deze vragen aangepast. Een voorbeeld is dat in een van de oorspronkelijke vragen het Engelse woord 'cumbersome' gebruikt wordt. Dit wordt veranderd naar 'awkward' omdat dit vaker gebruikt wordt(Oxford University Computing Service, 2001).
In het paper van Aaron Bangor et al. worden meerdere Usability-vragenlijsten met elkaar vergeleken aan de hand van de betrouwbaarheid. De betrouwbaarheid van de SUS is 0.85 [2]. In vergelijking met de andere Usability-vragenlijsten is dit volgens de betrouwbaarheid niet de beste keuze. In dit onderzoek zullen de studenten de vragenlijst schriftelijk invullen en dit is de reden ervoor dat de SUS gebruikt wordt omdat de andere vragenlijsten, met een betere betrouwbaarheid, via internet ingevuld worden. De reden daar-

voor is dat studenten op online enquêtes vaak niet reageren. In de tabel van Aaron Bangor et al. [2] worden nog twee andere Usability Scales opgesomd waar het ook mogelijk is, de vragenlijst schriftelijk in te vullen. Dit onderzoek maakt er geen gebruik van omdat vam de eerste de betrouwbaarheid niet bekend was("Usefulness, Satisfaction and Ease of Use", Kirakowski , Claridge, and Whitehand (1998)) en de tweede slechts uit drie vragen bestond (After Scenario Questionnaire, Lewis (1995)).
In José Luis Fernández Alemán ([1]) wordt een andere interessante conclusie getrokken. De conclusie was dat ranking en een directe beoordeling van de ingeleverde programmacode een positief effect heeft op het leren van de studenten. Bij het experiment van het onderzoek heeft een groep studenten met *Mooshak* gewerkt en een controlegroep heeft gebruik gemaakt van het oude proces. Daarbij viel op dat de cijfers van de *Mooshak* groep gemiddeld hoger waren dan de cijfers van de controlegroep. De conclusie, dat ranking de studenten motiveert wordt in mijn onderzoek ook getest.
Rubio-Sanchez [11] concluderen dat het gebruik van AATs geen meetbaar positief effect heeft op de programmeergewoonten of het uitvalpercentage. Het doel van het onderzoek was te testen of studenten het leuk vinden met een AAT, in dit geval *Mooshak*, te werken en of het een positief effect heeft op het uitvalpercentage. In het onderzoek [11] worden de studenten ook met een vragenlijst geënquêteerd. In totaal worden er 25 vragen gesteld. Deze zijn gericht op de gebruikte AAT en zijn verdeeld over vier gebieden. Het resultaat laat zien dat studenten het niet beter vinden met *Mooshak* te werken en het geen positief effect heeft op de uitvalpercentage. Conclusies uit dit onderzoek zijn vooral dat Mooshak het beoordelen voor de docent makkelijker maakt en de docent de opdrachtbeschrijving goed moet maken, zodat deterministische programma's ontstaan, die met een AAT getest kunnen worden. Dit onderzoek probeert te testen of studenten het beter vinden met een AAT te werken en maakt daarbij gebruik van een aantal vragen die dit onderzoek ook gebruikt heeft.
AATs zoals *SharifJudge* zijn alleen bij programmeercrusussen te gebruiken, maar er zijn ook tools die geschikt zijn voor andere soort cursussen. Het systeem *QUEST*([13]) heeft als doel dat de studenten beter kunnen leren als zij dit actief doen met de tool. Het resultaat van een onderzoek waar zij de tool gebruikt hebben, heeft positieve uitkomsten opgeleverd.

De studenten die met het QUEST-systeem gewerkt hebben, hadden bij het experiment betere eindcijfers dan de studenten die dit niet gedaan hebben. Verder vonden de studenten het leuk om met de tool te werken([10]).
De genoemde onderzoeken proberen het onderwerp van actief leren toe te passen. Dat dit een goede manier is wordt aangetoond in ([4][12]).

3.1 Wat doen Automated Assessment Tools?

Het doel van een AAT is dat programmacode automatisch beoordeeld wordt. De AAT's die in dit onderzoek worden gebruikt, testen het ingeleverde programma op syntax en op semantiek. De semantiek wordt getoetst door het programma uit te voeren op een aantal testcases.

3.2 Selectiecriteria

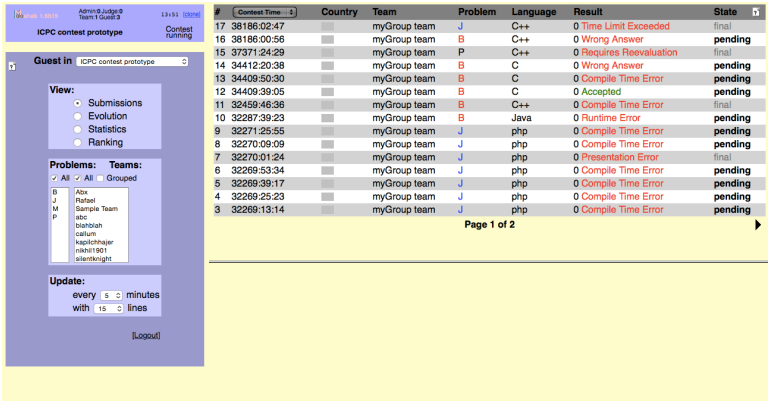
Om de juiste AAT voor dit onderzoek te vinden stellen we een aantal criteria op waarmee de AAT geselecteerd wordt. Het feit dat een AAT al bij een onderzoek of onderzoeksinstelling gebruikt wordt is één van de selectiepunten. Verder is het ook belangrijk dat het programma een aansprekende layout heeft. Bij de selectie van de AAT is het ook goed als het opensource is, omdat het makkelijk aangepast kan worden. Verder moet het de mogelijkheid bieden C++ programma's te beoordelen, omdat de studenten bij het experiment C++ programma's ontwikkelen. Er zijn AAT's die gemaakt zijn voor programmeerwedstrijden en AAT's die gemaakt zijn voor programmeercursussen. Bij deze selectie is het belangrijk dat de AAT gemaakt is voor programmeercursussen.
Dit levert dan de volgende criteria op:

- Uitkomsten van andere onderzoeken
- Onderzoeksinstellingen/universiteiten gebruiken het systeem
- GUI (Graphical User Interface) - Ziet het programma goed uit?
- Gemaakt voor programmeercursussen
- Opensourcesoftware
- Mogelijkheid om C++ programma's te beoordelen

Omdat ik de AAT in een programmeercursus wil gebruiken, is het belangrijk voor mijn experiment een AAT te kiezen die gemaakt is voor een programmeercursus. AAT's die gemaakt zijn voor programmeerwedstrijden hebben een andere opbouw dan AAT's die gebruikt worden voor cursussen. Bijvoorbeeld bestaat bij de cursus *Imperatief Programmeren 2* een opdracht uit verschillende kleinere opdrachten. Omdat het heel subjectief is of een AAT er mooi uitziet, wordt ervoor gekozen mijn eigen mening daarvoor te gebruiken.

