

Pair programming and underrepresented groups in Information Technology in South Africa

D Litlhakanyane



orcid.org/0000-0002-1268-2200

Dissertation accepted in partial fulfilment of the requirements for the degree *Master of Science in Computer Science* at the North-West University

Supervisor: Dr JA Liebenberg

Graduation May 2022

25064576

ABSTRACT

It is troubling that in a nation where the lion's share of the population is made up of Black South Africans, they are the minority within the IT industry. Furthermore, the gender inequality in IT remains a matter of concern where females remain the minority in IT classes and the workplace. It is widely known that the saying "two heads work better than one" can be applied in all aspects of work and life. Programming has become part of curriculums in classrooms around the globe. The aim of this study was to examine the impact that pair programming has on shaping the experiences and perceptions of underrepresented minority students in Information Technology (IT) in South Africa. There are limited mixed-method studies in South Africa that investigated the factors that impact the experiences of students using pair programming as an educational tool. The study included a total of 284 first-year BSc. in IT students that were from different ethnic groups, cultures, genders and socio-economic backgrounds. The students were taking the User Interface programming module at the Potchefstroom Campus of the North-West University (NWU) in South Africa. The participants for this research study included students who had experience with IT subjects in high school, in addition to students who were completely new to programming. The students were divided into pairs, and the students were given the opportunity to pick a desired programming partner for the duration of the semester. At the end of the semester, the students received a questionnaire that was developed, based on the Unified Theory of Acceptance and Use of Technology (UTAUT2) model to better understand the students' intentions to use pair programming in an IT course. The findings from the quantitative and qualitative data suggested that the group of students had a positive experience with pair programming. However, the underrepresented minority students had an even more positive learning experience with pair programming. The female students (White and Black) and male black students perceived pair programming in an exceedingly positive light, as it improved their overall programming skills, critical thinking and social skills. It can be recommended that with timely training and adequate implementation of the pair programming principles, all students, but indeed more so the underrepresented minority students in a programming course can reap the benefits of the use of pair programming. This approach may attract and increase the retention rate of underrepresented groups in IT and encourage them to further pursue IT careers in South Africa.

Keywords: pair programming, Information Technology, minorities, underrepresentation, South Africa, UTAUT.

ACKNOWLEDGEMENTS

I have received a great deal of support and assistance throughout the writing of this dissertation. Foremost, I would like to express my sincere gratitude to my supervisor, Dr Liebenberg, for the continuous support of my study and research. Her expertise was invaluable in formulating the research questions, as well as methodology. Her patience, immense knowledge and insightful feedback pushed to sharpen my way of thinking and helped me a great deal with the research and writing process of this dissertation. I could have not asked for a better supervisor and mentor for my Master's Degree study.

Besides my supervisor, I must also thank Dr Fourie from the North-West University's Statistical Consultation Services. You provided me with the results required to complete my dissertation. Your great statistical analysis throughout helped me better understand the research that I was doing. I am also grateful for the participants of this study; thank you for your time and feedback.

Last but not the least, I would like to thank my parents, Mr. and Mrs. Litlhakanyane, for their helpful advice, unwavering support and sympathetic ear. My completion of this study could not have been accomplished without you; my heartfelt thanks.

TABLE OF CONTENTS

ABSTRACT	I
ACKNOWLEDGEMENTS	II
LIST OF FIGURES.....	X
LIST OF TABLES	XI
LIST OF ABBREVIATIONS	XIII
CHAPTER 1 ORIENTATION.....	1
1.1 BACKGROUND AND PROBLEM STATEMENT	1
1.2 TERMINOLOGY CLARIFICATION	8
1.3 LITERATURE REVIEW.....	8
1.3.1 MINORITY GROUPS IN IT	9
1.3.2 PAIR PROGRAMMING	10
1.3.3 UNIFIED THEORY OF ACCEPTANCE AND USE OF TECHNOLOGY	11
1.4 AIM OF THE RESEARCH.....	12
1.4.1 OBJECTIVES OF THE RESEARCH STUDY	12
1.5 RESEARCH DESIGN AND METHODOLOGY	12
1.5.1 RESEARCH PARADIGM	12
1.5.2 RESEARCH DESIGN	13
1.5.3 PARTICIPANTS.....	13
1.5.4 DATA COLLECTION METHOD	14

1.5.5 DATA ANALYSIS	14
1.5.6 ETHICAL CLEARANCE	15
1.6 DISSERTATION STRUCTURE	16
1.6.1 CHAPTER 1: ORIENTATION.....	16
1.6.2 CHAPTER 2: MINORITIES IN IT	16
1.6.3 CHAPTER 3: PAIR PROGRAMMING	16
1.6.4 CHAPTER 4: RESEARCH METHOD	16
1.6.5 CHAPTER 5: RESULTS	17
1.6.6 CHAPTER 6: CONCLUSION AND RECOMMENDATIONS	17
1.6.6.1 STUDY CONTRIBUTIONS	17
1.6.6.2 STUDY LIMITATIONS	17
CHAPTER 2 MINORITIES IN INFORMATION TECHNOLOGY	18
2.1 INTRODUCTION	18
2.2 HISTORICAL VIEW OF STEM IN SOUTH AFRICA.....	18
2.3 THE DIGITAL DIVIDE	21
2.4 UNDERREPRESENTATION OF MINORITIES IN CS AND IT	24
2.5 WOMEN IN IT	26
2.6 CLOSING THE DIGITAL DIVIDE	28
2.6.1 CLOSING THE DIGITAL DIVIDE WITH PAIR PROGRAMMING.....	28
2.6.2 CODING INITIATIVES IN SOUTH AFRICA	29
2.7 SUMMARY	30

CHAPTER 3 PAIR PROGRAMMING	31
3.1 INTRODUCTION	31
3.2 PAIR PROGRAMMING IN THE INDUSTRY	31
3.2.1 DEVELOPMENT OF SOFTWARE	31
3.2.2 AGILE SOFTWARE DEVELOPMENT METHODOLOGIES	32
3.2.3 EXTREME PROGRAMMING EXPERIENCES IN THE INDUSTRY	33
3.2.4 PAIR PROGRAMMING AND EXTREME PROGRAMMING	35
3.3 PAIR PROGRAMMING AND THE CLASSROOM	37
3.3.1 INTRODUCTORY PROGRAMMING COURSES	37
3.3.2 A LOOK AT PAIR PROGRAMMING	39
3.3.3 IMPLEMENTATION AND USE OF PAIR PROGRAMMING	41
3.3.4 PAIR PROGRAMMING AND COMMUNICATION	43
3.3.5 GUIDELINES FOR EDUCATORS IN THE PAIR PROGRAMMING CLASS	44
3.4 CHALLENGES OF PAIR PROGRAMMING	46
3.5 THE ADVANTAGES OF PAIR PROGRAMMING	47
3.5.1 TIME	47
3.5.2 SOFTWARE QUALITY	48
3.5.3 TRANSFER OF KNOWLEDGE.....	49
3.5.4 PAIR PRESSURE.....	49
3.5.5 SATISFACTION.....	50
3.5.6 CONFIDENCE	50
3.5.7 PERCEPTION.....	51

3.5.8 WORKLOAD REDUCTION	52
3.6 THE DISADVANTAGES OF PAIR PROGRAMMING	52
3.6.1 COMMUNICATION	53
3.6.2 INCOMPATIBLE PAIRS.....	53
3.6.3 PERSONALITY TRAITS	54
3.6.4 ENVIRONMENT AND SCHEDULING	55
3.6.5 TRAINING AND MENTORING.....	55
3.6.6 PAIR PROGRAMMING AND NON-COMPUTER SCIENCE STUDENTS.....	56
3.7 SUMMARY	57
CHAPTER 4 RESEARCH METHODOLOGY	58
4.1 INTRODUCTION	58
4.2 AIM OF THE RESEARCH.....	58
4.2.1 OBJECTIVES OF THE RESEARCH STUDY	58
4.3 RESEARCH PARADIGM, DESIGN AND METHODOLOGY	58
4.3.1 RESEARCH PARADIGM	58
4.3.2 RESEARCH DESIGN	61
4.3.3 PARTICIPANTS.....	65
4.3.4 PAIR PROGRAMMING TRAINING	66
4.3.5 DATA COLLECTION METHODS AND INSTRUMENTS	68
4.3.5.1 INSTRUMENT	68
4.3.5.2 UTAUT.....	69
4.3.5.3 DATA COLLECTION METHOD	71

4.3.5.4 FACTOR ANALYSIS.....	75
4.3.6 DATA ANALYSIS.....	79
4.4 SUMMARY.....	80
 CHAPTER 5 RESULTS AND DISCUSSION.....	 81
5.1 INTRODUCTION.....	81
5.2 DESCRIPTIVE STATISTICS.....	81
5.3 ADVANTAGES OF PAIR PROGRAMMING	89
5.3.1 FAST AND EFFICIENT	90
5.3.2 SOCIAL SKILLS AND COMMUNICATION.....	91
5.3.3 HELPING AND SHARING KNOWLEDGE.....	91
5.3.4 PROBLEM SOLVING AND DIFFERENT PERSPECTIVES/SOLUTIONS	92
5.3.5 TEAMWORK.....	92
5.3.6 ERROR DETECTION/EASE OF USE	93
5.3.7 CONFIDENCE	93
5.3.8 NONE.....	94
5.3.9 WORKLOAD.....	94
5.4 DISADVANTAGES OF PAIR PROGRAMMING	95
5.4.1 CONFLICT	95
5.4.2 TIME-CONSUMING	96
5.4.3 INCOMPATIBLE PERSONALITIES	97
5.4.4 SKILL INCOMPATIBILITY	98
5.4.5 LACK OF KNOWLEDGE AND UNDERSTANDING	99

5.4.6 ABSENCE/TARDINESS	100
5.4.7 COMMUNICATION/LANGUAGE BARRIER.....	100
5.4.8 ANXIETY	101
5.5 DIFFERENCES AMONG STUDENTS.....	101
5.5.1 GENDER	103
5.5.2 ETHNICITY	103
5.5.3 COMPUTER ACCESS SINCE GRADE 1.....	104
5.5.4 INTERNET ACCESS SINCE GRADE 1	104
5.5.5 IT AS A HIGH SCHOOL SUBJECT.....	105
5.5.6 ACADEMIC PERFORMANCE	106
5.6 RELATIONSHIPS BETWEEN RACE AND GENDER AND THE UTAUT2 FACTORS ...	107
5.6.1 PERFORMANCE EXPECTANCY	108
5.6.2 HEDONIC MOTIVATION	108
5.6.3 EFFORT EXPECTANCY	109
5.6.4 SOCIAL INFLUENCE.....	109
5.6.5 HABIT.....	110
5.6.6 FACILITATING CONDITIONS	111
5.6.7 BEHAVIOURAL INTENTION	111
5.6.8 PAIR COMPATIBILITY	112
5.6.9 TEAMWORK.....	112
5.6.10 PAIR PROGRAMMING OVER SOLO PROGRAMMING.....	113
5.7 CONTRADICTORY CONCLUSIONS.....	114

5.8 SUMMARY	115
CHAPTER 6 CONCLUSION AND RECOMMENDATIONS.....	116
6.1 SYNOPSIS OF THIS STUDY	116
6.2.1 SYNOPSIS OF THE LITERATURE STUDY.....	117
6.2.1.1 UNDERREPRESENTED GROUPS IN IT	117
6.2.1.2 PAIR PROGRAMMING	118
6.2.2 DISCUSSION OF THE FINDINGS OF THE EMPIRICAL STUDY	120
6.3 STUDY CONTRIBUTIONS.....	126
6.4 STUDY LIMITATIONS	126
6.5 CONCLUSION AND RECOMMENDATIONS.....	127
6.5.1 RECOMMENDATIONS ARISING FROM THIS STUDY	128
6.5.2 RECOMMENDATIONS FOR FURTHER RESEARCH	129
BIBLIOGRAPHY	130
APPENDIX A – ETHICS APPROVAL LETTER	150
APPENDIX B – LANGUAGE EDITING CERITFICATE.....	151
APPENDIX C – INFORMED CONSENT	152
APPENDIX D – TURNITIN DIGITAL RECEIPT.....	153

LIST OF FIGURES

Figure 1.1 High Demand Job Sectors in South Africa (BusinessTech, 2019b).....	3
Figure 1.2 High Supply Job Sectors in South Africa (BusinessTech, 2019b)	3
Figure 1.3 Unemployment Rate in South Africa.....	4
Figure 1.4 Poverty Rate in South Africa.....	5
Figure 1.5 Underrepresented Women in High-Level Managerial Roles	6
Figure 1.6 Unemployment Rate of South Africa by Population Group and Gender	7
Figure 1.7 UTAUT and Suggested Future Research Directions.....	11
Figure 4.1 Sequential Explanatory Design.....	63
Figure 4.2 The Original UTAUT Model (Venkatesh <i>et al.</i> , 2003)	70
Figure 4.3 The UTAUT2 Model with Moderators and Additional Constructs (Venkatesh <i>et al.</i> , 2012).....	71
Figure 5.1 Frequencies of Single Items	85
Figure 5.2 Frequency of Selections of Constructs	89
Figure 5.3 Gender Distribution Chart.....	102
Figure 5.4 Ethnicity Distribution Chart	102

LIST OF TABLES

Table 1.1 – Fitch Solution Industry Report South African IT Forecast May 2021	1
Table 4.1 – An Adapted Sequential Explanatory Design used for this Research Study	64
Table 4.2 – Profile of Students (n=284)	65
Table 4.3 – A Five-Point Likert Scale.....	72
Table 4.4 – Questionnaire	74
Table 4.5 – Kaiser-Meyer-Olkin Measure and Bartlett’s Test of Sphericity	76
Table 4.6 – Total Variance Explained	76
Table 4.7 – Communalities.....	77
Table 4.8 – Pattern Matrix	78
Table 4.9 – The Reliability Coefficients of the Factors	79
Table 5.1 – Questions, Factors and Measure of Frequency (n=284)	83
Table 5.2 – Factors, Questions and Measures of Tendency and Variation (n=284).....	86
Table 5.3 – Differences based on Gender	103
Table 5.4 – Differences based on Ethnicity.....	104
Table 5.5 – Differences based on Computer Access since Grade 1	104
Table 5.6 – Differences based on Computer Access since Grade 1	105
Table 5.7 – Differences based on IT as a High School Subject	106
Table 5.8 – Differences based on Academic Performance	107
Table 5.9 ANOVA – Performance Expectancy	108
Table 5.10 ANOVA – Hedonic Motivation.....	109

Table 5.11 ANOVA – Effort Expectancy	109
Table 5.12 ANOVA – Social Influence	110
Table 5.13 ANOVA – Habit.....	110
Table 5.14 ANOVA – Facilitating Conditions	111
Table 5.15 ANOVA – Behavioural Intention.....	112
Table 5.16 ANOVA – Pair Compatibility	112
Table 5.17 ANOVA – Teamwork	113
Table 5.18 ANOVA – Pair programming over Solo programming	114

LIST OF ABBREVIATIONS

4IR - Fourth Industrial Revolution

ANOVA – Analysis of Variance

ASD - Adaptive Software Development

AWS - Amazon Web Services

BBBEE - Broad-Based Black Economic Empowerment

BI - Behavioural Intention

C3 - Chrysler Comprehensive Compensation

CAT - Computer Applications Technology

CFA - Confirmatory Factor Analysis

COVID19 - Coronavirus Disease of 2019

CS - Computer Science

DSDM - Dynamic System Development Method

EE - Effort Expectancy

EFA - Exploratory Factor Analysis

FC - Facilitating Conditions

FDD - Feature-driven Development

FNASREC - Faculty of Natural and Agricultural Sciences Research Ethics Committee

GDP - Gross Domestic Product

H - Habit

HM - Hedonic Motivation

IaaS - Infrastructure-as-a-Service

ICT - Information and Communications Technology

IS - Information Systems

IT - Information Technology

JMMR - Journal of Mixed Methods Research

JSMMR - Japan Society for Mixed Methods Research

LSD - Lean Software Development

MBTI - Myers-Briggs Type Indicator

MMF - Methodes Mixtes Francophonie

MMIRA - Mixed Methods International Research Association

NWU - North-West University

PE - Performance Expectancy

SA – South Africa

SaaS - Software-as-a-Service

SAP - Systems Application Protocol

SAT - Scholastic Assessment Test

SDLC - Software Development Life Cycle

SET - Science, Engineering and Technology

SI – Social Influence

SPSS - Statistical Package for the Social Sciences

STEM - Science, Technology, Engineering and Mathematics

TIMSS - Trends in International Mathematics and Science Study

UB - Use Behaviour

US – United States of America

UTAUT - Unified Theory of Acceptance and Use of Technology

VCAPS - Vehicle Costing and Profit System

XP – Extreme Programming

CHAPTER 1 ORIENTATION

1.1 BACKGROUND AND PROBLEM STATEMENT

South Africa is currently in the middle of the emerging Fourth Industrial Revolution. In response, the South African National Department of Education has proposed programming (coding and robotics) as a new subject in Grades R-9 in order to prepare the youth of this country for the skills and jobs of the future (Govender & Govender, 2021). Information Technology (IT) refers to hardware and software that is used to store, retrieve and process all forms of electronic data and in the context of commercial use, it includes computer technology and telecommunications (Castagna, 2021). According to The Information Technology Association of South Africa, in spite of South Africa's political and regulatory environment the country is not yet leveraging the potential benefits associated with the Information and Communications Technology (ICT) sector (The Department of Communications and Digital Technologies, 2014). This is mainly caused by the great lack of basic skills in large sections of the country's population, coupled with the high cost of accessing ICT services.

According to the United States of America (US) International Trade Administration (2021), the South African Information Technology sector has the largest ICT market in all of Africa, displaying its leadership in areas, such as the mobile software development field, software security and also electronic banking services. (Table 1.1 below reflects the estimated market overview and trade data of May 2021 Unit: \$billion).

Table 1.1 – Fitch Solution Industry Report South African IT Forecast May 2021

South African IT Industry Report	2019	2020 (estimated)	2021 (forecast)	2022 (forecast)
Total Market Size	6.38	5.53	5.53	6.75
Hardware Sales	2.09	1.50	1.50	1.76
Software Sales	1.86	1.34	1.34	1.58
Service Sales	2.86	2.68	2.99	3.40
Exchange Rate: \$1	14.58	16.44		

South Africa's ICT sector is an important contributor to the country's Gross Domestic Product (GDP) with several international corporates, such as IBM, Unisys, Microsoft, Intel, Systems Application Protocol, Dell, Novell and Compaq operating subsidiaries from South Africa. In recent

years, the international corporations supplied the country with new fixed and wireless telecommunications networks stabled across the continent (International Trade Administration, 2021; Mzekandaba, 2020). The South African Government is now focusing on increasing ICT usage in this sector in order to facilitate socio-economic justice and inclusion in preparation for the Fourth Industrial Revolution (4IR) (Govender & Govender, 2021). ICT has seen a reduction in expenditure to counteract the increasing debt caused by the COVID19 pandemic.

There are opportunities in South Africa for organisations looking to utilise Software-as-a-Service (SaaS) and Infrastructure-as-a-Service (IaaS) from cloud computing with it gaining even more significance due to its improved bandwidth availability, security and low cost. Local internet providers, such as MTN, Vodacom, Neotel and BCX have implemented some form of cloud computing since 2016, making them competitors in this market alongside Huawei, Amazon Web Services (AWS), Microsoft and Oracle (International Trade Administration, 2021). The increase in interconnectivity in South Africa has redefined the relationships between consumers and enterprises, as well as between citizens and the government and as a result, new opportunities and challenges for the IT sector with regard to privacy, security, cybercrimes and access to personal data and information have not been introduced (The Department of Communications and Digital Technologies, 2014).

E-commerce has shown growth with the number of South African online shoppers increasing year-on-year and in 2020 retail sales increased by 66% due to the COVID19 pandemic and its consequent lockdown regulations introduced many people to online shopping (International Trade Administration, 2021). However, on a global scale, SA only makes up around 1% of total market for consumer goods which is nowhere near the online retailers of the rest of the world who make up 10-12% of the total retail. It is a concern because in South Africa only 41% of households has at least one member who has access to the internet at home and in general only 10% of households has access to the internet at home. South Africans have adopted mobile internet and the use thereof has been widespread in recent years which as a result, provides e-commerce with a golden opportunity on mobile devices with the mobile penetration rate at 165% in 2018 (International Trade Administration, 2021).

South Africa requires skilled IT employees in order to build a digital economy (Chutel, 2016). Currently, there is a shortage of IT specialists in the country, and IT is rated as the most positive and promising employment prospect in SA (Mafilika, 2021). Therefore, there is a strong market

demand for skilled IT employees, in particular software developers (MyBroadband, 2014). Software development is the most sought-after skill set with a fast growing market demand of over 25% in comparison to other jobs in the same category (BusinessTech, 2019b). The number of job opportunities in this field increased by 22% from January to February 2021, and the demand in the South African job market remains significant (McKane, 2021). As stated by BusinessTech (2019b), the top five sectors that are promising prospects in South Africa are as follows: Information Technology, Business and Management, Finance, Sales, and Admin, Office and Support. (Figure 1.1 below displays sectors in terms of major demand in South Africa and Figure 1.2 below displays sectors in terms of major supply in South Africa).

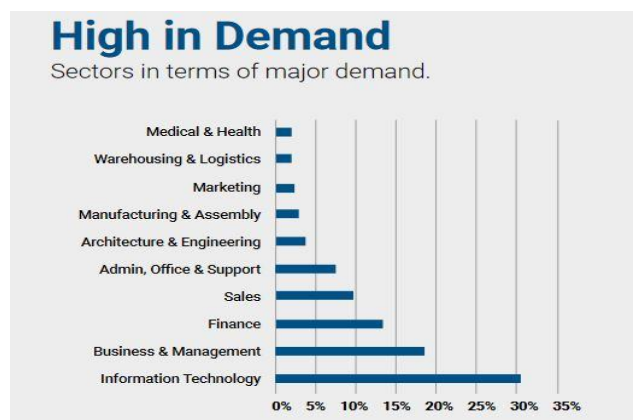


Figure 1.1 High Demand Job Sectors in South Africa (BusinessTech, 2019b)



Figure 1.2 High Supply Job Sectors in South Africa (BusinessTech, 2019b)

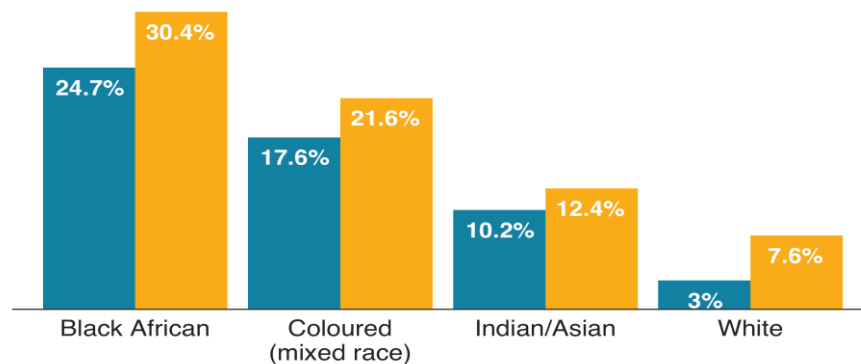
A study conducted by Calitz *et al.* (2014) on the South African ICT graduate skills requirements, found that, even with the Black Economic Empowerment requirements introduced by the South African government, it is difficult to find black candidates who are adequately qualified ICT

employees during the recruitment processes performed by IT companies in South Africa. These findings indicate that there is a black minority in the IT and ICT industry. It is surprising that in a country where the majority of the population is made up of Black South Africans, they are the minority in the IT industry. In the skills jobs market, Black South Africans are still underrepresented. In contrast the White South Africans are wealthier and more skilled than Black South Africans and they make up 10% of the work population due to them being able to attain higher levels of education (Fihlani & Dominic, 2019). (Figure 1.3 below reflects the unemployment rate in South Africa).

Unemployment rates have risen, but still highest among black Africans

How unemployment rates have changed

■ 1994 ■ 2018



Source: Stats SA, 2018 figures are Q4, 2018

BBC

Figure 1.3 Unemployment Rate in South Africa

The shortage of tech talent in Africa is closely linked to the continent's lack of infrastructure. South Africa is Africa's most advanced economy and already has the foundation in industries, such as financial technology and it is seen as the best place to train the next generation of programmers for the 4IR (Chutel, 2016). However, South Africa has not yet escaped the grinding poverty suffered by the country's majority black population. More than half of the country lives in poverty (Chutel, 2017; Fihlani & Dominic, 2019; Statistics South Africa, 2019). (Figure 1.4 below reflects the poverty rate in South Africa).

Poverty is still highest among black Africans

% living in poverty, by population group

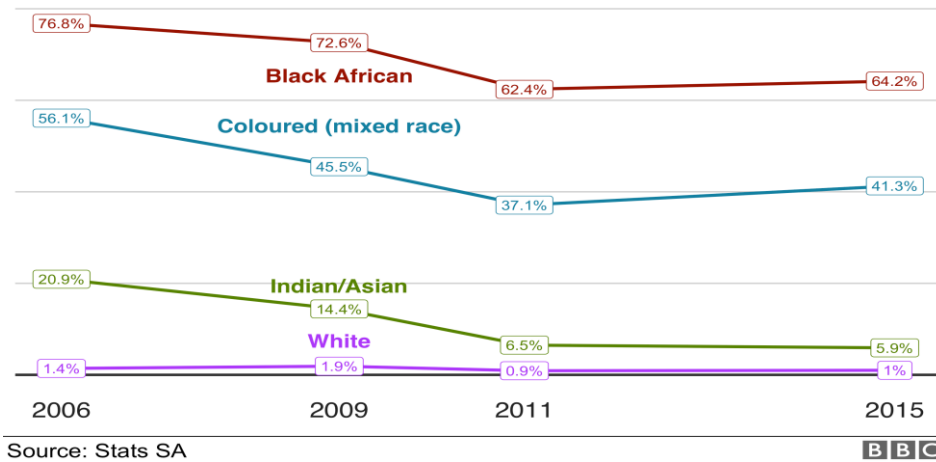


Figure 1.4 Poverty Rate in South Africa

Apart from the less privileged youths that attend schools in townships there are many other schools in South Africa that lack the proper infrastructure for learning programming and it makes it difficult for them to aspire to become IT specialists (Gool, 2018b). The gap in South Africa's public education system means that many potential IT geniuses may never get the opportunity to program (Chutel, 2016). However, a limited number of young South Africans are getting exposure to computers by means of programming initiatives. Project CodeX, CodeSpace, WeThinkCode, Code for South Africa, ByteAcademy, and GirlHYPE are programming schools in South Africa that provide intense programming lessons with classes that have nearly as many female students as there are male (Chutel, 2016).

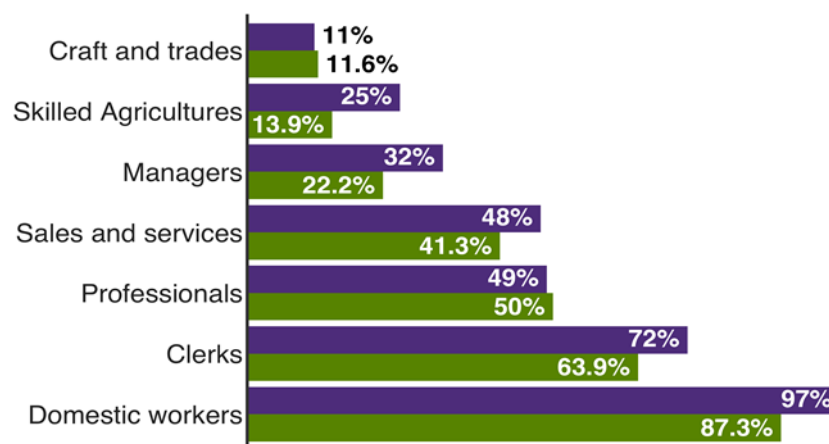
The Department of Basic Education is developing a new curriculum to be introduced in 2024 with new subjects, including entrepreneurship, coding and robotics and the draft curriculum is expected to be ready by the end of 2021 (BusinessTech, 2021). According to the Minister of Basic Education, Angie Motshekga, South Africa has developed a coding and robotics curriculum from Grade R-9, with 54 schools already implementing this new curriculum to develop children's problem-solving, critical thinking and collaborative skills in order to function in a digital and information-driven future in preparation for the Fourth Industrial Revolution (4IR) (BusinessTech, 2019a; BusinessTech, 2021).

South Africa has a population of more than 58 million people, of which women constitute 50.5% (Worldometers, 2019). The number of women who enrol in IT courses in higher education institutions is almost half that of their male counterparts (Pretorius *et al.*, 2015). According to Pretorius *et al.* (2015), the lack of women in IT courses will be reflected in the number of women in the IT and ICT industry. Software developers in the IT industry of South Africa are 83% male and 17% female (PayScale.com, 2018). The South African government does enforce the empowerment of women and gender equality through drafted dedicated policies, such as the Broad-Based Black Economic Empowerment (BBBEE) Act. The South African labour market favours men over women, regardless of race. In South Africa only 32% of managers are women which has risen from 10% in 1995 (Fihlani & Dominic, 2019). (Figure 1.5 below reflects the underrepresentation of women in high-level roles in South Africa).

Women under-represented in high-level roles

But proportion of women managers has risen since 1995

■ 1995 ■ 2018



Source: Stats SA, 2018 figures are Q2, 2018

BBC

Figure 1.5 Underrepresented Women in High-Level Managerial Roles

The proportion of women in employment is lower than that of men, and the labour force participation rate of men is higher than that of women. Remarkably, the unemployment rate of women amongst women is higher than that of men amongst men (Statistics South Africa, 2021). It is evident that female IT specialists are a minority within a minority, with an even greater scarcity

of black females in IT. (Figure 1.6 below reflects the unemployment rate by population group and gender).

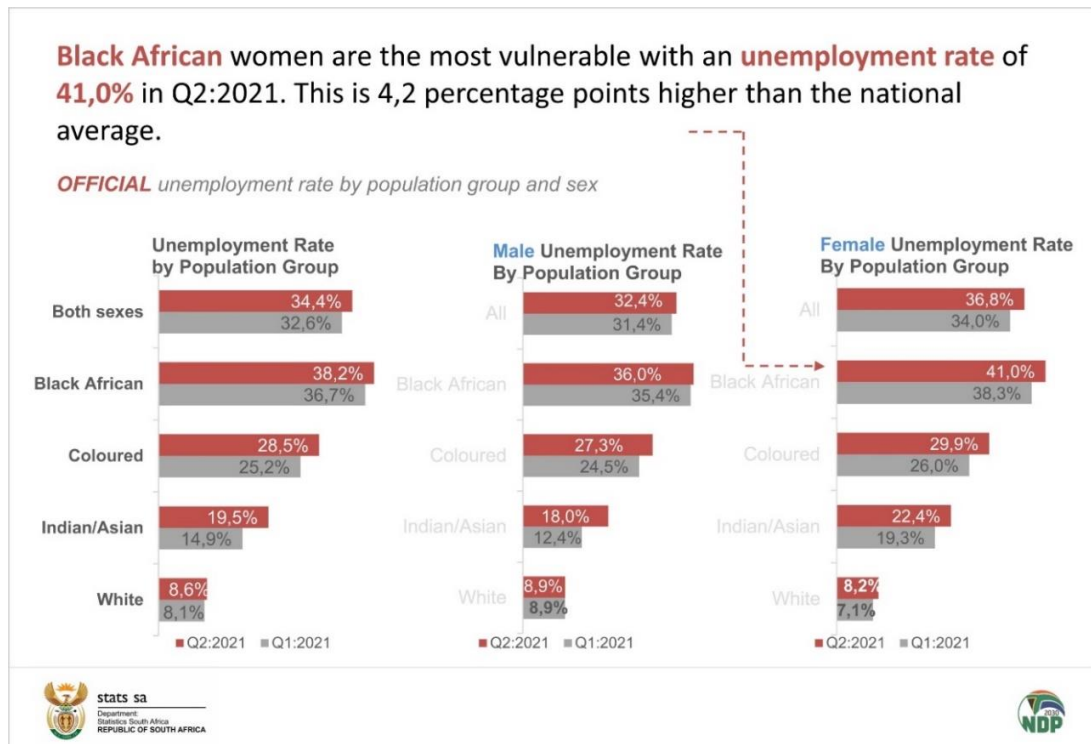


Figure 1.6 Unemployment Rate of South Africa by Population Group and Gender

A gap seems to exist in the literature regarding underrepresented groups in software development in South Africa. An in-depth study has not yet been conducted on the influence of pair programming on the experiences of underrepresented groups in IT in South Africa. Pair programming might address the factors that potentially limit their participation in IT-related fields. Studies conducted by pair programming pioneers, Laurie Williams and Robert Kessler, (Williams & Kessler, 2000a) found pair programming to be beneficial in learning how to program. Williams and Kessler (2001) define pair programming as a technique where two students work together simultaneously at one computer to complete one program. Students can benefit from pair programming because students agreed that they learned programming languages faster when working with a partner (Williams & Kessler, 2001). Research conducted on minorities' experiences with pair programming could provide valuable information that might attract and increase the retention rate of underrepresented groups in IT and encourage them to pursue ICT careers in South Africa.