3.2.1 Mooshak

Mooshak[1][9] is een tool die gemaakt is voor programmeerwedstrijden. Het is een opensource programma. Kenmerken van het systeem zijn volgens de website het automatisch beoordelen van code en het feit dat het voor meerdere wedstrijden gebruikt kan worden. In figuur 3.1 is de GUI van Mooshak te zien. Het is een GUI die er oud uitziet. Het is mogelijk met Mooshak



Figuur 3.1: GUI van Mooshak

C++-programma's te beoordelen. Mooshak wordt veel in andere onderzoeken gebruikt als een AAT([11, 1, 5]). In de sectie 'Theoretisch kader' is te zien, dat het niet bij ieder onderzoek het gewenste effect opgeleverd heeft.

3.2.2 Peach3

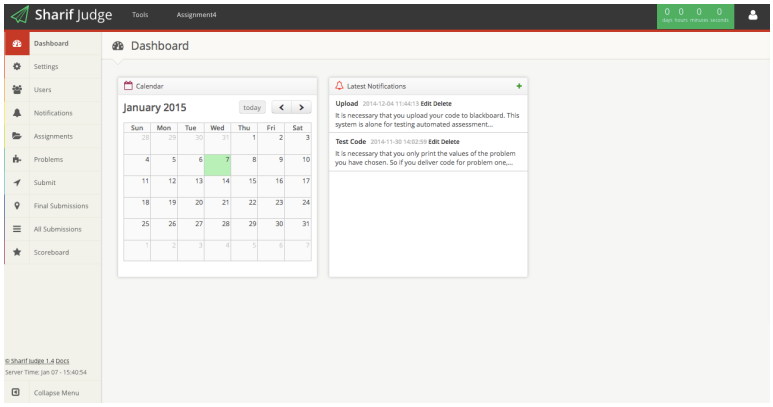
Peach3[2] is ontwikkeld door de Technische Universiteit Eindhoven. Het is een Assessment Tool die vooral voor programmeercursussen gemaakt is, maar ook gebruikt kan worden voor programmeerwedstrijden. De tool wordt op drie onderwijsinstellingen gebruikt (Eindhoven University of Technology, Fontys Hogescholen, ETH Zürich).
Om het systeem goed te evalueren werd met de installatieinstructies op de pagina, de tool geïnstalleerd. Omdat de versie verouderd was, bleek een installatie niet mogelijk. Daarna werd contact te hebben opgenomen met een van de ontwikkelaars en het lukte een nieuwere versie te krijgen en na drie weken een werkende versie van Peach3 te installeren.
Daarbij viel op dat het systeem recentelijk een nieuwe GUI had gekregen. Daaruit volgte dat op sommige plekken de nieuwe GUI gebruikt wordt en op ander momenten de oude. Na de installatie van Peach3 trad een groter probleem op: de automatische beoordeling is niet mogelijk. Daardoor is het niet mogelijk C++ programma's te beoordelen. De tool kan op dit moment

[1] *https : //mooshak.dcc.fc.up.pt*
[2] http://peach3.nl/trac/

alleen als een Assessment Tool gebruikt worden. Dit feit maakt de tool niet bruikbaar voor mijn onderzoek.

3.2.3 SharifJudge

SharifJudge is een opensource AAT die alleen voor programmeercursussen ontwikkeld is. De tool is ontwikkeld door Mohammad Javad Naderi in 2014[3]. Het is een tool die gebruikt kan worden voor de programmeertalen C, C++, Python2, Python3 en Java. Daarvoor moeten op de server de bijbehorende compilers geïnstalleerd worden.
In het algemeen hebben AAT's het probleem dat de programmacode van de studenten op de server draait. Daardoor is het bijvoorbeeld niet mogelijk om vooraf te weten of het malicious code bevat. Het gebruiken van een *sandbox* lost dit probleem op. De code van de studenten wordt in een omgeving gedraaid die geen rechten heeft voor bepaalde acties die noodzakelijk zijn om bijvoorbeeld files op de server te lezen waar men geen rechten voor heeft. Sharifjudge heeft de mogelijkheid Sandboxing te activeren.
Het systeem is snel te installeren en de instructies op de website zijn goed te volgen.
De GUI van het programma ziet er naar mijn mening goed uit. In figuur 3.2 is deze te zien. Bij SharifJudge bestaat een opgave uit verschillende kleinere opgaven. Dit is typisch voor opdrachten die bij cursussen als *Imperatief Programmeren 2* horen. Ik heb geen onderzoeken of onderwijsinstellingen



Figuur 3.2: GUI van SharifJudge

gevonden die dit programma gebruikt hebben.

3.2.4 Alternative tools

Er zijn vele programma's op het web te vinden, maar sommige daarvan zijn niet te gebruiken voor mijn experiment. Bijvoorbeeld het systeem *DomJu-*

[3]https://github.com/mjnaderi/Sharif-Judge

dge [4]. Bij dit systeem zijn veel problemen opgetreden bij het installatieproces. Het systeem CourseMaker[5]([6]) heeft alleen de mogelijkheid Java programma's te beoordelen en is dus niet toepasbaar op mijn experiment, omdat er in C++ geprogrammeerd wordt. De tool *BOSS*([7]) wordt in een ander onderzoek ([1]) genoemd. In dit onderzoek heb ik niet verder naar dit systeem gekeken, omdat [1] al Mooshak als AAT gekozen heeft in plaats van *BOSS*.

3.2.5 Keuze

	Mooshak	Peach3	SharifJudge
Opensourcesoftware	X	X	X
Gemaakt voor programmeercursussen		X	X
Uitkomsten van andere onderzoeken	X		
Mooie GUI (Graphical User Interface)			X
Gebruikt door andere onderzoeksinstellingen/universiteiten	(X)	X	
C++ programma's beoordelen	X		X

Tabel 3.1: Beoordeling AATs

Ik heb SharifJudge gekozen omdat het een tool is die voor programmeercurssussen gemaakt is, sandboxing heeft en omdat de GUI mooier is dan de GUI's van andere programma's. Mooshak heb ik niet gekozen omdat dit een tool is die voor wedstrijden gemaakt is en de GUI een oude layout heeft. Doordat Peach3 geen automatische beoordeling heeft, kan het in dit onderzoek niet gebruikt worden. In tabel 3.1 is de beoordeling samengevat.