1.2 TERMINOLOGY CLARIFICATION

For the purpose of this study, it is necessary to explain and clarify the following key terms:

- According to Merriam-Webster (2021), minorities are defined as a part of a population differing from others in some characteristics and often subjected to differential treatment. In the context of this study, members of a minority group are all ethnicities excluding whites and it includes all females in the field of Computer Science and IT.
- Underrepresented means inadequately represented which in the context of this study refers to a group of people that is not adequately represented in a field or course (Merriam-Webster, 2021).
- Pair programming refers to a technique where two programmers work side-by-side on one computer to complete a programming task (This definition will be expanded on in detail in the literature review in Chapter 3).
- Information Technology is the technology that involves the development, maintenance, and use of computer systems, software and networks for the processing and distribution of data (Merriam-Webster, 2021).
- According to the online dictionary by Merriam-Webster (2021), Computer Science is a branch of science that deals with the theory of computation or design of computers.
- Programming is the process of preparing an instructional program for a device (such as a computer) (Merriam-Webster, 2021).

1.3 LITERATURE REVIEW

The information gathered for the literature review of this study was from primary research sources, such as academic journals, books and research publications to address the objectives of this study. The researcher used ScienceDirect, Google Scholar in addition to IEEEExplore, and ACM DL research databases for the literature study. The keywords used were as follows: “pair programming”, “minorities”, “underrepresentation”, “UTAUT”, and “minorities in IT”.

Prior research in the literature has shown that students who enrol into a programming course are often first-year university students. It is vital that those students become accustomed to programming so that they can acquire the necessary skills required throughout the duration of their degree so that they can integrate into the industry (Teague, 2011).

Programming has always been regarded as an individual activity. In the literature, studies conducted on pair programming often focused on a few differences for example, skill levels and personality types. Salleh *et al.* (2010a) state that comprehensive studies that involved social individual difference variables which are deemed pivotal in education terms for pair determination, are required. No obvious conclusions have been reached in the literature on whether homogenous or heterogeneous pair programming pairs are more effective, according to variables, such as personality type. Investigating compatibility aspects is crucial in the sense that pair programming innately involves a social interaction between two people (Salleh *et al.*, 2010a). Thus, it is important to make it easier for all programming groups to work together at a higher level (Salleh *et al.*, 2014).

The limited access to a computer at school, and at home and the differential use thereof is crucial since it is the main contributor to the unequal participation in IT (Varma, 2002). The responses to the questionnaire by minority IT students indicated that many of them do not have a computer at home due to the fact that these students come from an economically disadvantaged background (Varma, 2002; Varma, 2020). It is likely that these students attended schools that do not have computers at their school. According to Varma (2002), white male students, in addition to white females to a certain extent, were exposed to a computer at earlier ages in comparison to non-White students.

1.3.1 MINORITY GROUPS IN IT

Minority graduates in STEM fields are far less in number than traditional graduates in those fields. Ethnic and gender patterns have been established in prior research in the literature as a way to address research questions with regard to the underrepresentation of minorities in STEM. It has been found to be a good way to explore the experiences of undergraduate minority and female students. It allowed the researchers to compare the experiences of the ethnic and gender minorities against the experiences of the ethnic and gender majority while focusing on their representation in their own respective ethnic groups (Kuleshov *et al.*, 2021). Groups and communities that have largely been considered as minorities in Information Technology are from non-English speaking backgrounds, have a low socio-economic status and also live in rural and remote areas (Leung, 2014). Little has been published about who these minority graduates are and how long it took them to pursue their respective degrees in STEM. There are some opinions

that exist that can explain who these individuals are, where they come from and what their interests are (Kuleshov *et al.*, 2021).

In South Africa, these groups of people from a demographic standpoint, make up the majority of the country's population, but in the context of IT in South Africa they are part of the minority group which remains underrepresented. Sections 2.4 and 2.5 of the literature study focus on the various aspects of underrepresentation of black people and women in IT courses, such as the decline in the number of undergraduate students in IT, understanding the background, the psychosocial factors that either attract or repel students, and the need to increase the number of students in Computer Science courses.

1.3.2 PAIR PROGRAMMING

Pair programming is a collaborative learning technique where two programmers work together using one computer by assuming different roles in order to develop an algorithm or code (Williams & Kessler, 2003). The “driver” is tasked with typing and developing the code, while the “navigator” monitors the driver's code to check for errors and suggest solutions for problems (Williams & Kessler, 2000a). Both of the programmers brainstorm throughout the development process of their project by exchanging roles frequently (Williams & Kessler, 2003; Williams & Kessler, 2000a).

In recent years, there has been an increased need for individuals with high-level skills such as critical thinking, problem solving and teamwork. The main focus of modern day education is to equip students with these vital skills. With the rise of the 4IR, coding education has been introduced in the curricula of many countries across the globe. These coding subjects aim to provide young individuals with basic coding skills. In South Africa, IT as a subject can only be chosen at the end of Grade 9 which has a significant impact on students' choice of IT as a subject (Liebenberg *et al.*, 2012). The South African Department of Education has designed a coding and robotics curriculum and will be implemented at different levels from Grade R to Grade 9 (BusinessTech, 2021).

Pair programming is a popular strategy in Computer Science education to teach programming to novices (Vandenberg *et al.*, 2021). According to Demir and Seferoglu (2021), pair programming draws attention to how to effectively and efficiently conduct coding education since it is based on

the collaborative learning method which forms part of the constructivist learning approaches. Prior studies in recent decades have shown that students who learn how to program by means of a collaborative approach perform better in completing a programming task and improve their programming skills than those who program on their own (Zhong *et al.*, 2016). Pair programming has shown to be an effective strategy to teach programming compared to other approaches. Section 3.3 of the literature study focuses on the various aspects of pair programming, such as its principles, implementation and the difference between solo programming and pair programming, and the impact that pair programming has on students.

1.3.3 UNIFIED THEORY OF ACCEPTANCE AND USE OF TECHNOLOGY

The Unified Theory Of Acceptance And Use Of Technology (UTAUT) is a technology acceptance model formulated by Venkatesh *et al.* (2003) which aims to explain the user intentions to use an information system, in addition to subsequent usage behaviour. This model has been used to study the entire spectrum of technology from initial adoption to post-adoptive use (Venkatesh *et al.*, 2012). The UTAUT model has four predictors of intention and technology use, namely Performance Expectancy (PE), Effort Expectancy (EE), Social Influence (SI), And Facilitating Conditions (FC) (Venkatesh *et al.*, 2003; Venkatesh *et al.*, 2012; Venkatesh *et al.*, 2016) (Figure 1.7 below displays the UTAUT model).

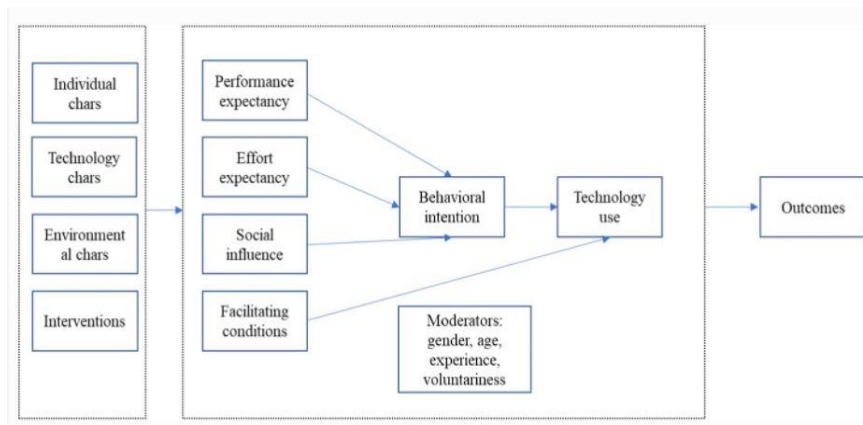


Figure 1.7 UTAUT and Suggested Future Research Directions

In this research study, the researcher used the UTAUT2 model formulated by Venkatesh *et al.* (2016) as a basis for the development of the questionnaire which includes four more constructs, namely Hedonic Motivation (HM), Habit (H), Behavioural Intention (BI) And Use Behaviour (UB).

These constructs serve as a backdrop for conducting the research and to better understand student adoption and use of pair programming as a learning tool (Venkatesh, 2021).

The questionnaire developed to gain data for this research study had the aforementioned eight constructs and its questions were based on a five-point Likert scale. Sub-section 4.3.5.2 of the research design discusses how the questionnaire was developed using the UTAUT2 model in order to gain a better understanding of the impact of pair programming on the perceptions and experiences of underrepresented groups in IT.

1.4 AIM OF THE RESEARCH

The primary aim of this research study is to investigate the impact of pair programming on the experiences and perceptions of underrepresented groups in Information Technology in South Africa.

1.4.1 OBJECTIVES OF THE RESEARCH STUDY

In order to accomplish the primary aim of the study, the following secondary objectives are set:

- 1.4.1.1 To examine the literature on the underrepresented groups in Information Technology;
- 1.4.1.2 To examine the literature on pair programming; and
- 1.4.1.3 To investigate the impact of pair programming on the students' experiences and perceptions.

1.5 RESEARCH DESIGN AND METHODOLOGY

1.5.1 RESEARCH PARADIGM

The positivist research paradigm does not always rely on quantitative methods. Positivism is aligned with the hypothetic-deductive model of science. This method is a circular process in the sense that it starts from the literature in order to form a hypothesis that can be tested, designs an experiment via the identification of variables that can be manipulated and measured by means of group assignments (operationalising variables), and conducts an empirical study, based on experimentation (Park *et al.*, 2020). Therefore, the research findings from such a study are used

to aid in forming of theory and also contribute to the literature. In doing so, the circular process of the hypothetic-deductive scientific model is complete.

In the context of this research study, empirical evidence findings help to refine and strengthen theory. A hypothesis that confirms the effectiveness of an instructional approach (pair programming) to a group of first-year students can aid in refining theory. The study was conducted, using the positivistic paradigm and acted as a basis that supported quantitative and qualitative research methods (mixed-method approach) (Leech *et al.*, 2010).

1.5.2 RESEARCH DESIGN

According to Parfrey and Ravani (2021) the aim of research design is to minimise error, to ensure adequate samples, to appropriately measure input and output variables, to consider the internal, in addition to external validities, to limit biases, and also to address statistical relevance. The research approach that was used for this study was the mixed method research approach. Both quantitative and qualitative data was collected and provided distinctive types of information. This study, similar to others, used the quantitative research design to make use of inferential statistics which included statistical significance tests. However, to counteract the adequacy of the significance tests and the limitations of p values, this study also collected data, using a qualitative research design (Nassaji, 2021). This mixed-method approach was applied because significance tests only revealed to the researcher whether an effect exists, but did not tell the researcher about the importance of the effect. It was key for the researcher to supplement this quantitative data with qualitative data which allowed the researcher to better understand and interpret the effect sizes qualitatively (Nassaji, 2021). This was done by collecting open-ended data from the survey questionnaire developed for this study. A comparison was made from the different perspectives of individuals drawn from both types of data, which created a greater understanding of the necessary changes that were required for a marginalised group by integrating qualitative and quantitative data (Creswell & Creswell, 2017).

1.5.3 PARTICIPANTS

The target population for this study was first-year students from the Potchefstroom Campus of the North-West University (NWU) in South Africa. The researcher used convenience sampling and the selected study population is a group of students that reflect the demographics of the

target population. The sample size was 284 first-year BSc IT students in the User Interface Programming class from the NWU. This programming module comprised students from different ethnic groups, cultures, genders and socio-economic backgrounds. The participants who were used for this research study included students who had experience with IT subjects in high school, and students who were completely new to programming. The students were taught the principles of pair programming and were partnered with a friend and used one computer to complete practical programming assignments on a weekly basis for the duration of the semester.

1.5.4 DATA COLLECTION METHOD

A questionnaire based on the Unified Theory of Acceptance and Use of Technology model was used to gather information from the students. The questionnaire consisted of a set of five-point Likert scale response questions (quantitative) based on the determinants of the UTAUT2 model and a set of open-ended questions (qualitative). The determinants used were Performance Expectancy, Effort Expectancy, Social Influence, Facilitating Conditions, Hedonic Motivation, Habit and Behavioural Intention. The results from both quantitative and qualitative data should provide a better understanding of the impact of pair programming on the students' experiences.

1.5.5 DATA ANALYSIS

Data analysis enabled the researcher to induce the underlying inference from the collected data by means of removing duplicate observations, missing codes, empty fields and data that was formatted in an incorrect manner. This was done to prevent the researcher from making false conclusions (Bernadita, 2021).

This study used the mixed methods approach. Firstly, the researcher used of factor analysis to aid in determining if any relationships existed between a set of variables which describes patterns in a relationship (Arora, 2021). Secondly, quantitative analysis was used to measure the differences between groups. Quantitative analysis was deemed adequate because the descriptive statistics described the sample for the study and inferential statistics drew conclusions about the entire target population. With regards to inferential statistics, the researcher used of T-tests in order to compare the means of two groups of data which is important in understanding how different or similar these two groups are (Jansen & Warren, 2020). The researcher also used analysis of variance (ANOVA) which provided a statistical test for multiple groups involved in the

study to assess whether these groups are equal; thus it generalises the T-test beyond two means (Arora, 2021; Jansen & Warren, 2020). The researcher reported the quantitative statistical results and identified and discussed the qualitative findings (themes) that either confirmed or disproved the quantitative results. The application of statistical analysis is a technique that allows the researcher to perform statistical manipulation in order to quantify the data. The tool that was used to perform the statistical analysis in this study was Statistical Package for the Social Sciences (SPSS) (Arora, 2021). The information obtained from the quantitative data was analysed by the North-West University's Statistical Consultation Services.

Qualitative data is usually presented as unstructured text-based data which is rich and subjective. In the context of this study the textual data was collected from a structured questionnaire. As stated by Wong (2008), qualitative research explores the perspective of meaning of experiences and can identify social structures that explain people's behavioural meaning. The information acquired from the qualitative data was analysed, using ATLAS.ti 9, a data analysis software package that aided in content analysis by coding the data and categorizing the codes into themes as they emerged. Content analysis was performed in this study with the intention of identifying patterns of deeper underlying interpretations (Jansen & Warren, 2020). The results were used to come to a conclusion on the impact of pair programming on the experiences of underrepresented groups in IT courses.

1.5.6 ETHICAL CLEARANCE

The study adhered to the ethical requirements of the NWU. Permission to conduct this study was obtained from the Faculty of Natural and Agricultural Sciences Research Ethics Committee (FNASREC) under the ethics number: NWU-01471-20-A9 (see Appendix – A). The students who participated in this study signed and completed the informed consent form at the start of the questionnaire that was distributed to them electronically and they were also informed that by providing consent, the confidentiality of their responses in addition to their anonymity would be ensured.

1.6 DISSERTATION STRUCTURE

1.6.1 CHAPTER 1: ORIENTATION

In Chapter 1, an overview of the study is provided, with the focus on the background and the research problem statement which investigates the factors that impact the experiences of underrepresented minorities in IT in South Africa. Furthermore, in this chapter the research questions and objectives that the study aimed to address are also stated. A brief description of the research methodology and data analysis is included.

1.6.2 CHAPTER 2: MINORITIES IN IT

In Chapter 2, the cause of historical underrepresentation of ethnic minorities in STEM in South Africa is outlined, in addition to their participation in those fields at a tertiary level and in the industry. Furthermore, in this chapter, the presence of women in STEM, the digital divide and the closing thereof and also a solution for the lack of minorities in STEM fields are addressed.

1.6.3 CHAPTER 3: PAIR PROGRAMMING

In Chapter 3, the relevant literature regarding pair programming is explored in-depth. The development of software in the industry implementing pair programming, pair programming as a software development methodology, pair programming in the classroom setting as an educational tool, based on the prior research done on this topic in past decades are discussed. The necessary guidelines required for implementing pair programming, the challenges, in addition to the advantages and disadvantages of this collaborative learning approach are also provided.

1.6.4 CHAPTER 4: RESEARCH METHOD

In Chapter 4, the focus is on the development of the data collection instruments that were used in this study. The research paradigm and its application in the study are discussed and an extensive discussion about the mixed-method research design approach is included. Secondly, the students who were involved in the study are described briefly. In conclusion, the development of the survey questionnaire which focused on the qualitative and quantitative data and collection methods is discussed.

1.6.5 CHAPTER 5: RESULTS

In Chapter 5, the results of the study are presented. The quantitative data focused on the descriptive statistics such as the demographical information of the students, followed by the presentation of the results from the T-tests, and ANOVAs. The qualitative data presents the thematic analysis that emerged from the data of the open-ended questions.

1.6.6 CHAPTER 6: CONCLUSION AND RECOMMENDATIONS

Based on the statistical results presented and analysed in Chapter 5, the evaluation of the study findings concludes this final chapter by outlining the recommendations to the relevant issue of underrepresentation in STEM and the possible future research that can be undertaken on this topic to better understand the individuals that belong to these minority groups is also discussed.

1.6.6.1 STUDY CONTRIBUTIONS

Little research has been conducted on pair programming and the manner in which it impacts minorities' experiences in IT. In addition, there is also little research found on this topic in the South African context. This in-depth mixed method study can contribute to a better and deeper understanding of minorities' perception and experiences with programming and the field of IT and Computer Science, which might lead to increased enrolment and retention of minorities at a tertiary level, in addition to their pursuit of careers in IT and Computer Science fields.

1.6.6.2 STUDY LIMITATIONS

This study used the mixed-methods approach and a set of open-ended questions in the questionnaire. Some of the students of the study did not provide sufficient information with regards to the open-ended set of questions. Perhaps in the future in addition to the use of the mixed-methods approach the researcher could gather more detailed information by means of interviews. This would greatly benefit future research in order to gain a better understanding of the perceptions of underrepresented groups in IT in South Africa. The generalisation of the findings from this study is limited in the sense that it focused and gathered information only from one campus of a university in South Africa.

CHAPTER 2 MINORITIES IN INFORMATION TECHNOLOGY

2.1 INTRODUCTION

In as much as there is a global demand for graduates with computing skills, and a global shortage of female computing professionals, in Africa these aforementioned shortages are even more critical (Nicolau & Seymour, 2017). Regardless of the various efforts made in the industry to increase the participation of both racial, as well as all ethnic minorities. In STEM fields, these groups of people remain underrepresented (Aish *et al.*, 2018). In recent years, increased focus has only been put on support structures to aid this group of underrepresented people to get through the industrial pipeline, yet little attention has been paid to increasing minorities' intrinsic motivation to follow careers in STEM fields (Aish *et al.*, 2018; Uzzaman, 2019). Artificial intelligence, the Internet of Things, edge computing, mass scale commercialisation of quantum computing use, aerospace, Starlink broadband/5G internet, etc. are exciting technological trends that promise to revolutionise our lives and the society around us (Whitney & Taylor, 2018). Despite the excitement that technology engenders, there is still a notable lack of participation from women, people of colour, and people with disabilities (Whitney & Taylor, 2018). Minorities and women are essential in filling this perceived gap in STEM fields (Varma, 2018). In this chapter, the researcher discusses the history of STEM in South Africa, the current landscape of STEM in academia, and in the industry. Furthermore, the importance of a diverse pool of role models in STEM fields, and possible solutions to increase the participation and persistence of minorities in IT to bring forth change and to increase diversity, and inclusion are discussed.

2.2 HISTORICAL VIEW OF STEM IN SOUTH AFRICA

The historical, cultural, in addition to political policies of South Africa are responsible for the production of scientific knowledge and the education of the future scientists of this country. Inequalities in class are firmly established in scientific practices, which makes it difficult to attain the promise of transformative or developmental effects for society through Science, Engineering and Technology (SET) fields (Sooryamoorthy, 2015). With regard to the structures in class, historically there has been gendered and racialised discourse entangled within it by means of racist and sexist formulations of science in South Africa (Liccardo & Bradbury, 2017).

“The Bantu Education Act of 1953 provided the establishment of an education system that was racially differentiated and purposely designed to lodge the privileged position of white South Africans by intentionally giving black South African learners education that was inferior” (Kallaway, 1984). Therefore, in most black school’s subjects such as mathematics, in addition to science were not available. In severely underfunded institutions whereby the aforementioned subjects were available, the educators were in most cases underqualified and supplied with a neglected curriculum to teach science and mathematics to black learners (Dalvit & De Klerk, 2005).

In similar fashion, institutions of higher education in South Africa developed along the same lines. In 1829, the University of the Cape of Good Hope which is the oldest English university in the country and today known as the University of Cape Town was established. Then, in 1874, the oldest Afrikaans university namely the University of Stellenbosch was built and received full university status in 1918. The University of Witwatersrand was established in 1896, Rhodes University in 1904, the University of Pretoria was constructed in 1908 and lastly, the University of Natal was founded in 1909. These universities were historically white and were established with the primary objective of serving the educational needs of White South African males. Provisions for the establishment of the first university for black South Africans were made after the passage of the University Act of 1916 (Mabokela, 2000). This aforementioned legislation established the South African Native College, which today is known as the University of Fort Hare, founded in 1916, the oldest black university in South Africa, which was designated for the Xhosa-speaking Africans (Makura *et al.*, 2011).

When the National Party came to power in 1948, four Afrikaans universities were instituted: in 1950, the University of the Orange Free State, the University of Potchefstroom for Christian Higher Education in 1951, now known as the North-West University, the University of Port Elizabeth in 1965 and lastly, the Rand Afrikaans University in 1968. These new universities, alongside the University of Stellenbosch, adhered strongly to the principle of separate development and also implemented strict policies against the admission of black South Africans. African students were viewed as a noticeable group from the generic ‘black’ nomenclature, which comprised Native African, Indian and Coloured people and they belonged to the racial and ethnic groups that were regarded as disadvantaged during this era (Ramrathan, 2016). The Extension of the University Act of 1959 brought forth racially, and ethnically-based universities for Black, Indian and Coloured (mixed race) South Africans.

The first group of ethnically-based universities comprised of seven rural African universities. The universities included in this category were the Universities of the North and Zululand, both established in 1960 (Mabokela, 2000). The five remaining rural universities were created in the former independent homelands of Bophuthatswana, Ciskei, Transkei, and Venda. In the self-governing homeland of Qwa-Qwa, a branch campus of the University of the North was established (Muller, 1991). The University of the Western Cape and the University of Durban-Westville were established for the Coloureds and Indians of South Africa, respectively (Mabokela, 2000). Despite the English universities (Cape Town, Witwatersrand, Natal and Rhodes) admitting a few black students, they enforced policies that professed academic freedom, but maintained social segregation during the Apartheid era which left Black South African students without any form of equality (Murray, 1990).

This study was conducted at the North-West University (Potchefstroom Campus), here follows a quick look at the historical background of this institution of higher education. The NWU is multi-campus university located in South Africa on three campuses in Potchefstroom, Mahikeng and Vanderbijlpark (NWU, 2022b). In 1869, the Potchefstroom University College for Christian Higher Education was developed out of the Theological School of the Reformed Churches in South Africa and was located in Burgersdorp in the Cape Province (Jooste, 1957). In 1905, it was transferred from Burgersdorp to Potchefstroom in the Transvaal. In 1951, The Potchefstroom University College for Christian Higher Education was recognised as an independent and was renamed to the Potchefstroom University for Christian Higher Education (Ferreira, 2006). In order to provide tertiary education in the Vanderbijlpark area, in 1966 the Vaal Triangle Campus was established. In 1979, Lucas Mangope (the then president of the Bophuthatswana homeland), founded the University of Bophuthatswana with many residents of Bophuthatswana contributing to establish the university (Bantham, 2017). The university was under-resourced at first, but the South African Government invested heavily in the university. The open and progressive agenda behind the university attracted black South African academics and British post-colonial academics (Bantham, 2017). In 2004, the University of the North-West (formerly the University of Bophutatswana) merged with the Potchefstroom University for Christian Higher Education as well as its branch in Vanderbijlpark (NWU, 2022b). The merger into the unified NWU is hailed as one of the most successful mergers in the country and was seen as a strong symbolic act of reconciliation and nation building (NWU, 2022a).

The creation of segregated black universities in South Africa succeeded in legitimising and solidifying the intention of segregating the racial, in addition to the ethnic groups of the country. This was done by the governing party to provide personnel to control and support structures in newly-created homelands, in order to preserve and propagate the inferior social and economic position of black South Africans (Mabokela, 2000). During this era, the funding for black higher education was severely low, and as a result, it led to reduced participation by the majority of the black population of South Africa that was only being schooled for jobs that were low-skilled (Brown, 2006). The 1980s in South Africa were plagued with massive revolts against the National Party's government which included mass meetings, strikes, boycotts, economic sanctions, and rebellious attacks by political movements that faced exile. In response, the University Amendment Act was passed in the year 1983. The passage of this legislation legalised the admission of black South African students into historically white universities (Mabokela, 2000). The apartheid regime racialised and gendered the landscape of higher education in South Africa, and urgent reform in the higher education landscape was imperative and was one of the areas of concern that the African National Congress government undertook when it gained power in 1994 (Akala & Divala, 2016). The 1997 Higher Education Act stated that all of the universities of South Africa were required to comply with the proper measures for the redress of past inequalities and were not allowed to unfairly discriminate in any way (Department of Higher Education & Training South Africa, 2014). This legislation aided institutions of higher education in overcoming racism and sexism (Sehoole, 2013).

2.3 THE DIGITAL DIVIDE

The digital divide can be defined as consumer disparities in access to information, in addition to communication technologies, based on factors, such as age, gender, race, socio-economic status and geographic location (Boone *et al.*, 2014; Mossberger *et al.*, 2003). This divide mainly focuses on the differences of those people who are able, and are not able to use computers and the Internet for communication and obtaining information effectively. In the Social Sciences it has been proved that the environment does have a significant impact on shaping an individual's opportunities, particularly when poverty and segregation are involved (Mossberger *et al.*, 2006). Serious inequities persist in poor urban communities in the United States of America and in South Africa despite decades of civil rights and legislation in respect of fair housing in both countries (Harloe, 2008; Huchzermeyer, 2003). With the dawn of the Information Age, these disparities

have transformed over the decades and new barriers to access to technology and skill in poor communities have developed.

With the increase in proportion of American households with Internet access during the 1990s, it spurred many studies that documented the Internet usage trends across population subgroups and these studies identified a digital divide with regard to computer and Internet access with respect to race and gender in the United States of America. These studies documented that Caucasians in America, compared to other races and ethnicities, are most likely to have access to the Internet. Consequently, this divide occurred due to the difference in socio-economic positions related to income and education attainment (Campos-Castillo, 2015). Research suggests that the gender gap has been closed, and that now, women have more access to the Internet and use technology more than men (Talukdar & Gauri, 2011). More recent data from studies conducted by the National Telecommunications and Information Administration still suggests that the digital divide exists and has also shown that men are more likely to use the Internet than women, regardless of racial and ethnic groups, because of gender stereotypes that exist regarding technology usage (National Telecommunications & Administration, 2013).

The digital divide is an area of concern that has a grave impact on thousands of minority households. These countless households worldwide have very limited or no access to educational programs that involve computer technology (Boone *et al.*, 2014). Poor communities in America are African American and Hispanic. Due to their low income, these communities have the least home access to the Internet and computers and information literacy.

A survey by Mossberger *et al.* (2003) on virtual inequality, showed that race and ethnicity do matter for Internet and computer access and found that African Americans and Hispanics are statistically less likely to have access to computers and the Internet because the use of technology requires reading skills, and the ability to search, use and evaluate information which makes them information illiterate. Approximately 20% of the respondents in this study reported that they required assistance using a keyboard and mouse in order to find information on the Internet and to make use of the programs on a computer.

According to Mossberger *et al.* (2003), race and ethnicity only have an influence on an individual's attitude towards technology. Surprisingly, compared to white respondents, African Americans have a more positive attitude towards IT, and view it as important for economic opportunities and

being up to date with current affairs and world news. Furthermore, African Americans are more willing to learn new computer skills online than their Caucasian counterparts. Hispanics' attitude towards the Internet and technology does not differ from that of Caucasian respondents, but they are also more likely to learn a variety of skills from online courses.

Over the last 25 years, the use of computers and the Internet in South Africa has grown at an accelerated rate according to census data, but there is a systemic difference between the citizens who are connected to the Internet and those who are not connected (Salahuddin & Gow, 2016). The acquisition of Internet access and other related computer technologies by black South Africans, minorities, including the less fortunate, is closing the digital divide at a very slow rate. A study about the use of mobile phones by South African university students conducted by North *et al.* (2014) found that these students use mobile phones for socialising, safety and privacy. Furthermore, in their survey, they found that in their sample of 362 students that only 1% of them did not own a mobile phone (North *et al.*, 2014; Parry & le Roux, 2018). Students who attend low-income schools use technology less, due to their educators having inadequate training and being unable to teach students the basic technical skills that are required to operate a computer, whereas their affluent counterparts learn these skills in the comfort of their homes. The complexity of this problem is very difficult to solve due to issues in these communities, such as the lack of computers, the lack of experience with computer technology, in addition to the lack of access to the Internet (Boone *et al.*, 2014). Racial segregation (Apartheid) and poverty are environmental factors in poor communities that account for the differences in access to technology, and skills with respect to race and ethnicity dissimilarities (Mossberger *et al.*, 2006).

Thus, living in a region that is less fortunate, might have an effect on the technology access and skill of individuals due to poor services and regional employment. Environments, such as schools, libraries, and community centres have been identified as geographical locations that have an impact on technological skill and access. In recent years, the use of computing has increased substantially in schools in the form of mobile devices and mobile gaming. However, in South Africa, interest in studying computing at a tertiary level has not increased substantially (Nicolau & Seymour, 2017).

Employment is another factor that influences individuals use of computers and the Internet. In almost all workplaces in recent years, employees are provided with formal, in addition to informal computer training. This means that individuals from impoverished communities are introduced to

computers and the Internet at work and thereafter they eventually acquire computers for themselves to use at home, clearly indicating that jobs present a crucial initial step in adopting technology (Telecommunications & Administration, 2013). The problem is that in South Africa the unemployment rate was 27,5% in the first quarter of 2019 (Statistics South Africa, 2019). The majority of these individuals who are unemployed only attained high school education, meaning that their level of education limits them to low-skill jobs that are less likely to demand the use of computers and the Internet. Informal settlements in South Africa are areas of concentrated poverty and high levels of unemployment, thus lacking information and contacts that could lead to job opportunities (Hurlbut, 2018).

2.4 UNDERREPRESENTATION OF MINORITIES IN CS AND IT

The severe underrepresentation of minorities and women in undergraduate Computer Science and Information Technology and Information Systems courses is a prevalent problem worldwide. It is critical that there is a better understanding of the background, as well as the psychosocial factors that either attract or repel students who are in the minority in computing disciplines (Buzzetto-More *et al.*, 2010). There is a shortage of high-quality professionals in CS and IT. With the rise in demand for programmers and computer scientists who are qualified, there is a great need to increase the number of students in Computer Science courses (Baser, 2013).

Primarily, Computer Science (CS) is a degree that is obtained by white men due to the fact that it is a degree that typecasts candidates and it has yet to put much effort into diversifying with respect to minorities and women (Bock *et al.*, 2013; Stross, 2008). The IT industry has had a questionable record of hiring workers of colour. In 2011, America only had 15% of Hispanic workers and 10.8% of African American workers in their labour force. The computer and mathematics occupations only comprised 5,7% Hispanic and 6.9% African American professionals (Bock *et al.*, 2013).

A study conducted in 2010 in the United States of America, established that the below par advertisement of the CS and IT majors, the lack of experience studying computing prior to college and the lack of exposure to coding cause minority enrolment, success and retention rate in CS and IT (Bock *et al.*, 2013). A similar study by Buzzetto-More *et al.* (2010) found that African Americans with a higher socio-economic status had higher ownership of electronic devices, including computers than those with a low socio-economic status, but they are still inadequately

informed when selecting a Computer Science or IT degree. Socially constructed factors, such as early advanced engagement with computers and programming, the knowledge of the nature of computers, foundation in mathematics and science and technological incubation have a great impact on their pursuit of CS and IT degrees. Regardless of the high levels of ambition and positive attitude towards IT that African Americans do display, these aforementioned social factors are the contributing factors to their attainment of CS and IT degrees (Bock *et al.*, 2013).