3.3 Context

Het experiment vindt plaats in de cursus *Imperatief Programmeren 2*. Het idee van het experiment is dat studenten een AAT gebruiken en hun code op dit systeem inleveren. De beoordeling van dit systeem heeft bij dit experiment geen invloed op hun cijfer. De programmacode wordt nog steeds door een student-assistent nagekeken om te voorkomen dat het systeem fouten oplevert en de student daarvan last heeft.

3.4 Cursus

De cursus *Imperatief Programmeren 2* is een programmeercursus die gebruik maakt van de programmeertaal *C++*. De werkwijze van de cursus wordt met het volgende citaat van de studiegids beschreven:

[4] *http://www.domjudge.org*
[5] *http://www.coursemaker.co.uk/whatis.html*

"Practicumopgaven bereid je voor en werk je uit in koppels. Opgaven worden toegelicht in de vorm van een werkcollege. Er zijn practicumzalen gereserveerd waar je hulp kunt krijgen van student-assistenten. Zij beoordelen ook je uitwerkingen en voorzien deze van constructief commentaar."[6]

[6]http://www.studiegids.science.ru.nl/2014/socsci/prospectus/ki/course/32215/

Methode

In dit hoofdstuk wordt de opbouw van het experiment uitgelegd en vervolgens opgesomd wat er gedaan is om dit experiment voor te bereiden. Daarnaast wordt beschreven hoe de vragenlijst, het interview en de gebruiksstatistieken opgezet zijn en welke theoretische variabelen met dit onderzoek getest worden.

4.1 Theoretische variabelen

De theoretische variabelen uit de tabel 4.1 worden met een vragenlijst, een interview en de gebruiksstatistieken van SharifJudge beantwoord. Met behulp van deze theoretische variabelen is het mogelijk een antwoord op de vraagstelling te vinden. Daarbij probeert het interview de antwoorden op verschillende vragen van de vragenlijst te onderbouwen. Verder kan met de gebruiksstatistieken van SharifJudge getest worden of en hoe de studenten het programma gebruiken.

Theoretische variabele	Indicatoren	Metingen
Usability	System Usability Scale (SUS)	SUS resultaat
	Gevoel van studenten	Interview
	Bruikbaarheid	Antwoord op Q19,Q22,Q23,Q25, interview
Beïnvloeding gebruik	Gebruik t.b.v. testen/debuggen code	Antwoord op Q27
	Hoeveel submissies per probleem	Aantal submissies
Structureler werken	Vergelijking studenten Scoreboard	Antwoord op Q26, interview
	Tool helpt de studenten bij het leren	Antwoord op Q16,Q17,Q18
	Effect van het feedback van SharifJudge	Antwoord op Q20,Q21,Q24

Tabel 4.1: Theoretische variabelen

Iedere theoretische variabele heeft een aantal indicatoren. Met deze indicatoren wordt geprobeerd een antwoord op de theoretische variabelen te vinden. Voor iedere indicator is een bepaalde meting nodig om een antwoord te vinden. Deze metingen staan in de tabel in de kolom 'Metingen'.

Bijvoorbeeld wordt de theoretische variabele 'Beïnvloeding gebruik' opgesplitst in twee indicatoren:

- Gebruik t.b.v. testen/debuggen code

- Hoeveel submissies per probleem

Beide van deze indicatoren hebben een meting. De data die nodig zijn om de eerste indicator te beantwoorden is het antwoord op de vraag Q27(Tabel 5.5 en voor de tweede indicator is dit het aantal submissies per probleem.

4.2 Opbouw experiment

Het experiment wordt binnen de cursus *Imperatief Programmeren 2* gedraaid en maakt gebruik van het SharifJudge systeem. De cursus draait in het tweede kwartaal van het studiejaar 2014/2015. Er zijn in totaal 6 assignments die de studenten moeten maken. Uit deze zes assignments worden assignment drie en vier gekozen omdat deze assignments geschikt zijn voor het experiment en het qua tijdsplanning overeen kwam met de planning van dit onderzoek.
Het experiment wordt met een groep studenten uitgevoerd. Deze werken twee weken lang met SharifJudge en leveren de oplossingen van de opgaven op dit systeem in. De studenten hebben de oplossing ook op Blackboard ingeleverd. Bij de cursus *Imperatief Programmeren 2* wordt in koppels van twee gewerkt. Ieder koppel heeft een account op SharifJudge.
Na afloop van het experiment hebben de studenten een vragenlijst gekregen en een aantal teams is geïnterviewd. Daarnaast wordt met behulp van logging getest hoe de studenten het systeem gebruikt hebben.

4.2.1 Voorbereiding experiment

Zoals in het hoofdstuk Achtergrond verteld is, wordt bij dit experiment het SharifJudge systeem gebruikt. SharifJudge controleert de correctheid van het ingeleverde programma.
Er bestaan twee verschillende mogelijkheden dit met Sharifjudge te doen. Sharifjudge vergelijkt de output van de ingeleverde programmacode met de verwachte output voor de gegeven input. Het systeem laat het programma van de student met iedere testcase (input) draaien. De eerste mogelijkheid die SharifJudge biedt, is voor iedere testcase een input.txt en output.txt bestand aan te maken. Het programma wordt dan met iedere testcase uitgevoerd.
Bij de tweede mogelijkheid is er voor iedere opdracht een tester-file nodig. Dit bestand is een C++-bestand en heeft als invoer drie strings nodig. Iedere string representeert een pad naar een file. De eerste string verwijst naar het input.txt bestand; de tweede string verwijst naar het bijbehorende output.txt bestand. Het derde argument verwijst naar de output, die de code van de student gecreëerd heeft.
Om de tweede mogelijkheid uit te leggen, wordt in dit hoofdstuk gebruik

gemaakt van de volgende voorbeeldopdracht:

$$sum(n) = \begin{cases} 0 & \textbf{if } n = 0 \\ n + sum(n-1) & \textbf{if } n > 0 \end{cases}$$

De bedoeling van de tester is dat de useroutput met de gewenste output vergeleken wordt. Die tester rekent bijvoorbeeld het juiste resultaat voor het sum-probleem uit en vergelijkt dit met de uitkomst van het programma van de student. Als het resultaat goed is levert de mainfunctie een 0 op en anders een 1.
Voor complexere opdrachten zal met een tester gewerkt worden, omdat het anders onoverzichtelijk wordt als men voor ieder inputbestand het output bestand moet aanmaken. Verder is het overzichtelijker omdat er niet op tikfouten gelet moet worden bij de output bestanden. Daarnaast is het werken met een tester dynamischer. Het is bijvoorbeeld makkelijker een testcase te veranderen omdat alleen de input veranderd moet worden.
Een voorbeeld van een tester is in 4.1 te zien. In deze code wordt duidelijk waarvoor de tester een verwijzing naar het output bestand heeft. Het is mogelijk de twee mogelijkheden te combineren. Dat betekent dat het systeem gewoon een output-bestand vergelijkt met de useroutput als het output-bestand bestaat. Als het bestand niet bestaat, dan wordt de gewenste output met de tester uitgerekend en vergeleken.
In dit experiment is voor iedere opdracht een tester aangemaakt.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int calculateSum (int a){
        if(a == 0){
                return a;
        }else if(a >0){
                return a + calculateSum(a-1);
        }
}

int main(int argc, char const *argv[])
{
        ifstream test_in(argv[1]);
        ifstream test_out(argv[2]);
        ifstream user_out(argv[3]);

        int sum, user_output;
        user_out >> user_output;


        if ( test_out.good() ) {
                //if test's output file exists
                test_out >> sum;
```

```
        }else{
        int a;

                test_in >> a;
                sum = calculateSum(a);
        }

        if (sum == user_output)
                return 0;
        else
                return 1;