One of the reasons why students do not choose Computer Science at undergraduate level is that students have a false perception that CS courses that involve programming are too difficult, boring and anti-social (Baser, 2013). Consequently, students tend to develop a negative attitude towards programming. It is evident that a positive or negative attitude towards something has an influence on how we process behaviour in general and our behaviour in particular. Accordingly, their attitude towards programming affects their performance in programming. The identification of students' attitude towards computer programming is a critical issue in the education of Computer Science, since it is clear that attitude has an immense effect on a student's computer programming performance. Attitude does indeed differ between genders (Baser, 2013). The identification of the factors causing the underrepresentation of minorities in IT fields is also an important issue in the retention gap in the IT industry (Tapia & Kvasny, 2004).

In 2014, technology companies, such as Google, Facebook and Microsoft provided a momentous public disclosure of the demographic make-up of their respective workforces which further perpetuates the misrepresentation of minorities in Information Technology fields (McGee, 2018). The data from the disclosures was predictable for some because it confirmed a phenomenon that is widely-known, namely that the U.S. technology workforce is predominantly male, and white and becoming increasingly Asian (McGee, 2018). This indeed does prove that these public disclosures shed light on the lack of diversity in the technology sector. Even though there is an anticipated increase in STEM occupation in recent times, of which technology is the quickest growing occupation, it implies that there are insufficient IT workers to fill this demand (McGee, 2018).

Research in the IT industry has shown that a workforce that is inclusive, and diverse, correlates with the increase in creativity, innovation, in addition to higher profits which results in greater revenue for tech companies (McGee, 2018). However, these statistics on U.S. tech companies suggest that large portions of the US population, namely women and ethnic minorities remain

underutilised. Public workforce disclosures have led to tech companies investing millions of dollars every year in recruitment initiatives that are meant to increase workforce diversity, but little change has occurred due to the diversity numbers remaining stagnant. As a result, there has been a surge in gender discrimination and sexual harassment allegations while retention of minorities, as well as their advancement in this field receive little or no attention (Dickey, 2017; Fiegerman, 2017; Simon-Lewis, 2017).

2.5 WOMEN IN IT

It is difficult to uncover the nuances that potentially could give a better understanding of the underrepresentation of women in STEM fields, but due to researchers' depiction of women as a homogenous group, ignoring the differences in the technology between women, they portray white women's experiences as a prototype for all women. Industry research tends to put people of colour into one group and identifies them as the minority. The results of these studies universalise the experiences and challenges that people of colour face in tech companies which overlooks the collective, as well as the individual experiences of women of colour (McGee, 2018).

There are many scholarly texts about this category of minorities in IT on a global scale. Therefore, the IT and ICT professions can only be understood by considering their historically race- and gender-segregated workforce. Female IT specialists are a minority within a minority, with an even greater scarcity of black female IT specialists (Whitney & Taylor, 2018). Race or ethnicity, and gender create different perceptions and experiences on a qualitative level which are factors that are commonly believed to hamper the advancement of women into senior roles in IT and ICT (Bosch *et al.*, 2019).

Information Technology is a relatively new profession that stems from the Computer Science field and the development thereof is not well documented, even though women played a significant role (Thompson, 2019). Women aided in the development of the first programming languages, influenced the design, as well as functionality of the first computers, and also came up with new ideas that shaped the field to what it is today (McGee, 2018). It is of the utmost importance that the omission and the minimisation of women's contributions to computing are acknowledged in order to stop the misconceptions of women being uninterested in technology and being incapable of being leaders in computing (Thompson, 2019). In spite of the fact that the history of computing only focuses its attention on the contributions of white women to the field, their experiences shed

light on gender barriers and gendered and race concepts that continue to have an influence on the career advancement of women of colour in IT (McGee, 2018).

It is recognised that women have less prior experience with computers and the Internet than men when starting the Computer Science degree and Computer Science departments at universities tend to favour applicants who have had CS or IT as a subject in high school (Boone *et al.*, 2014). Another hurdle that impedes women from pursuing a CS degree is that academic staff tend to treat female students differently from male students and it is also a male-dominated degree which gives women the impression that CS is a degree that better suits men (Coger *et al.*, 2012). High school teachers, as well as university lecturers should be aware of their attitude towards females in CS and IT because through their actions they might deter females from IT (Liebenberg *et al.*, 2012). Another obstacle that women face is the notion that CS is too difficult because it only involves programming (Papastergiou, 2008). Gender stereotyping exists among learners in high school which has an influence on their peers without them even realising it (Liebenberg *et al.*, 2012). This is also a barrier that keeps women from Computer Science degrees because of the fallacy that it is only a degree for “nerds”, “geeks” and “hackers” which are characteristics usually associated with young, white American males who are intelligent, computer savvy and socially inept (Stoilescu & McDougall, 2011).

Science education research agrees that there is a positive correlation between attitude and achievement; attitudes play significant roles in the perception of information and affect the degree of its retention (Baser, 2013). Research has shown that the reason for underrepresentation of women in CS-, and IT-related disciplines is due to subconscious behaviour that results in the promotion of education, in addition to more careers for males rather than for females (Varma, 2002). It is critical that school systems need the proper infrastructure to provide girls and minorities with easy access to computers. Female educators who are part of the CS department at universities should teach at local high schools in order to help expose girls to IT education to inspire them to pursue careers in IT (Aish *et al.*, 2018). Universities can also do more by making sure that their admission policies clearly specify that prior experience with computing is not required to enrol in a CS degree over time, so that more women can develop an interest in computing (Varma, 2002). To solve the confidence crisis that women face regarding their programming abilities, advisors should convey to them that they should not measure themselves against the typical stereotype associated with Computer Science. Informal events between high

school girls and female graduates would be a starting point to open a channel of communication between them (Burrell, 2019). This would be important for the representation of women in IT because these high school girls could interact with IT professionals at work.

2.6 CLOSING THE DIGITAL DIVIDE

It is a widely accepted fact that education is important in the developmental process of any nation. The Internet is a valuable source of education due to the fact that computers are tools that facilitate quality education and in order to achieve the maximum productivity from an educated person, he/she is employed to access great amounts of information over the Internet (Ofosu & Asgill, 2003). Proper educational infrastructure plays a significant role in society because it enhances an individual's character and self-belief which are attributable to genuine values instilled within them when young. As a result, it enables them to set high expectations for themselves.

According to National Telecommunications and Administration (2013), information tools, such as the computer and the Internet are increasingly critical to economic success and personal advancement. It is clear that minorities who are uneducated living in low-income areas are among the group of people who do not have access to information resources. In order to solve this problem, young people need to be provided with basic skills in preparation for them to pursue a profession of their choice, and to develop their mental attitude and critical thinking to prepare them to take on new challenges and to deal with different life situations (Boone *et al.*, 2014). For these individuals to progress in life, they need the tools to acquire information through the education process. This is all possible through computers that have revolutionised the education system (Ofosu & Asgill, 2003). Therefore, it is imperative for any government concerned with the welfare of its people to provide them with assistance in the form of loans for technological equipment so that they are able to achieve high academic standing and in doing so achieve a high standard of living (Ofosu & Asgill, 2003).

2.6.1 CLOSING THE DIGITAL DIVIDE WITH PAIR PROGRAMMING

The attitude of women towards CS is one of several factors that explains their low participation in computing. Women's negative attitude towards CS and IT can be attributed to their low confidence and programming abilities. Based on the findings of Liebenberg *et al.* (2012), pair programming

made a positive contribution towards their enjoyment and the perception that high school girls have of IT. The girls in the study experienced increased motivation, focus and encouragement to pursue a degree in IT and start a career in the CS industry. Thus, pair programming could potentially provide an opportunity to dissipate this negative attitude towards computer programming (Baser, 2013). Research shows the benefits of pair programming for the retention of minorities, as well as performance in coding. Educators and researchers across the globe want to change the face of computing learning. Pair programming will make learning how to code accessible to all students at a young age (Werner *et al.*, 2013). Even though pair programming has shown to be beneficial for students to learn coding, more research is needed on which pair programming qualities are most beneficial in terms of effective code learning (Werner *et al.*, 2013). These identified qualities of pair programming may support black South Africans who are the underrepresented minority in IT and CS in South Africa.

2.6.2 CODING INITIATIVES IN SOUTH AFRICA

According to Lebo Miya, Managing Director of Ekasi Code Club, the innovation and creation of new ideas is important for South Africa and it is made possible by coding. The Ekasi Code Club is a company that takes in youth who are unemployed and turns them into junior developers (Gool, 2018a). A study performed at the University of Houston, focused on the impact of computer programming on cognitive outcomes, the results found that the students with knowledge of computer coding score much higher in varied cognitive ability tests (Gool, 2018b). These findings support the benefit of coding in improving a child's basic logic. According to Moira De Roche, a training specialist, teaching coding improves children's logical-thinking and develops their problem-solving skills (Rawlins, 2015). In their studies, Trends in International Mathematics and Science Study (TIMSS) have revealed that the public education system in South Africa ranks very low in comparison to the rest of the world in key skill area subjects, such as mathematics and science (Mzekandaba, 2016).

Computer Applications Technology (CAT) and Information Technology (IT) are the only two computer-based subjects that are taught in high schools in South Africa. CAT teaches pupils the use of programs, such as Microsoft Word and IT teaches them computer programming (Rawlins, 2015). South Africa struggles to include coding in its curriculum, thus not providing enough educational resources regarding coding. Less privileged youths in South Africa attend schools in their townships that only focus on implementing the basic school curriculum. Most of them only

get to use a computer for the first time when they study IT at universities or colleges. The lack of proper infrastructure in these schools makes it challenging for students to apply to universities and aspire to become IT specialists. Slowly but surely, young South Africans are receiving the necessary exposure to the digital sector by coding initiatives, such as CodeSpace, WeThinkCode, Code for South Africa, and GirlHYPE, to name a few. These initiatives offer them opportunities to code programs. The benefit of teaching children computer coding at an early age is that it will require less teaching as they advance in school. With the knowledge of coding at hand, all the children have to do is to learn new programming languages (Rawlins, 2015). According to Claire Jowell, the school curriculum should be industry-led, based on the industry's demands and trends that are updated on a regular basis (Gool, 2018b). There are many useful online coding courses on the Internet for those who want to learn how to code. South African youths merely need access to online coding content to ensure that they will master core skills in order to be employed in the world of software development.

2.7 SUMMARY

The cause of the historical underrepresentation of ethnic minorities in STEM fields in South Africa, the digital divide and the closing thereof, the presence of women in IT and CS and coding initiatives in South Africa were reported on and discussed in this chapter. In Chapter 3, the history of pair programming, its use in the industry and the implementation thereof in the classroom are discussed. In addition, the necessary guidelines for pair programming are detailed in-depth.

CHAPTER 3 PAIR PROGRAMMING

3.1 INTRODUCTION

Pair programming is one of the twelve Extreme Programming (XP) main practices created by Kent Beck in the late 1990s (Gómez *et al.*, 2013). Pair programming is an example of an XP practice, where two programmers collaborate at the same computer to accomplish a task and in the process, both programmers acquire knowledge and experience from each other frequently (Karthiekeyan *et al.*, 2018).

Pair programming requires two programmers to work together side-by-side at one station to solve the same problem or accomplish the same task. Essentially, it can be viewed as two programmers who are working at two different levels of abstraction with two distinct roles. Laurie Williams and Robert Kessler made the idea of programming in pairs more definite by using the driver-navigator metaphor (Wray, 2009). Amongst the two, one programmer assumes the role of the “driver”, who is responsible for the task of coding by operating the keyboard and mouse, while another assumes the role of the “navigator” who is tasked with observing and analysing the driver's code and in doing so, searches for possible solutions (Gómez *et al.*, 2013). In this chapter, the researcher discusses how pair programming originated in the industry, its use as an agile software development technique in the industry and as a teaching methodology in the classroom.

3.2 PAIR PROGRAMMING IN THE INDUSTRY

3.2.1 DEVELOPMENT OF SOFTWARE

In the modern world, software plays an important role in our daily lives and software development has always been regarded as a difficult task to perform. In the software development industry, the delivery of a high-quality software program is the primary objective for the developers involved. The development of low-quality software is as a result of the lack of planning and a systematic approach towards the development of software. A software development approach is crucial in dictating the quality of the software being delivered. Many different development methodologies are proposed by developers and researchers alike to guide the process of software development in its entirety. The term methodology by definition is the collection of methods, procedures and standards that define an integrated synthesis of engineering approaches to the development of a

product (Parikh, 2019). Producing applications and/or products in a cost-effective manner with few errors as well as high usability, is the main reason for software development. Current agile software development approaches emphasise perfecting functionality requirements and technical implementation rather than developing software that meets the needs of customers (Sohaib *et al.*, 2019).

3.2.2 AGILE SOFTWARE DEVELOPMENT METHODOLOGIES

The need for a sound software development approach led to the creation of several software development life cycle models by software practitioners (Matharu *et al.*, 2015). A Software Development Life Cycle lays out the framework for the planning, in addition to the controlling of the development or the modification of software products, accompanied by the methodologies and models that are used for the development of software (Roebuck, 2012).

In response to the ever-increasing complexity of software and dynamic user requirements, the software development industry has transitioned from traditional software development models to agile-based development (Matharu *et al.*, 2015). Agile-based development methods have shorter development cycles, higher customer interaction, incremental delivery, frequent re-designs with accommodation of changes necessitated by the dynamic user requirements (Dhir *et al.*, 2019; Dingsøyr *et al.*, 2012; Matharu *et al.*, 2015). In the last few decades, agile software development models have gained great popularity attributable to their simplicity, flexibility, and sustainability to the various requirements of software development. There are many agile approaches used by software development organisations and businesses alike, for example, there is Extreme Programming (XP), Scrum, Feature-driven Development (FDD), the Dynamic System Development Method (DSDM), Kanban, Lean Software Development (LSD), and Adaptive Software Development (ASD). Scrum and Extreme Programming, also called lightweight development methodologies, are the most widely used agile development models and are suitable for small-scale projects (Anwer *et al.*, 2017b).

XP is a collection of the best principles, values and practices that are suitable for small- to medium-sized teams. Kent Beck's twelve practices include the planning game, small releases, metaphor, simple design, continuous testing, refactoring, pair programming, collective ownership, continuous integration, 40-hour week, on-site customer and coding standards (Beck, 2000). The

values that are the core of XP are communication, simplicity, feedback, courage and respect (Parikh, 2019).

3.2.3 EXTREME PROGRAMMING EXPERIENCES IN THE INDUSTRY

Most enterprises are enthusiastic about investigating and evaluating new management strategies and process brought forth by the new dimension opened by e-business (English, 2002). XP was introduced by Kent Beck in 1996 while leading C3 (Chrysler Comprehensive Compensation) which was a successful long-term project in order to re-write Daimler Chrysler's payroll system. Kent Beck wrote the book "Extreme Programming Explained: Embrace Change" where he describes XP as a collection of highly disciplined practices that companies can initiate formally into an already existing process or use in addition to individual processes for project management, change management, requirements planning, and testing (English, 2002). The C3 project commenced in January 1995. Kent Beck was sought out a year later to aid with performance tuning for the already existing payroll system. Beck discovered code that was factored poorly, no repeatable tests were performed and that the current team and management had lost confidence in the project. Beck developed a schedule for the team to follow, rules for testing were established as well as accepting the standard, tried and tested pair programming technique while incorporating user stories from the customers (Beck, 1999). At the end of thirty-three weeks the development team had a system ready for performance tuning and that increment was launched mid 1997 which was not as soon as they had hoped. The first full XP project had several new system features added to it throughout the life cycle, but this long development experience had fallen short of the promises made to management, and the customers because the team failed to follow all the XP principles (Beck, 1999).

In 1993, Ford launched the Vehicle Costing and Profit System (VCAPS) which is an analysis tool that produces reports on production revenues, expenses and variable costs, such as labour hours (Beck, 2000). The VCAPS project posed a challenge to Ford in the sense that the team was unable to keep up with the ever-changing requirements and that the system was time-consuming when processing input manually prior to producing the final output. Ford solved these problems by adopting XP practices. It offered the Ford development team the agility to meet constantly changing requirements on time and deliver a product of high quality. After a year of adding automated testing units that enabled continuous integration and small releases, the team reported a 40% drop in bug reports (Beck, 2000). A year and a half later, the developers used pair

programming which was awkward at first but the team was able to decrease the number of releases and system failures. As a result, the customer had a stable system which indicates that it is successful.