}
```

Listing 4.1: Voorbeeld testerbestand voor probleem 1 van assignment 3

4.3 Vragenlijst

De vragenlijst bestaat uit verschillende onderdelen. In deze sectie wordt uitgelegd waarvoor ieder onderdeel gebruikt wordt. Iedere vraag van de vragenlijst heeft als antwoordmogelijkheden de waarden van 1 t/m 5. Daarbij staat de waarde 5 voor 'helemaal mee eens' en de waarde 1 voor 'helemaal oneens'.

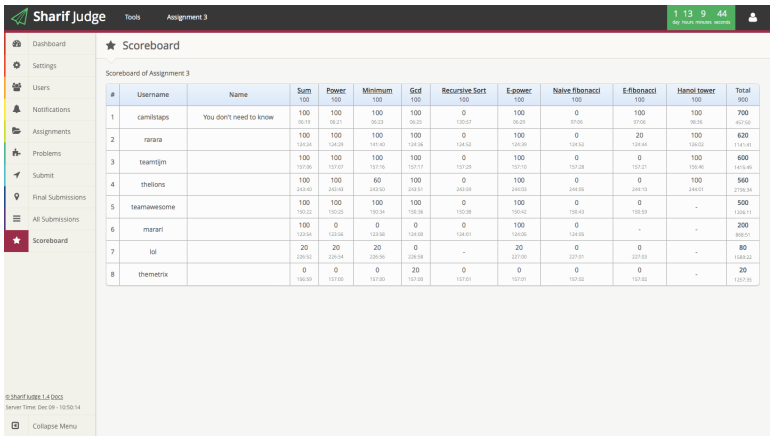
4.3.1 Usability

In dit onderzoek wordt de usability met de 'System Usability Scale'(SUS) getest ([3]). De SUS bestaat uit tien vragen die alleen gericht zijn op de usability van het systeem dat getest wordt. In [2] wordt het woord 'systeem' vervangen door 'product'. Omdat bij het experiment echt een systeem gebruikt wordt, wordt dit niet aangepast. Door het gebruiken van de System Usability Scale worden uit de antwoorden met behulp van het bijbehorende algoritme uitkomsten berekend. De resultaten van dit algoritme hebben waarden van 0 t/m 100 waarbij 100 voor een maximale Usability staat en 0 voor een helemaal slechte. Het algoritme wordt beschreven in het paper van John Brooke[3].

4.3.2 Scoreboard

In [1] was het resultaat dat de studenten het leuk vonden tegen elkaar te spelen. Dit kan alleen bereikt worden als de AAT een 'Scoreboard' heeft. Op dit Scoreboard staat voor iedere student per opdracht de score. Met 'tegen elkaar spelen' wordt hier bedoeld dat studenten proberen beter te zijn dan andere studenten. Dit onderzoek wil met behulp van twee vragen testen of de studenten het Scoreboard daarvoor gebruiken. In figuur 4.1 zie

je als voorbeeld de scoreboard van assignment 3.



Figuur 4.1: Scoreboard assignment 3

4.3.3 Het effect van AATs

In deze sectie wordt de vragenlijst van [11] besproken. Het idee is dat wij een aantal van de vragen overnemen.

Behulpzaamheid Met het eerste onderdeel, "Students' perception of Mooshak towards its helpfulness in learning", wordt getest of de studenten het gevoel hadden of en hoe het systeem de studenten helpt bij het leren. Het onderdeel bestond uit 6 vragen. Drie vragen worden in de vragenlijst van dit onderzoek hergebruikt (Tabel 5.5, vragen: Q16, Q17, Q18).
Deze vragen kunnen studenten beantwoorden na twee weken gewerkt te hebben met de AAT maar er zijn nog andere vragen die niet geschikt zijn voor dit onderzoek. Een van de niet gebruikte vragen was of de AAT geholpen heeft een betere programmeur te worden. Het is niet mogelijk dat de studenten dit kunnen beantwoorden na twee weken.

Bruikbaarheid "Effect of Mooshak's feedback in the tool's usefulness" was het derde onderdeel van de vragenlijst. Dit onderdeel was goed toe te passen op het experiment (Tabel 5.5, vragen: Q20, Q21, Q24).
Vraag 24 is ook van de vragenlijst overgenomen. Deze vraag had in de oorspronkelijke versie geen sectie. In dit onderzoek wordt de vraag aan deze sectie toegevoegd omdat vraag 24 ook met bruikbaarheid te maken heeft. De vragen zullen testen of de feedback van de tool handig is voor de studenten.

Het idee van AATs In het onderzoek van Rubio-Sanchez et al. wordt ook met de vragenlijst getest, hoe en of Mooshak een effect heeft in het vervolg. Daarbij testen de vragen bijvoorbeeld of de studenten het goed vinden als zij vanaf de eerste programeercursus met een AAT werken. De vragen 19, 22 en 25 van dit onderzoek zijn in mijn onderzoek hergebruikt en zullen een antwoord geven op dit gebied (Tabel 5.5, vragen: Q19, Q22, Q25).

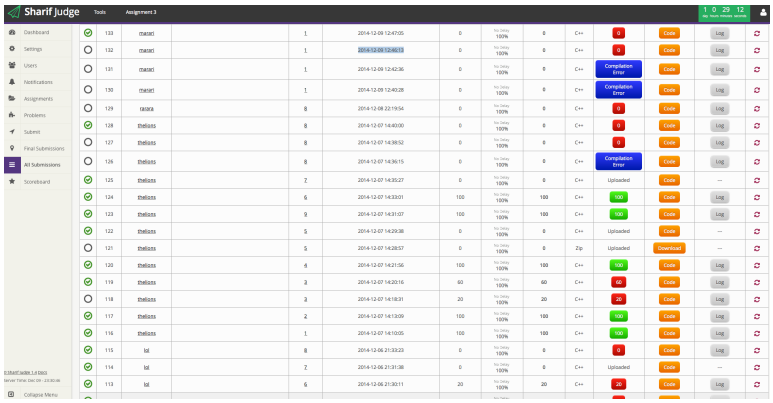
Neveneffect Het vierde onderdeel "Mooshak's effect op persistence"test of er ook neveneffecten optreden naast de verwachte effecten. Een effect zou bijvoorbeeld zijn dat de studenten na het gebruiken van een AAT het leuk zouden vinden om aan een programmeerwedstrijd mee te doen (Tabel 5.5, vraag: Q23).

4.3.4 Vergelijking met Blackboard

Het laatste onderdeel van de vragenlijst gaat over een vergelijking met Blackboard. Het is bedoeld voor iedere soort cursus en biedt geen mogelijkheid voor een automatische beoordeling van programmacode. Het doel van de vragen uit het hoofdstuk 'Comparison with Blackboard' is om te testen welk systeem de studenten beter vinden.
De laatste vraag is een open vraag waar de studenten kunnen aangeven welk systeem zij beter vinden en waarom zij dit denken.

4.4 Gebruiksstatistieken

Er bestaan verschillende mogelijkheden iets te observeren bij dit experiment. Het zou bijvoorbeeld mogelijk zijn om de studenten te observeren als zij met het systeem werken. Deze observatie wordt in dit onderzoek niet gebruikt, omdat het niet mogelijk is deze observatie uit te voeren, omdat de studenten niet verplicht zijn de opdrachten op de universiteit te maken.
Een analyse van gegevens die in dit onderzoek uitgevoerd wordt, test hoe de studenten hun code inleveren. SharifJudge heeft een mogelijkheid alle pogingen per student in te zien. Een screenshot van deze tabel laat figuur 4.2 zien. Uit deze analyse kan een aantal conclusies getrokken worden. Het is mogelijk te testen hoeveel pogingen een student gemiddeld gebruikt heeft om een goed resultaat te krijgen. Een andere mogelijkheid is te testen of de studenten vaak 'Compilation Errors' krijgen. Als dit het geval is, dan kan dit resultaat vergeleken worden met het resultaat van vraag 27(Tabel 5.5, vraag: Q27)



Figuur 4.2: Submissions assignment 3

4.5 Interview

De resultaten van de vragenlijsten worden geanalyseerd en vervolgens worden deze geïnterpreteerd. Om deze interpretaties te onderbouwen worden een aantal teams geïnterviewd.
Bij de interviews wordt iedere groep individueel geïnterviewd zodat de interpretaties van de lijsten per groep gedaan kunnen worden. Omdat de vragen van het interview aan de uitkomsten van de vragenlijst gekoppeld zijn, kunnen wij niet van tevoren weten welke vragen precies gesteld moeten worden. Het plan is dat het interview test of de interpretaties van de vragenlijsten kloppen. Een vraag van het interview test hoe de studenten de usability van het systeem vonden, een tweede test hoe de studenten het idee van AAT's vinden, een derde of de studenten het scoreboard gebruikt hebben om beter dan andere studenten te zijn en een vierde die test of een AAT goed zou zijn voor programmeer beginners. De andere vragen van het interview zijn gebaseerd op individuele uitkomsten van de vragenlijst. Daarbij worden de meest interessante interpretaties uit de vragenlijst expliciet aan de studenten gevraagd.

Resultaten

5.1 Algemene informatie

De tabel 5.1 laat zien hoeveel teams zich in totaal ingeschreven hadden voor dit experiment en hoeveel teams daadwerkelijk meegewerkt hebben (8 van 13 teams).

	aangemeldt	een keer ingelogd	first login == last login	totaal teams
Aantal teams	13	11	3	8

Tabel 5.1: Aantal teams

De volgende tabel (tabel 5.2) laat zien hoeveel teams bij iedere assignment iets ingeleverd hebben. Bij de eerste assignment van het experiment (Assignment 3) hebben 8 teams meegewerkt en bij de tweede assignment (Assignment 4) hebben 5 teams meegewerkt.

	Assignment 3	Assignment 4
Aantal teams	8	5

Tabel 5.2: Aantal teams per assignment

Iedere student had de mogelijkheid de vragenlijst in te vullen. Voor dertien teams zijn dit 26 studenten. Omdat één team uit een student bestond zijn het in totaal 25 studenten.
Van deze 25 studenten hebben twaalf de vragenlijst beantwoord waarvan 2 studenten nooit met SharifJudge gewerkt hebben. In totaal zijn er dus tien bruikbare resultaten van de vragenlijst.
Voor het interview worden twee teams gekozen, waarvan een team redelijk goede beoordelingen gekregen had van SharifJudge en een team dat problemen had met SharifJudge te werken.

5.2 Vragenlijst

Usability

Algemeen De System Usability Scale(SUS) bevat tien vragen. Uit deze tien vragen kan met behulp van het bijbehorende algoritme [3] een score berekend worden. Deze score kan minimaal 0 en maximaal 100 zijn. De uitkomsten van de SUS worden in tabel 5.3 weergegeven.

	System Usability Score (SUS)
Min	52,5
Max	87,5
Mediaan	68,75
Gemiddelde waarde	68,25

Tabel 5.3: System Usability Scale uitkomst

De resultaten op iedere vraag zijn in de tabel 5.5 te zien (Tabel 5.5, vragen: Q5-Q14). De gemiddelde waarde op de vraag of het systeem makkelijk te gebruiken is(Q7), is 3,6 en de mediaan 4,0. De vraag of het systeem snel te begrijpen (Q11) is, wordt gemiddeld met een 4,3 beantwoord en de mediaan op de vraag is 4,5.
De uitkomsten op de vragen Q6, Q10 en Q12 zijn relatief hoog. Vraag Q6 test of de studenten het systeem complex vinden, vraag Q10 of het systeem te veel inconsistenties bevat en Q12 of de studenten SharifJudge onhandig vinden. De gemiddelde waarden op de vragen Q6, Q10 en Q12 zijn 2,5, 3,11 en respectievelijk 2,8.

Bruikbaarheid In tabel 5.5 zijn de uitkomsten van de vragen Q19,Q22,Q23 en Q25 te zien. Het gemiddelde resultaat op de vraag of zij het idee van het systeem goed vinden(Q22) is 3,8. Verder is het gemiddelde resultaat op vraag Q19 3,3. De vraag of het systeem een tijdsverspilling is, hebben de studenten gemiddeld met een 2,4 beantwoord. Het gemiddelde antwoord op de vraag of de studenten na het gebruiken van SharifJudge aan een programmeercontest deelnemen, is 3,3.

Structureler werken

De resultaten van de vragenlijst laat zien, dat vele studenten het scoreboard gebruiken om te checken hoeveel punten andere teams gekregen hebben. Het gemiddelde antwoord op de vraag (Q26) is 2,7.
Een andere indicator van de theoretische variabele 'Programma als aansporing' is "Tool helpt de studenten bij het leren". De antwoorden op de vragen Q16, Q17 en Q18 zijn daarvoor de metingen. Vraag Q16 test of de student na het gebruiken van SharifJudge verantwoordelijker programmeert. Het

gemiddelde antwoord op deze vraag is 2,3. Het gemiddelde antwoord op Q17 is 2,2. Deze vraag test of de studenten na het experiment er bewust van zijn dat het belangrijk is dat de programmacode correct is. Verder test vraag Q18 of SharifJudge helpt bij het bepalen van hun programmeerkennis. Deze vraag is door de studenten gemiddeld met een 3,3 beantwoord.
De laatste indicator 'Effect van het feedback van SharifJudge' wordt met de antwoorden op de vragen Q20, Q21 en Q24 gemeten(Tabel 4.1). Op de vraag of de studenten het goed vinden dat SharifJudge direct feedback geeft (Q20) hebben de studenten gemiddeld met een 3,6 geantwoord. Vraag Q21 is door de studenten gemiddeld met een 2,3 beantwoord. De vraag test of zij motiveerd zijn de fouten in de programmacode te vinden als SharifJudge deze niet als goed beoordeelt. Of de studenten tevreden zijn als SharifJudge de programmacode als goed beoordeeld heeft, test vraag Q24. De studenten hebben op deze vraag gemiddeld met een 4,4 geantwoord.

Beïnvloeding gebruik

Het gemiddelde resultaat op de vraag(Q27), of de studenten hun code goed testen voordat zij deze inleveren is 3,3.

Vergelijking met Blackboard

De vragenlijst heeft ook een vergelijking met SharifJudge en Blackboard gedaan. Het gemiddelde antwoord op de vraag of het makkelijker is met SharifJudge te werken is 2,3 en op de vraag of de studenten liever met SharifJudge werken 3,1.

Opmerkingen van de studenten

Op de laatste vraag van de vragenlijst kunnen studenten aspecten noemen die zij positief of negatief vonden. Vele studenten hebben dit ook gedaan. Vaak hebben de studenten genoemd dat het systeem de programmacode niet goed beoordeeld heeft en zij vervolgens hun code moeten aanpassen zodat het werkt. Daar bovenop vonden zij het niet duidelijk hoe de programmacode ingeleverd moest worden en dat zij alleen de mainfunctie moesten aanpassen en niet voor iedere opgave een eigen bestand moesten aanmaken.

5.3 Interview

Usability

Algemeen Het team 'camilstaps' heeft bij de interview uitgelegd dat zij de usability van het systeem niet goed vinden. Verder laat het interview zien dat het team vindt dat er te veel inconsistenties binnen SharifJudge zijn waardoor het systeem lastig te gebruiken is. Het team 'lol' vindt dat

de usability van het systeem redelijk goed is. Zij hebben geen negatieve aspecten over de usability genoemd.

Bruikbaarheid Het resultaat uit de interviews laat zien, dat de studenten het idee van AATs goed en leuk vinden. Het team 'camilstaps' had veel programmeerervaring en het vindt dat het vooral voor studenten goed is die nog niet veel ervaring met programmeren hebben omdat zij dan direct response krijgen of hun programmacode goed of fout is. Het team lol had nog niet zo veel programmeerervaring en zij denken dat het een echt goed idee is. Verder denken zij dat het niet alleen voor beginners goed is, maar ook voor oudere jaars. Het team 'camilstaps' vond SharifJudge vooral een tijdsverspilling omdat het systeem inconsistenties bevat en een rare besturing heeft. Verder laten beide interviews zien dat de teams het idee van AATs goed vinden en het goed zou zijn dat het tool vanaf het begin gebruikt wordt.

Structureler werken

Het interview laat zien dat niet iedereen het gebruikt om betere resultaten te krijgen dan andere studenten, maar wel om te checken hoeveel punten de andere studenten behaald hebben. Het team 'camilstaps' heeft bijvoorbeeld het scoreboard ook gebruikt om beter dan andere studenten te zijn. Een ander resultaat uit het interview met het team 'camilstaps' is, dat het team denkt, dat een 'tegen elkaar spelen' kan onstaan als de hele cursus het systeem gebruikt en iedereen met de echte naam 'speelt'.

Opmerkingen van de studenten