Iona Technologies failed to adopt one of the twelve practices (40-hour work week) and were hesitant to use pair programming. However, they found that XP practices were the best way to maintain code for software re-engineering (Erickson *et al.*, 2005). Another experience report stated that an in-trouble project by ThoughtWorks was saved by the development team implementing XP practices. Similarly, they were also a bit hesitant about the use of pair programming and chose not to switch to a 40-hour work week, but these practices ensured that they delivered the customer's final product on time (Erickson *et al.*, 2005). Motorola conducted a study that partially opted for the development of mission-critical software. The development team made notable changes to a legacy code base in order to develop a new release. The Motorola development team saw an increase in productivity, higher velocity and far less defects in comparison to company averages (Erickson *et al.*, 2005). In France in 2009, a survey was conducted, by The French Scrum User Group, on the usage in addition to the usefulness of agile-based development methodologies. The respondents of the survey were 230 professionals from over 150 companies. The survey concluded that 52% of the companies recommended the use of the XP approach (Matharu *et al.*, 2015).

The experiences from the aforementioned projects indicate that some of the twelve XP practices can be useful in large companies. Most importantly, it must be implemented in such a manner that they fit into the company's processes and structure. XP projects may dynamically prioritise requirements and implement additional features. This allows the development team to be flexible to compromise and commit to a schedule that is many months away (Harrison & Stokes, 2017). Other XP practices, such as pair programming may be adopted by a company if it fits with the project's processes and culture and its use is recommended. Not all projects follow all the XP practices and the reason for deviation is that certain XP practices are not appropriate for large-scale software projects and they may not fit with other practices in existing projects (Harrison & Stokes, 2017). XP greatly emphasises the implementation of its practices to the extreme level in order to obtain a high-quality end result. XP is suitable for small development teams; however some of its practices may slow down the development processes which indicates that teams

should only adopt practices that fit with the process and culture of the project in development (Anwer *et al.*, 2017a).

3.2.4 PAIR PROGRAMMING AND EXTREME PROGRAMMING

In the 1990s, the introduction of pair programming as a practice was part of the Extreme Programming (XP) software development methodology. Currently, it is a vital element of XP software development methodology in the Information Technology industry (He & Chen, 2015). In the IT industry, programming in pairs is used by many software development organisations in modern times. In software development, pair programming is an obligatory technique for all work that involves development because the expectation is that it stimulates communication and increases the quality of code among development teams. Large software systems can only be built by the participation of a large team of software developers. Plenty of communication is required for teamwork to be successful, but the programming of a large software system is divided into tasks that are separate and are completed by individuals alone (Vanhanen, 2011).

In essence, pair programming focuses on the principle of sharing expertise across the teams involved in a project. It is important to keep the developers in a team working on a project focused which is essential in such a knowledge-based activity. Several studies have been conducted investigating how the different levels of attention affect the quality of work. Software developers are exposed to the risk of losing attention due to their work being performed on a computer. Limited attention can cause lower quality work, and an overall decrease in social ability (Sillitti *et al.*, 2012).

This decrease comes with negative consequences for the entire software development process that may have a severe impact on the interactions between colleagues and customers when requested changes need to be made. To increase attention levels, an external stimulus can be introduced to keep attention high or an internal cognitive control mechanism that keeps attention aligned with the priorities of the objectives that need to be completed (Sillitti *et al.*, 2012).

Existing research on attention levels only focuses on individual developers who are externally driven. Ignoring the fact that the nature of work, the work environment and the pair compositions between the development teams influences how developers handle external and self-initiated interruptions (Di Bella *et al.*, 2012). Pair programming can be an external stimulus because of the

two developers working together and communicating (exchanging ideas, gestures and emotions). As an internal cognitive control mechanism in software development, pair programming is also effective because the developers feel morally obliged not to waste time. They want to always deliver their best work and they do not want to look incompetent in front of their development team, and the developer with whom they are paired (Sillitti *et al.*, 2012). Ethnographic observations made by Chong and Siino (2006) indicate that factors, such as interruption length, content, type, occurrence time, and interrupter/interrupted strategies differ for pair programming versus solo programmers.

When working with large teams, there is a great possibility that many problems can arise. These problems that are associated with large teams can have a negative impact on the overall performance of a team (Pieterse *et al.*, 2012). Empirical evidence suggests that two developers working as a pair are twice as effective compared to two individual developers working alone in terms of the speed and the possibility of finding a solution for a problem (Karthiekheyan *et al.*, 2018). Over a period of 10 months, an industrial case study was conducted at an IT department of a large Italian manufacturing company by Sillitti *et al.* (2012) where a team of 17 software developers worked in pairs during observation. This case study, in particular, aimed to determine how pair programming affects how much developers stay focused on their tasks. It was found that the developers spent more time in directly productive activities and switched less often between tools within specific work sequences. When developers switch tasks, they tend to gather information required quicker. Notably, they stayed more focused on a task for a longer amount of time before switching to other tasks.

Pair programming is an interesting feature of XP and is one of the twelve practices that distinguishes the XP methodology. Programmers write their code and integrate it together, meaning that the development team as a whole has collective ownership of the application code. This allows each development team member to change any part of the code (Dalalah, 2014). As beneficial as this practice is, it can be detrimental to the development process of a project if it is implemented in the wrong environment. Pair programming and its effect on attitude changes, functionality and solution quality should be taken into account for the development of an application.

Altogether pair programming is effective in the work environment because the developers are constantly brainstorming, changing roles between each other through reviewing their code which reduces the number of defects to a great extent. This results in the development of a quality product without errors. In the industrial data analysed by Sillitti *et al.* (2012), there is strong evidence that pair programming aids in the elimination of workplace distractions and significantly increases the focus developers have on their work. Pair programming has risen to popularity alongside Extreme Programming in both the IT industry and educational fields. With the increased popularity of agile software development methods, this collaborative form of programming has become a great alternative to the traditional practice of solo programming (Aguirre-Urreta *et al.*, 2015).

3.3 PAIR PROGRAMMING AND THE CLASSROOM

3.3.1 INTRODUCTORY PROGRAMMING COURSES

The exponential growth of ICT has a great impact on education in addition to the capacity of teaching and learning (Isong *et al.*, 2016). Programming is an essential part of the Computer Science problem solving and design process, and a central feature of most introductory Computer Science curricula is learning to program (Ghobadi *et al.*, 2017). The lecturers and researchers in Computer Science are constantly trying to come up with better and more effective ways of teaching programming. Several studies have been conducted about the state of programming learning and the results thereof are of great concern (Kaila *et al.*, 2015). Students' active learning in the form of physical motor movement is positive for knowledge retrieval. These positive effects are conceptual and practical (Von Hausswolff, 2017). Active learning is considered to be a valid method for engaging students. In education, it leads to improvement in results and motivation (Freeman *et al.*, 2014).

Introductory programming courses are regarded as one of the most challenging courses for beginner programmers (Malik & Coldwell-Neilson, 2017). Teaching introductory programming classes is listed as one of the seven challenges in Computer Science Education, due to it being a multifaceted process that involves gaining knowledge of the syntax and semantics of a programming language and developing general problem-solving skills (Umapathy & Ritzhaupt, 2017). It is a complicated skill that requires computational thinking, extensive programming

language knowledge, in addition to the expertise to differentiate, merge, and manipulate appropriate components to solve a problem (Inventado *et al.*, 2017).

To teach programming effectively, practicing is a strategy that encourages skill mastery when students receive adequate experience and feedback. Solving problems is a sought-after skill in pursuit of being successful in learning. Typically, introductory programming classes for undergraduates require the students to individually complete assignments that are content-specific. Programming courses require students to spend a vast amount of time practising which weakens their academic motivation in the course. Therefore, many students think that programming is comparatively difficult, which as a result leads to poor academic performances and also contributes to a high course withdrawal rate. Novice programmers have to simultaneously focus on various skills, such as comprehension, strategies to solve problems and program design (Malik *et al.*, 2019). Many novices do not have adequate computational thinking and problem-solving skills when utilising key programming concepts to express their solutions in programming code (Koulouri *et al.*, 2015). “Computer programming courses require strong cognitive ability, rigorous logical thinking as well as abstract conceptual understanding” (Dou *et al.*, 2017).

Many studies have been performed in the past to determine the challenges and difficulties that novice programmers face. The two major factors for the high failure rate associated with introductory programming courses are the teaching methods and curriculum organisation (Malik *et al.*, 2019). The development of software is incremental and iterative, meaning that a software project goes through several iterations before successful completion. In the classroom, software development is traditionally solitary and non-incremental. This form of programming is not representative of what happens in the industry which poses a major problem for educators (Umapathy & Ritzhaupt, 2017).

Iqbal and Harsh (2013) pointed out that the traditional approach to introductory programming places its focus mostly on programming language syntax and semantics. According to Kaila *et al.* (2015), pair programming is an example of active learning. Students who work and participate in a pair transfer knowledge actively through pair programming.

In their study, Umapathy and Ritzhaupt (2017) established that pair programming had a significant and positive effect on students' grades on programming assignments when compared to solo programming. According to the results of a study on pair programming improving student retention, it was reported that students who used pair programming in introductory Computer Science courses were more likely to remain enrolled in the course three years after the study was conducted. These same students were also more likely to have a Computer Science major in comparison to the students who used solo programming (McDowell *et al.*, 2002). Previous research findings demonstrate that the use of pair programming in introductory programming courses provides notable benefits to programming overall (Smith *et al.*, 2017; Williams & Kessler, 2000a; Williams & Kessler, 2001).

3.3.2 A LOOK AT PAIR PROGRAMMING

Pair programming is a technique where both students work together closely and attentively, writing down the things that they have to do and pointing out parts of the code on which they are working on the computer screen. The parties are required to change roles during the period of programming frequently in order to play to their respective advantages. Working in pairs entails cooperation, consultation, exploration and the exchange of knowledge by the two students (Dou *et al.*, 2017).

Laurie Williams and Robert Kessler conducted a comprehensive research study that entailed the effects of pair programming on a student's learning style, ability, personality, in addition to other factors (Williams & Kessler, 2001). The research findings suggest that pair programming can be a useful approach to teach computer programming in higher education to facilitate effective knowledge sharing among the students (Kavitha & Ahmed, 2015). Thus, pair programming in education reaps more benefits in comparison to pair programming implemented in the IT industry. Research shows that having two students sharing a computer while programming can increase their learning and retention in a Computer Science major. In addition, it instils a sense of belonging among the underrepresented students (Lewis & Shah, 2015).

Pair programming is widely used in the classroom and it is introduced typically in tertiary education (Zarb & Hughes, 2015). It is seen as a valuable technique in the classroom due to the number of identified conditions and interactional forms conducive to learning in collaborative contexts (Begel & Nagappan, 2008; Lewis, 2011). Pair programming does not deter individual student

performance. The pairing of students has shown that students are more likely to complete courses that are related to Computer Science, as well as to achieve satisfactory grades for their practical assignments when compared to the performances of their solo counterparts (Zarb & Hughes, 2015). An empirical study conducted by Kavitha and Ahmed (2015) found that students exposed to pair programming in the classroom reported that working with a partner improved their comprehension of unfamiliar programming topics. Dewi *et al.* (2021) from their research found that there is a significant positive influence between the variable pair programming learning model and student learning outcomes. As a result, it means that through the application of the pair programming model, the outcomes of student learning can be improved.

In terms of learning programming in academic environments, pair programming has been considered as a promising strategy. Students tend to participate and interact more with fellow learners when they engage in collaborative activities such as pair programming in comparison to individual programming. Studies conducted by Williams and Kessler (2003) have revealed that pair programming complemented the learning process significantly within a short period. Furthermore, it helped students gain practical software development experience due to them discussing the work with each other through sharing knowledge and experience in the process of collaboration.

According to Pieterse *et al.* (2012), team success, and levels of participation are influenced by the distribution of academic ability among team members. University students are more compatible when they are of the same gender, perceive their partner as having an equal or higher programming skill level, share the same work ethic and have different learning styles (Denner *et al.*, 2014). In terms of academic alignment, a team that consists of members with similar academic skills and abilities experiences lower occurrences of conflict due to having a sense of balance between them (Pieterse *et al.*, 2012). The educators can allow students to choose desired partners or the educator can be proactive and form student pairs that are most likely to cooperate well together. To form pairs that are highly compatible, educators are encouraged to pair students together who have the same level of programming skill as measured by a total grade for a previous Computer Science subject. In addition, in order to attain success with pair programming it is important to understand the social aspects of pair programming (Cockburn & Williams, 2000). An individual's personality can be measured across quantifiable parameters grouped under a set of dimension often specified in a framework (Poonam & Yasser, 2018). There are several

personality assessment methods such as the Myers-Briggs Type Indicator (MBTI), Eysenck Personality model and the Five Factor Model (FFM) (McCrae & John, 1992; Poonam & Yasser, 2018; Salleh *et al.*, 2010b). The MBTI loosely inspired by Carl Jung's psychological types, is one of the most widely-used and suitable personality assessment tools for measuring personality traits of programmers (Hannay *et al.*, 2009; Poonam & Yasser, 2018; Salleh *et al.*, 2014). Educators can use the Myers-Briggs personality test to form pairs. Students with a similar work ethic can be identified by prompting students to provide a number from one to nine which indicates how hard the student works, nine indicating that they work hard to get the best grade (Williams, 2010a). It is up to the educator if he/she wants the pairs to stay together for an assignment or remain the same for the duration of a semester.

Studies have proved that collaborative learning in the form of pair programming establishes an environment that is conducive to a more advanced level of learning and active social engagement, which as a result produces more confident and interested students and also decreases their frustration levels (Williams, 2010a). Pair programming fosters a communal and supportive working environment due to the social interaction with a peer in the computer laboratory. This learning approach is beneficial for Millennials due to this current generation placing its value on collaborative environments (Oblinger, 2003). Furthermore, the collaborative aspect within pair programming gives the students exposure to teamwork and reinforces communication skills that are necessary in the IT industry.

3.3.3 IMPLEMENTATION AND USE OF PAIR PROGRAMMING

Under the assumption that a pair is compatible, pair pressure is a positive form of pressure that students in a pair put on each other. The students tend to work smarter and harder on a programming assignment due to the fact that they do not want to let their partner down (Williams & Kessler, 2000b). This form of pressure instils focus into the pair because they will not be distracted by surfing the web, making phone calls or reading text messages. Pairs work more intensively for a few hours than they would if they were alone because there are no interruptions (Williams & Upchurch, 2001). The presence of their partner by their side makes recovery after an interruption occur quickly and also returns the level of focus required for the primary task at hand (Lee & Ahlswede, 2007). Pair programming needs coordination that is scheduled, it forces students to work intensively, and the explicit accompanying deadlines highly motivate the students to complete the task at hand during the practical session (Williams, 2010b).

In pair programming, brainstorming is important because this is how a pair provides a solution to a problem together. When a pair works at its best, each student brings his/her skill set, ability, and perspective to the table to achieve a shared goal and consequently complete the task put before them (Williams, 2010a). Each student has a different outlook and therefore an alternative way of approaching a shared problem. The students must brainstorm ideas and evaluate more alternatives on how to approach the problem (Williams, 2001). This is beneficial because when working alone, one tends to settle for the first approach that comes to mind. Furthermore, in a pair, students consider each other's ideas and include their suggestions in the determination of the best solution for a problem (McDowell *et al.*, 2003; Nawrocki & Wojciechowski, 2001). The navigator in the pairing also plays the role of a reviewer, continuously pointing out any syntax or semantic errors in the code on a regular minute-by-minute basis (Jones & Fleming, 2013). Pair reviewing is also helpful for picking up unconsidered cases in program design that may go unnoticed when worked alone. The navigator can mitigate problems even before they can occur. The review process in the pair in the form of brainstorming helps to avoid careless errors that a programmer working alone would unknowingly make. It also saves time for debugging to uncover and fix bugs (Williams, 2010a).