Het team 'lol' heeft bij de open opmerkingen van de vragenlijst het volgende geschreven: 'If the feedback would be representative. We had 20% and got a sufficient mark from studentassistent.'. In het interview met dit team, heb ik gevraagd wat zij hiermee bedoelden.
Zij vertelden dat SharifJudge hun code als niet goed beoordeeld had maar de studentassistent het wel goed vond. In het interview hebben zij dan nog een keer uitgelegd gekregen hoe SharifJudge werkt. Daarna hebben zij begrepen hoe het werkt en dat zij een verkeerd beeld hadden van de werking.

5.4 Analyse van gegevens

Tool voor debugging Tabel 5.4 laat het aantal pogingen per assignment zien. Verder laat het zien hoeveel pogingen ervan goed zijn en hoeveel niet. Bij Assignment 3 zijn er in totaal 118 pogingen waarvan 30,5% een goede beoordeling(score = 100) gekregen hebben en 59,3% een slechte(score = 0). Een score van 0 betekent, dat de oplossingen voor iedere testcase fout waren en een score van 100 betekent dat de oplossingen voor iedere testcase goed

	Assignment 3	Assignment 4
Totaal	118	34
Score = 100	36	7
Score = 0	70	27
Andere score	12	0
(Score = 100)/Totaal	0,305	0,205
(Score = 0)/Totaal	0,593	0,794

Tabel 5.4: Pogingen per assignment

waren. Met 'Andere score' worden scores bedoelt tussen 0 - 100 waar niet alle oplossingen overeenkwamen met de juiste resultaten. Bij Assignment 3 trad dit in totaal 12 keer op. Het aantal pogingen bij assignment 4 is sterk afgezakt. In totaal zijn er 34 pogingen. 20,5% zijn daarvan goed beoordeeld en de rest slecht.

Gebruiksstatistieken De gebruiksstatistieken laten zien, dat studenten vaak iets ingeleverd hebben, SharifJudge de programmacode als slecht beoordeelt heeft (score = 0) en de studenten daarna niet een nieuwe, verbeterde versie ingeleverd hebben. Een analyse van de ingeleverde programmacode tijdens het experiment heeft aangetoond, dat vele studenten fouten maakten in het aanpassen van de programmacode waardoor SharifJudge het kan beoordelen. De studenten hebben daarom na het eerste assignment een mail gekregen waarin uitgelegd wordt hoe de code moet worden aangepast zodat SharifJudge het kan beoordelen. Verder stond in de mail een voorbeeld uit het eerste assignment zodat het duidelijker wordt wat verwacht wordt van de studenten. Het resultaat daarvan was, dat tijdens het draaien van het tweede assignment deze extra uitleg geen groot positief effect had op de code van de studenten.
Om te testen of studenten niet begrepen hebben hoe de mainfunctie moet worden aangepast zijn meerdere pogingen geanalyseerd. Het blijkt dat bij een deel van de pogingen alleen de mainfuntie niet klopte waarmee SharifJudge het programma kan beoordelen.
In tabel 5.4 is te zien dat het aantal pogingen veel kleiner geworden is. Verder valt in tabel 5.2 op dat bij het eerste assignment 8 teams meegedaan hebben en bij het tweede assignment alleen 5.

ID	Vraag	Mediaan	Average
Q5	I think that i would like to use this system frequently	3	3,2
Q6	I found the system unnecessarily complex	2,5	2,5
Q7	I thought the system was easy to use	4	3,6
Q8	I think that I would neet the support of a technical person to be able to use this system	1	1,4
Q9	I found the various functions in this system were well integrated	4	3,5
Q10	I thought there was too much inconsistency in this system	3	3,11
Q11	I would imagine that most people would learn to use this system very quickly	4,5	4,3
Q12	I found the system very awkward to use	2,5	2,8
Q13	I felt very confident using the system	4	3,8
Q14	I needed to learn a lot of things before I could get going with this system	1,5	1,6
Q15	I used the scoreboard to see how many points other students have	2	2,22
Q16	SharifJudge has forced me to program more responsibly	2	2,3
Q17	SharifJudge has made me more aware of the need to write correct code	2	2,2
Q18	SharifJudge helps to measure my current programming skills	3	3,3
Q19	It would have been useful to use SharifJudge from the first programming course	5	4,6
Q20	I value the fact that a tool like SharifJudge returns feedback in real time about the correction of my programs	4	3,6
Q21	If SharifJudge does not accept my code I feel motivated to find and fix the errors	2	2,3
Q22	In general, using SharifJudge has been a good idea	3,5	3,8
Q23	Knowing SharifJudge can motivate me to take part in a programming contest	3	3
Q24	I feel satisfied when my programs pass SharifJudge's tests	4	4,4
Q25	SharifJudge has been a waste of time	2,5	2,4
Q26	During the process of making the exercises I want to be better than other students at the scoreboard	2	2,7
Q27	Before I hand in an assignment I check my code thoroughly to verify that it does not contain errors	3,5	3,3
Q28	Its easier to work with Sharifjudge as with Blackboard	2,5	2,9
Q29	I would like to work with SharifJudge and not with Blackboard	3	3,1

Tabel 5.5: Vragenlijst

Conclusie en discussie

In dit hoofdstuk worden uit de resultaten van het voorgaande hoofdstuk de conclusies getrokken en met behulp van deze conclusies de onderzoeksvraag beantwoord. Verder wordt toegelicht wat er voor problemen zijn met deze conclusies.

6.1 Conclusies

In deze sectie worden op alle deelvragen de antwoorden gegeven en aan het einde de onderzoeksvraag beantwoordt.

Zijn Automated Assessment Tools bruikbaar voor programmeercursussen?

Algemeen De uitkomsten van de System Usability Scale laten zien dat SharifJudge gemiddeld een score van 68,25 behaald heeft. Dit is niet echt een slecht resultaat maar er zijn nog wel dingen die volgens de studenten verbeterd kunnen worden. Een conclusie die uit de antwoorden op de vragenlijst en het interview getrokken worden kan, is dat de besturing van het systeem soms raar is. De oorzaak daarvoor zou mogelijkerwijs zijn dat het systeem te vele inconsistenties bevat. Dit zou een verklaring zijn voor het gegeven dat sommige studenten het systeem onhandig vinden. De antwoorden op de vragen Q7 en Q11 tonen aan dat de AAT niet moeilijk te gebruiken is en het voor nieuwe gebruikers snel te begrijpen is.

Bruikbaarheid Uit de resultaten kan de conclusie getrokken worden dat de studenten het idee van AAT's goed vinden en denken dat het goed zou zijn, als zij vanaf het begin van de studie daarmee zouden kunnen werken. Een conclusie zou dus zijn dat de studenten vanaf de eerste programmeercursus met een AAT moeten werken.
Verder kan de conclusie getrokken worden dat de studenten door het gebruiken van SharifJudge eventueel bij programmeerwedstrijden meedoen. Het gebruik van een AAT zou mogelijkerwijs het effect opleveren dat studenten deelnemen aan een programmeerwedstrijd.

De antwoord op de deelvraag: *Zijn Automated Assessment Tools bruikbaar voor programmeercursussen?* is dat de gekozen AAT, SharifJudge, volgens het onderzoek bruikbaar voor een programmeercursus is omdat studenten graag met een AAT zouden werken en de usability van de tool ook redelijk goed was.

Werken studenten met een Automated Assessment Tool structureler?

In het algemeen laten de resultaten zien dat de studenten de AAT als aansporing zien. Het scoreboard is niet voor iedereen een aansporing maar voor sommige van de studenten wel. Dit laten vooral de interviews zien.
Verder is een conclusie, dat het systeem de studenten niet echt geholpen heeft verantwoordelijker te programmeren en de studenten te helpen correcte programmacode te schrijven.
De directe feedback van SharifJudge vinden de studenten goed en zij zijn helemaal tevreden als SharifJudge hun code als goed beoordeelt. De studenten zijn niet gemotiveerd de fouten te vinden als ShairfJudge hun programmacode als slecht beoordeelt. Een verklaring daarvoor zou zijn, dat het voor de studenten frustrerend was dat hun programmacode op hun eigen machine gewerkt heeft en SharifJudge het als slecht beoordeeld heeft.
Verder heeft de analyse van gegevens het resultaat laten zien, dat de studenten niet helemaal begrepen hebben hoe het systeem werkt en hoe de programmacode op het systeem ingeleverd moet worden.
Een andere conclusie is dat verschillende studenten iets voor assignment 3 ingeleverd hebben en voor assignment 4 niet meer, omdat zij geen positieve beoordeling gekregen hadden. Een verklaring daarvoor zou zijn dat de studenten denken dat SharifJudge niet goed werkt. Zij zoeken de fout niet bij zichzelf. Deze verklaring volgt uit het feit dat deze studenten één keer iets ingeleverd hebben en als SharifJudge deze programmacode met een score van nul beoordeeld heeft, zij geen tweede poging gedaan hebben. Als het experiment herhaald wordt met de hele groep, dan dient er wel vooraf voldoende uitleg te worden gegeven over de werking van SharifJudge.

In het algemeen kan men zeggen dat de studenten door het gebruiken van een AAT structureler werken. Zij schrijven de programmacode en leveren deze ter beoordeling in. De studenten krijgen direct feedback en kunnen zien wat goed en fout ging en als nodig de code aanpassen. De studenten vinden het verder goed met een AAT te werken en zijn helemaal tevreden als de AAT de programmacode als goed beoordeelt heeft.

Beïnvloedt de Automated Assessment Tool de werkwijze van een student?

Uit de resultaten van tabel 5.4 kan men de conclusie trekken, dat de studenten vaak niet werkende programmacode ingeleverd hebben. Dit kan mogelijkerwijs een indicator zijn dat de studenten de tool voor debugging gebruiken. In verband met de algemene conclusie zou het ook kunnen dat de studenten werkende programmacode ingeleverd hebben, maar niet met de juiste opmaak. Hiervoor zou ook het relatief hoge antwoord op vraag Q27 spreken(3,3), waarmee studenten aangeven of zij de programmacode goed testen voordat zij deze inleveren.

In verband met de algemene conclusie is het moeilijk van een beïnvloeding te spreken. Er kan wel de conclusie getrokken worden dat de studenten niet alleen de tool gebruiken om de programmacode te testen. Een verklaring ervoor is dat de studenten vaak dezelfde programmacode ingeleverd hebben omdat zij wisten dat deze werkt.

Welke aspecten zijn bepalend voor de succesvolle inzet van een Automated Assessment Tool?

Het onderzoek laat zien dat de studenten het tool bruikbaar vinden, maar het blijkt noodzakelijk te zijn dat de studenten een nog betere uitleg nodig hebben om het principe van een AAT te begrijpen. Dit bepaalt heel sterk het succes van het gebruik van een AAT. Verder is het ook belangrijk dat de studenten gemotiveerd zijn of zelfs verplicht zijn de AAT te gebruiken. Als deze twee indicatoren (begrip, motivatie) goed gedaan worden zou het een positief resultaat opleveren als een AAT binnen een programmeercursus gebruikt wordt.

6.2 Discussie

Het is moeilijk een conclusie te trekken, omdat bij het experiment te weinig studenten meegedaan hebben. Van de circa 200 studenten die de cursus gevolgd hebben, hebben slechts 25 studenten zich aangemeld voor het experiment. Daaruit volgt, dat niet veel studenten het oude proces slecht vinden of daar problemen mee hebben. Maar het zegt ook niet dat de studenten het nieuwe proces bij voorbaat slechter vinden. Het zou aan te bevelen zijn dit experiment verplicht te stellen voor alle deelnemers aan de cursus. Daardoor moeten alle studenten de AAT gebruiken.

Bibliografie

[1] José Luis Fernández Alemán. Automated assessment in a programming tools course. *Education, IEEE Transactions on*, 54(4):576–581, 2011.

[2] Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[3] John Brooke. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[4] Richard M Felder, Gary N Felder, and E Jacquelin Dietz. A longitudinal study of engineering student performance and retention. v. comparisons with traditionally-taught students. *Journal of Engineering Education*, 87(4):469–480, 1998.

[5] Ginés Gárcia-Mateos and José Luis Fernández-Alemán. A course on algorithms and data structures using on-line judging. In *ACM SIGCSE Bulletin*, volume 41, pages 45–49. ACM, 2009.

[6] Colin Higgins, Tarek Hegazy, Pavlos Symeonidis, and Athanasios Tsintsifas. The coursemarker cba system: Improvements over ceilidh. *Education and Information Technologies*, 8(3):287–304, 2003.

[7] Mike Joy, Nathan Griffiths, and Russell Boyatt. The boss online submission and assessment system. *Journal on Educational Resources in Computing (JERIC)*, 5(3):2, 2005.

[8] Carsten Kleiner, Christopher Tebbe, and Felix Heine. Automated grading and tutoring of sql statements to improve student learning. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, pages 161–168. ACM, 2013.

[9] José Paulo Leal and Fernando Silva. Mooshak: a web-based multisite programming contest system. *Software: Practice and Experience*, 33(6):567–581, 2003.

[10] Luisa M Regueras, Elena Verdú, María F Muñoz, María A Pérez, Juan Pablo de Castro, and María Jesús Verdú. Effects of competitive e-learning tools on higher education students: a case study. *Education, IEEE Transactions on*, 52(2):279–285, 2009.

[11] Manuel Rubio-Sanchez, Päivi Kinnunen, Cristóbal Pareja-Flores, and J Angel Velázquez-Iturbide. Lessons learned from using the automated assessment tool "mooshak". In *Computers in Education (SIIE), 2012 International Symposium on*, pages 1–6. IEEE, 2012.

[12] Brenda Timmerman and Robert Lingard. Assessment of active learning with upper division computer science students. In *Frontiers in Education, 2003. FIE 2003 33rd Annual*, volume 3, pages S1D–7. IEEE, 2003.

[13] Elena Verdú, LM Regueras, and MJ Verdú. Active e-learning in university courses: Quest environment for self-managed training. *2001 Education Odyssey: Continuing the journey through adaptation and innovation, Open and Distance Learning Association of Australia*, 2001:147–148, 2006.