When students have a partner, it builds their courage tremendously. Students receive a boost of confidence to accomplish tasks that they might be afraid to do alone due to the affirmation that is gained from their partner (Williams & Kessler, 2000a). It is also helpful because a partner is courageous enough to admit when he/she does not know the syntax or a programming language rule. Students who work by themselves become embarrassed when they do not know something and will desperately try to figure something out on their own instead of reaching out to a peer for assistance (Williams, 2010a). When two students do not know something, they simultaneously realise that it is better to seek help from an educator or assistant. Pair learning is a critical part of using pair programming in education because knowledge is constantly being transferred between the two students. The knowledge being passed varies from usage tips for programming tools as well as programming language rules. Unspoken skills and habits are shared by the students during the session (Cockburn & Williams, 2000). Depending on the setting, students can rotate and work with new partners and in doing so, share knowledge and new skills from the new

partnership. Switching partners on a frequent basis throughout the semester is an effective way of spreading knowledge around in the class (Denner *et al.*, 2014; Williams & Kessler, 2003).

Initially, the principles of pair programming are unfamiliar to students and it is important that educators thoroughly introduce them to the principles of pair programming. The students involved are required to cooperate closely by discussing and helping each other with a task, and frequently exchange roles to achieve the objectives as a cohesive unit. It is important that the pairing of the students is conducted in a controlled environment with the teacher actively involved in the correct implementation and use of pair programming principles by protecting the interests of the student pairs with strict participation and withdrawal rules (Dou *et al.*, 2017). It is important to note that the effect of pair programming is strongly influenced by personality types and individual abilities (discussed further in Section 3.6.3). The students' programming capabilities can be evaluated by the quantity of code written, and determining the level of ability the students possess. Previously, researchers have identified that personality, gender, and experience in programming affects the efficiency of pair programming. The most important priority in software development is the problem-solving and programming ability of a programmer. The quality of code and programming efficiency can be improved by means of organising and selecting partners based on their personality and ability (Dou *et al.*, 2017).

Plonka *et al.* (2015) recommend a partner combination that consists of a senior and a junior developer or at the least a pair with complementary knowledge. While participating in pair programming, developers learn from one another, particularly when a junior is paired with a senior developer; they learn quicker (de Lima Salge & Berente, 2016). Furthermore, in the industry, and in the classroom, programmers will pair up with different programmers throughout the semester, and as a result, this cross-pollination with different sets of pairs will improve the entire class (de Lima Salge & Berente, 2016).

3.3.4 PAIR PROGRAMMING AND COMMUNICATION

Pair programming consists of both verbal as well as non-verbal forms of communication, which makes it a highly communication-intensive process (Sharp & Robinson, 2010). Effective communication is paramount to success when it comes to teamwork. The silence within a pair should be considered as a signal of danger (Williams & Kessler, 2002). This style of coding requires constant communication for it to be successful in academia and the industry. Thus, pair

programming's success is due to the collaborative way in which assignments are completed. In a professional setting or even in an academic one, there is a necessity for constant communication between the individuals programming together in a pair. This necessity in pair programming is crucial because it benefits the exchange of ideas, problem solving, and feedback during the collaborative programming process taking place between two programmers. According to the responses from industrial developers, surveyed by Begel and Nagappan (2008), a good programming partner is a person who is a good listener, is articulate, enjoys debating and discussing code, eases the thought process through constant verbalisation, and is not afraid to ask questions and deliver own opinions.

Communication is intrinsic to the pair programming process. There is a positive correlation between the number of communication transactions within a pair and the pair's productivity (Zarb & Hughes, 2015). Other studies find that there is no correlation between a high level of communication and compatibility between partners (Choi *et al.*, 2009). Communication in this context is still undergoing investigation in terms of the final product and the process. Stapel *et al.* (2010) hypothesise that there could, in fact, be a difference between the level of communication between novice first-time pair programmers and industry pair programmers. They believe that this is caused by the more experienced industry pair programmers being more at ease with communication compared to the novice pair programmers that may be concerned about the consequences of sharing the wrong ideas during a paired practical session. The communication or the lack thereof within a pair might determine the success of a pair programming assignment. People differ from each other and therefore they individually possess different behaviours and opinions. It is imperative to the success of pair programming to understand how two students can best work together (Lewis, 2011).

3.3.5 GUIDELINES FOR EDUCATORS IN THE PAIR PROGRAMMING CLASS

The structure of a class can fail to encourage a consistent pair programming environment (Bevan *et al.*, 2002). Therefore, several guidelines must be used as a framework by educators interested in adopting pair programming (Zarb & Hughes, 2015). There are no scientifically-derived guidelines as yet for novice programmers instructing them on how to cope with the various scenarios that might be encountered during a practical pairing session (Zarb & Hughes, 2015). However, based on seven years of experience with more than 1000 students, Williams *et al.*

(2008) outline guidelines for the implementation of pair programming in the classroom setting. Williams *et al.* (2008) contend that:

- Students require training to pair successfully.
- Educators should check in on the pairs to ensure effective teamwork and address questions, but students are encouraged to find answers independently.
- Attendance should be required at each practical session and penalties for tardiness should be applied to the less diligent student.
- Educators should ask students for feedback about their programming partners and take action promptly when a partner is not performing equitably.
- Students must be encouraged to immediately report any problems so that the educator can intervene and avoid conflict.
- Students should be evaluated on paired and solo work.
- Educators should make every attempt to assign compatible partners. Throughout the course of the semester, students should be assigned to different partners.
- Paired students should be able to sit comfortably next to each other and have easy access to the computer screen, keyboard, and mouse.
- The pair should strictly be working together on the same problem.

When carrying out pair programming in a computer laboratory, educators need to determine how to effectively form pairs and observe the behaviour of the paired students. The educator must clearly describe the requirements of the practical assignment and prompt students to exchange roles at regular intervals. The educator may ask the students to pair with different partners on different occasions to reflect on the fairness of the pairing.

A large empirical study of 1 350 students was conducted to examine the factors that an educator can use to proactively form pairs that are most likely to be compatible (Williams, 2010a). The study has indicated that heterogeneous pairs consisting of a male and female, brought forth creative and high-quality solutions. If the rotation system of pairing is implemented correctly, the students' benefit from this practice will be even greater. Students learn more about development tools and solve problems quicker through communication between the students which results in effective knowledge management.

3.4 CHALLENGES OF PAIR PROGRAMMING

Pair programming has undergone extensive studying in the learning sciences and the results from research have proved that programming in pairs encourages the discussion of ideas among small groups and it also fosters critical thinking and engagement (Shi *et al.*, 2019). However, pair programming is not always implemented correctly and the driver and navigator roles do break down in practice. Pair programming is an effective practice, but it does pose several challenges. Williams (2010a) has identified four pair programming challenges, and how to overcome them.

- Traditionally programming is an individual practice, and it can hamper a skilled programmer from concentrating when working with a partner. The solo programmer may feel like programming in a pair is a waste of time, especially if their partner has less programming skills. The same applies to software engineers in the industry. However, those who successfully implement the principles of pair programming tend to then prefer working in a pair than working alone. To overcome this problem, educators are encouraged to ensure that pair programming principles are implemented correct and students are paired with a partner of a similar programming skill level.
- Scheduling can make it difficult for students to meet with their partners because students are often busy and may have conflicting class schedules. Scheduling is also perceived as a challenge for in the industry due to time conflicts as it reduces employees' choice of working hours. To solve scheduling problems, the educator should take attendance at every programming practical sessions.
- Incompatibility can be cumbersome task for the student that possess a higher programming skill level than his/her partner. It can lead to a lob-sided partnership where one student does all of the work required to complete an assignment. To solve this problem, the educator assigns the students in a classroom with a different pair programming partner for each practical session.
- Programming from a remote location may make students feel as if they are unable to program in a pair. There are students that prefer to work from their dorm rooms and other students that are registered for long-distance courses might not be able to physically meet and work with their partner. Similar to developers in the industry, these students depend on the Internet to program by making use of web-based tools. To overcome this problem,

the educator should pair long-distance students together in order for them to work remotely.

Here follows another challenge identified by Hanks *et al.* (2011):

- Computer science educators have expressed concern that students who learn to program in their introductory programming course in a pair, will not be able to program alone in later computer sciences courses. However, students at two institutions indicated that pair programming helped them to learn to program and provided them with a sturdy foundation for situations when they have to program individually. It is recommended that students in later programming courses be given clear instructions about appropriate and inappropriate collaboration.

3.5 THE ADVANTAGES OF PAIR PROGRAMMING

There are many benefits of pair programming reported for novices as well as and experts. These benefits include the pair experiencing greater enjoyment of the programming task at hand, an increased distribution of knowledge, and the production of high quality code (Cockburn & Williams, 2000). Student collaboration in introductory programming courses has many benefits such as exposure to different ideas, as well as an increased level of motivation as suggested by previous research on this topic (Celepkolu & Boyer, 2018). The benefits include: improvement in the quality of the students' code in terms of functionality and readability, solutions to a programming task are found faster, and an increased retention rate in the programming course, and an increase in the enjoyment of learning to program for the students, and significant improvement in students' self-confidence. Some studies have suggested that pair programming is especially beneficial for female and minority students (Braught *et al.*, 2011). The benefits of pair programming are time, quality, knowledge transfer, pair pressure, satisfaction, confidence, perception and workload reduction. In the context of education, these benefits are described as follows:

3.5.1 TIME

Pair programming can lead to people incorrectly assuming that the technique can only be implemented during code development, but actually programmers can pair up at any time during all development phases of a project. Paired programmers work more efficiently and productively making the whole programming experience more enriching (Benadé & Liebenberg, 2017). Pair

programmers take less time to complete set programming tasks. The pairs also experience a skewed perception of time, feeling that they worked for less time than they actually did (Zarb & Hughes, 2015). However, programmers report that programming in pairs consumes less time than it would take to produce the software programming individually. Less time is also spent on irrelevant tasks. Working in a pair forces them to concentrate more on the tasks (Demirşahin *et al.*, 2012). Otherwise, programmers feel that they are wasting their programming partner's time. The time spent communicating between teams and programmers involved in the development of software is less due to the established communication channels and pair programming protocols. (Demirşahin *et al.*, 2012; Williams & Kessler, 2000a).

3.5.2 SOFTWARE QUALITY

“Two minds are better than one”. With that being said, one can expect the quality of the software to increase from pair programming. Quality is increased by the additional substantive problem-solving capabilities an additional developer brings and this is enhanced by the constant continuous communication between developers (de Lima Salge & Berente, 2016). Potential program errors are identified earlier because of the navigator's continuous observation and code reviewing, which saves a significant amount of time on debugging (Karthiekheyen *et al.*, 2018). In the industry, re-factoring is a process of continuous design improvement with the aim to keep the product design as simple as possible. Refactoring throughout the life cycle of the project increases the quality of the code (Dalalah, 2014). Better quality code throughout the development process saves time later on in the development life cycle.

In XP, the automated unit tests and customer tests ensure that the application contains all the required features and in doing so, results in better overall product quality (Dalalah, 2014). In the educational context, pairs working together on an assignment produce programs that are significantly better than those of the students programming individually (Zarb & Hughes, 2015). Therefore, pair programming is considered as a good investment both in education as well as in the industry.

3.5.3 TRANSFER OF KNOWLEDGE

There is an exchange of knowledge that takes place between students in the pair programming process (Karthiekheyen *et al.*, 2018). The fast transfer of knowledge is one of the major benefits of pair programming which is widely acknowledged by a range of studies in the industry. For example, a study conducted by Hodgetts (2004) reported that a team with only one database expert decided to make use of pair programming in order to transfer knowledge amongst the developers. In doing so, they managed to learn quickly through pairing with the database expert and as a result they were able to complete the database tasks by themselves. Plonka *et al.* (2015) suggest that a vast amount of knowledge is gained about software development tools. Furthermore, in their study, they established that there is an increase in the knowledge of code, in addition to software systems when using pair programming. Team knowledge is greatly enhanced during daily meetings where all members are involved sharing their experiences and lessons learnt during the development process (Anwer *et al.*, 2017b). In the classroom, students working in pairs can answer each other's questions, rather than consulting their educator as their only source of advice which as a result contributes to the learning process (Zarb & Hughes, 2015). Communicating knowledge is important. In addition to switching roles amongst pairs, people are around in order to avoid the loss of any knowledge that may lead to a coding bottleneck (Dalalah, 2014). This enables the programmers to share knowledge, to learn from each other and also to improve their problem-solving skills.

3.5.4 PAIR PRESSURE

Pair pressure is experienced when two people are working together and it is shown to encourage programmers to communicate with each other, which adds to the pair pressure. Subsequently, the 'pair pressure' causes the students to start work earlier and manage their time wisely (Zarb & Hughes, 2015). The students are more focused and concentrate more on the task at hand and the chances of interruptions and distractions are less. Constant communication leads to fewer misunderstandings because the probability of one student incorrectly interpreting a task requirement is much higher than when there are two students. Students are unlikely to waste time by reading text messages, or surfing the web due to the pressure put on them by their partner, expecting and awaiting continuous input and contribution. Software engineers work harder and smarter for a few hours at a time in a pair and this helps the engineer to recover the state of a primary task after an interruption which, as a result, leads to quick interruption recovery (Williams,

2010a). Pair pressure adds benefits to the pair programming experience, such as increased knowledge distribution and an overall greater enjoyment of programming (Zarb & Hughes, 2015).

3.5.5 SATISFACTION

Kent Beck claims that your project will be successful if you have a reasonable environment in which to work in (Gallis *et al.*, 2003). A strong level of personal commitment along with a high degree of involvement in daily programming activities and programming in a pair, can have an impact on a programmer's satisfaction (Pedrycz *et al.*, 2011). Working together in a pair with a partner and developing a collectively-owned product brings programmers close which leads to them gelling together as a pair (Williams & Kessler, 2000a). The physical environment in which pair programming is implemented may affect the satisfaction of developers. The layout of the office, furniture arrangement, noise, and classmates can influence the motivation of programmers to work (Gallis *et al.*, 2003). Students also experience satisfaction after getting to know their classmates throughout the semester when partner switching takes place in the classroom (Williams, 2010a). Pair programming leads to students who are more satisfied with their work regardless of their programming ability and grade-level (Kavitha & Ahmed, 2015; Williams & Kessler, 2000a). Students who are successful at pair programming are happier, more self-sufficient and are most likely to continue using this practice in future endeavours and they tend to have a preference for it over solo programming. It is important to understand the reasons behind satisfaction when it comes to the management of software development. With that being said, a team member's personal satisfaction is dependent on the team dynamics which is a strong indicator of the product's success, and the achievement of the company's objectives (Lewis, 2011). From the research study conducted by Liebenberg *et al.* (2012) on pair programming and secondary school girls' enjoyment of programming, the findings show that pair programming does indeed positively contribute to the high school girls' enjoyment of programming and Information Technology as a subject.

3.5.6 CONFIDENCE

The quality of student interactions has a direct influence on students' learning during collaborative activities as shown by past research (Grover *et al.*, 2016). Collaborative work cannot be successful by itself; it requires support. Pair programming has shown to increase the confidence of university students in an introductory computer programming course and has since been

regarded as a best practice (Campe *et al.*, 2018; Grover *et al.*, 2016). Programming with a partner builds an immense amount of courage. The programmer gains affirmation from his/her partner which, as a result, gives him/her more confidence to do the things that he/she might be afraid to do when programming alone. Working collaboratively helps programmers to gather a sufficient amount of knowledge to feel confident in what they are doing (Williams, 2010a). Student pairs displayed a more positive attitude in their work than when compared to individual student programmers (Williams & Kessler, 2002). Students agree that they have more confidence in their final solution when it is completed in a pair (Williams & Kessler, 2000a). Research shows that pair programming helps students to solve problems effectively and quickly. Beginners who use pair programming exude higher levels of confidence than their counterparts that program solo (Shi *et al.*, 2019). The high self-confidence in programmers improves their ability to tackle and solve complex programming tasks faced at work or university (Parikh, 2019).

3.5.7 PERCEPTION

Perception is a complex process which involves getting information from the world around you and changing it into a real experience and it also involves other cognitive processes, such as memory, attention and language (Dhingra & Dhingra, 2011; Sharpe *et al.*, 2007). A study conducted on students' perceptions of pair programming by Chaparro *et al.* (2005) found that the programming task difficulty and the students' individual skill level played a major role in pair programming effectiveness. It is important to note that an individual's perception can change while he/she are performing a task. In this study, it was established that students preferred to work with a partner who has a similar skill level in order for them to contribute equally to the pair's project. According to Khomokhoana and Nel (2020), programmers form perceptions while comprehending source code, based on experience only which, as a result, can cause them to ignore all the stated facts in the process of solving a problem. Williams *et al.* (2006) conducted a comprehensive study in 2005 on the effects of student ability, learning style, personality type, as well as other factors on pair programming.

The results of this study showed that personality type did not play a major role in pair compatibility, but the perceived technical competence plays a role in dictating pair compatibility. A study carried out by Lewis (2011) on the perception of difficulty established that on average, students implementing pair programming rated the difficulty of various topics as slightly less difficult. A team member's behaviour is greatly affected by the mutual behaviour of both team members. On

a daily basis, a programmer is required to mediate his/her with someone else's ideas, and behaviour (Pedrycz *et al.*, 2011). Perception allows students to perceive Computer Science as a collaborative endeavour, rather than to isolate the programming process, which is a common misconception that novice programming students have regarding Computer Science (Lewis, 2011).

3.5.8 WORKLOAD REDUCTION

Grading student work is the most important task for a Computer Science/Information Technology educator. The purpose of grading is that it evaluates the students and it also provides the student with feedback on his/her performance (Levine & Walker, 2006). Educators can apply pair rotation to avoid one student saying he/she did all the work while the partner insists on his/her contribution towards the assignment. Pair rotation provides the educator with input from several different partners throughout the semester and this consequently allows him/her to deal with students who are ineffective (Srikanth *et al.*, 2004). Pair pressure causes students to manage their time more efficiently and to start working on an assignment earlier which reduces the number of teaching cases with which an educator needs to deal (Williams & Upchurch, 2001). Pair programming as a teaching methodology reduces the total grading time because it halves the number of submitted assignments, reduces student demands during practical sessions, and it also enhances work ethic due to the sense of mutual responsibility shared between the two students (Salleh *et al.*, 2010a; Williams & Upchurch, 2001; Wood *et al.*, 2013).

3.6 THE DISADVANTAGES OF PAIR PROGRAMMING

Despite the benefits and effectiveness of pair programming as a whole, not all its interactions can be deemed as successful (Celepkolu & Boyer, 2018). Some software developers remain sceptical of their first pair programming experience, and its promised collaborative value. Developers doubt their partner's work habits and the communication demands required by this style of programming (Williams & Kessler, 2000a). According to Cockburn and Williams (2000) and Begel and Nagappan (2008), communication is the most common barrier to pair programming as cited by novices. "Effective pairs chatter; silence is a danger signal" (Williams & Kessler, 2002). Programmers in a pair cannot program without communication; if they are not communicating they are not programming which results in them only reviewing each other's code (Zarb & Hughes, 2015). Articles, such as those by Begel and Nagappan (2008); Nawrocki and Wojciechowski

(2001), Williams and Kessler (2000a), and Williams and Kessler (2002) in the literature prove that communication is one of the main causes of pair programming's failure. The drawbacks of pair programming are miscommunication, pair incompatibility, negative personality traits, environment and scheduling, training and mentoring. In the context of education, these disadvantages are described as follows:

3.6.1 COMMUNICATION

According to Sanders (2002), roughly 50% of first-time student pair programmers reported that most of the forms of difficulty within the pair contributed to communication being the major problem with the pair programming process. Pair programming may inadvertently lead to off-task behaviour or conflict between the two partners (Lewis, 2011). Sfetsos *et al.* (2006) who investigated the impact of personality types on communication and collaboration in pair programming found that upon the data analysis of the communication variable, showed that the productivity of the pairs with the same personality type correlated negatively with communication. Benadé and Liebenberg (2017) reported that students viewed languages a drawback when the pairs in their study comprised students who spoke different languages. It is not surprising because their study on understanding students' experiences with pair programming was conducted in South Africa, a country that has 11 official languages.

3.6.2 INCOMPATIBLE PAIRS

Despite the benefits of pair programming, it was reported to be the most problematic when pairs are incompatible (Layman, 2006). Developers surveyed by Begel and Nagappan (2008) perceived working with a developer with different skills as a challenge. The development process can be slowed down by pairs having incompatible personalities, coding style and/or skill levels (Ying *et al.*, 2019). Williams and Kessler (2002) point out the fact that the pairing of an expert and a novice can cause problems, such as the novice slowing down the expert or the expert not having the correct attitude towards mentoring the novice, and that the expert they may feel that he/she is not getting valuable information from the novice. Incompatible pairs are less successful. They mention frustration, guilt, inability to trust someone else's code and they feel as if they wasted their time and it is a great cause of concern for highly skilled students. They are the least satisfied person when paired with a lesser skilled programmer (Zarb & Hughes, 2015). Each student may

try to force his/her solutions on how a problem should be solved. This can cause conflict when the pair disagrees which is disastrous for the partnership (Benadé & Liebenberg, 2017).

Skilled or top-performing students have the strong desire to work alone and do not want to be slowed down by a lesser skilled partner because they feel that their productivity will be compromised (Ying *et al.*, 2019). When a pair does not function optimally, the pair becomes less productive and, as a result, is unable to complete their practical assignments (Ying *et al.*, 2019). A study on pair rotation in a Computer Science course by Srikanth *et al.* (2004) found that 41,9% of the students stated that pair rotation may lead to the loss of a highly compatible partner which presents the risk of having an incompatible partner for the next assignment (Srikanth *et al.*, 2004).

3.6.3 PERSONALITY TRAITS

Pair programming as a technique, is a practice that involves close social interaction between two people to solve programming problems and it is clear that human-related factors, such as personality have an effect on pair compatibility (Salleh *et al.*, 2014). Students who experience pair programming with an incompatible partner dislike collaborative work, which might be due to a mismatch in psycho-social aspects such as personality and gender aspects or in terms of competency as an aspect, a mismatch in programming skill and experience levels (Lewis, 2011; Salleh *et al.*, 2014). In the context of programming and software engineering, personality traits have been a subject of interest (Salleh *et al.*, 2010b). It is important to understand how personality affects pair programming's effectiveness. The Myers-Briggs Type Indicator (MBTI) carries out personality type evaluation utilising a set of topics that assesses the type of personality each student has. The MBTI assesses an individual's personality traits based on four dichotomies: information (sensing/intuitive), decisions (thinking/feeling), favourite worlds (extraversion/introversion) and structure (judging/perceiving) (Hanks *et al.*, 2011). The FFM consists of five personality traits that provide a structure for categorizing dimensions of difference in human personality (McCrae & John, 1992). The five broad personality traits are (1) openness to experience, (2) agreeableness, (3) conscientiousness, (4) extraversion, and (5) neuroticism (Salleh *et al.*, 2010b). Conscientiousness is concerned with one's achievement orientation and students with low conscientiousness are irresponsible, impulsive and disordered. Students that display extraversion are assertive and may try to force his/her solutions to a problem on their partner. Students are considered to have low agreeableness when they display traits like scepticism, selfishness and hostility. Individuals with high levels of neuroticism are anxious,

moody, nervous and insecure. Students are not open to experience often display a lack of aesthetic sensibilities, preference for routine and tend to favour conservative values (Salleh *et al.*, 2014). These aforementioned negative personality traits can harm teamwork and cause conflicts which consequently makes the pair programming experience rather quite unpleasant for both parties.

3.6.4 ENVIRONMENT AND SCHEDULING

It is difficult to create a learning environment where students in pairs can engage in productive discourse to make progress in their conceptual understanding of programming (Lewis, 2011). Students who do not attend lectures impact their own ability to work productively and fairly contribute to the joint project (Williams *et al.*, 2008). A pair programming partner who is tardy disadvantages his/her partner by not having someone with whom to collaborate. Pair programming takes longer if the partner is not on the same programming skill level, which can be more time-consuming than when working individually, and as a result, lessens productivity as well as efficiency suffer (Benadé & Liebenberg, 2017). It can deter a present partner from using pair programming as a whole if this problem persists.

3.6.5 TRAINING AND MENTORING

When planned projects exceed the capabilities of the current work force, additional programmers are required and hired. Failure to hire programmers with inadequate lead time for sufficient training are of little use when their requirements are needed. Individuals with an impressive grade point average who have taken a wide range of computer programming courses at university still have problems understanding software development methodologies. Familiarity with programming languages does not mean that newly appointed programmers can produce good software design (Poff, 2003). For example, there is no university course that exists which describes payload application's one-of-a-kind framework. In the context of developing firmware for processors, complex specialised domain knowledge is required. Plonka *et al.* (2015) reported that the effect of pair programming training was low. Mentoring is the assignment of senior programmers to train inexperienced junior programmers. This training method is not consistently effective due to the fact that most senior programmers are usually too busy for the effort involved. Some resent mentoring because they are not good at explaining or teaching an inexperienced

programmer it and feel that it is not part of their job description (Ally *et al.*, 2005; Poff, 2003). Consequently, some junior programmers receive mentoring while other programmers do not.

3.6.6 PAIR PROGRAMMING AND NON-COMPUTER SCIENCE STUDENTS

ICT is an indispensable part of the contemporary world. ICT has an impact on the quality, in addition to quantity of teaching, learning, and research in traditional and distance-education institutions (Yusuf, 2005). ICT courses are not acknowledged as conventional programming modules where students learn to code in a particular programming language. Nevertheless, ICT courses and Computer Science (CS) courses have indistinguishable challenges. It is demanding for lecturers and student assistants to tend to every problem that students encounter due to the fact that the classes are large. One of the objectives of an ICT introductory course is to excite students about computing and programming via its dynamic, interactive, and engaging content (Benadé & Liebenberg, 2017; Yusuf, 2005). It also helps relate school experiences to a more realistic image of the collaborative nature of IT and CS jobs (Benadé & Liebenberg, 2017). An experiment performed by Nagappan *et al.* (2003) assessed the efficacy of pair programming in an introductory CS course and the results found that pair programming helped non-Computer Science majors to improve more than their CS major counterparts.

In their research study on the effects of pair programming on individual programming skills, Braught *et al.* (2008) established that pair programming as a learning technique levelled out the playing field in terms of acquiring individual programming skills. They saw students with lower standardized test (SAT) scores achieving better lab practical scores when making use of pair programming (Braught *et al.*, 2008). According to O'Donnell *et al.* (2015) who evaluated the efficacy of pair programming for non-CS students by assessing their attendance and performance, higher levels of attendance were found among non-CS pair programming students, thereby suggesting that pair programming may in some cases improve the performance of weaker non-CS students.

3.7 SUMMARY

An in-depth literature review of pair programming detailing the use of pair programming in the industry, in the classroom, and the advantages and disadvantages of pair programming and the implementation thereof were reported in this chapter. In Chapter 4, the research paradigm, the instruments used to gather data, the students, and the analysis of the data are discussed.

CHAPTER 4 RESEARCH METHODOLOGY

4.1 INTRODUCTION

In this chapter, the research paradigm, design and methodology, and the various constructs of the UTAUT2 model are discussed.

4.2 AIM OF THE RESEARCH

The primary aim of this research study is to investigate the impact of pair programming on the experiences of underrepresented groups in Information Technology in South Africa.

4.2.1 OBJECTIVES OF THE RESEARCH STUDY

In order to accomplish the primary aim of the study, the following secondary objectives are set:

4.2.1.1 To examine the literature on the underrepresented groups in Information Technology;

4.2.1.2 To examine the literature on pair programming; and

4.2.1.3 To investigate the impact of pair programming on the students' experiences and perceptions.

4.3 RESEARCH PARADIGM, DESIGN AND METHODOLOGY

4.3.1 RESEARCH PARADIGM

The study was conducted in the positivistic paradigm to act as a basis that supported quantitative and qualitative research methods (Leech *et al.*, 2010). A paradigm is "a selection of interrelated assumptions made about the social world which produces a philosophical and conceptual framework for the organized study of the world" (Filstead, 1979). It can also be viewed as a set of basic beliefs that deals with first principles which define and represent a certain worldview for an individual, and also provides the individual with a place (a sense of belonging) in this "world" along with the accompanying relationships (Guba & Lincoln, 1994). It consists of four components, namely ontology, epistemology, methodology and methods (Scotland, 2012). Understanding the paradigm-specific assumptions helps to shed light on the quality of the

research findings that support scientific studies and also aids in identifying gaps in generating sound evidence (Park *et al.*, 2020).

Research paradigms serve as guidelines to scientific discoveries by means of assumptions and principles and aid in the selection of tools, instruments, participants, and methods used in the study (Makombe, 2017; Ponterotto, 2005). It is essential to state the research paradigm used in a study due to the fact that paradigms have an influence on the research method, instruments and interpretations (Gringeri *et al.*, 2013).

There are four kinds of research paradigm, namely positivism, post-positivism, critical theory and constructivism (Romani *et al.*, 2018). Based on the literature, there are two paradigms that are positivistic under which post-positivism and anti-positivism fall. The others are normative paradigms which are interpretivism, social constructivism, criticism/critical theory and pragmatism (Makombe, 2017). Positivism, also referred to as the scientific method, can be regarded as a research strategy and approach that is rooted in the ontological principle and doctrine that truth, as well as reality, are free and independent of the viewer and the observer (Aliyu *et al.*, 2014). This study is conducted, using the positivistic paradigm to act as a basis that will support quantitative and qualitative research methods (mixed methods approach).

According to Scotland (2012), ontology can be defined as the study of being, and ontological assumptions are concerned with what constitutes the nature of existence. These assumptions aim to seek a reality to a research question by means of indicating an existing type of knowledge that can be found. This is called Naïve Realism, which claims that there is only one single reality in the social world and this reality can be perceived by a human in a valid manner (Wang, 2020). Thus, there is one true reality that is comprehensible, identifiable and measurable. Positivist ontology proposes that all researched objects reside in an objective or unbiased social world and it comprises human beings.

Epistemology refers to the nature of knowledge. It discusses what we do know in addition to what we can know (Wang, 2020). It is concerned with how a researcher aims to reveal knowledge to reach reality. Moreover, it is also considered as an internal factor for the researcher; how he/she sees the world by distinguishing what is right from what is wrong (Alharahsheh & Pius, 2020). Positivist epistemology is objective in the sense that positivist researchers go into the social world impartially and discover absolute knowledge about an objective reality, meaning that the

researcher and the researched object are independent entities. The aim of the researcher is to obtain meaning that resides in the object, not in the conscience of the researcher (Scotland, 2012).

Each and every paradigm builds upon its own ontological and epistemological assumptions. Therefore, inherently different research paradigms contain differing ontological and epistemological perspectives. Since all assumptions are conjecture, these differing paradigms have differing assumptions of reality, as well as knowledge which underpins their distinct research approaches (Scotland, 2012). These differences are displayed in their methodology and methods.

Igwenagu (2016) defines methodology as the systematic, theoretical analysis of the methods that apply to a field of study, which comprise the body of methods as well as principles that are related to a branch of knowledge. Methodologies involve the extensive research strategy followed in order to conduct research, and as a result, it identifies the methods that are used and that fit with the outlined research strategy. Methodology incorporates key concepts, such as the paradigm, theoretical model, phases and quantitative or qualitative techniques. This provides transparency regarding the means of data collection. However, it does not provide solutions, unlike a method, but directs its focus towards the understanding of which method or set of methods and/or best practices can be applied to determine a specific result (Alharahsheh & Pius, 2020; Igwenagu, 2016). Positivist methodology emphasises the engagement in research in settings where variables are controlled and operationalised. In terms of the social sciences, this needs to create an artificial environment where irrelevant factors that are beyond the study, are minimised (Park *et al.*, 2020). The main focus of this study is to examine the explanatory or causal relationships between the operationalised variables in the study, as also performed in natural sciences which are favoured in the positivist paradigm. Methodology simply identifies the methods that are used in a research study.

Methods, as described above, define the means of data collection or how a specific result is calculated. “Any description of a means of calculation of specific result is always a description of a method, and never a methodology” (Igwenagu, 2016). Methodological assumptions are the main influencers of the research methods, and procedures, techniques associated with the collection and analysis of gathered evidence (Alharahsheh & Pius, 2020). Research methods are used to produce and develop knowledge. Traditionally, there are two kinds of research method, namely quantitative and qualitative. Qualitative methods are normally associated with normative

paradigms and quantitative methods are normally associated with the positivist/post-positivist paradigms (Makombe, 2017). However, there is a third method which is a mixture of the qualitative and quantitative methods, and this combination research approach can be adopted in some cases and results in high-quality research.

Makombe (2017) agrees and states that there is a spectrum between qualitative and quantitative methods for the mixed methods approach and that there are countless ways of addressing mixed methods research. There are cases where either qualitative or quantitative methods are equally suitable. For example, in a study of rice flavour enhancers, an experimental approach can be used by means of statistical methods, such as random a selection, a control and a treatment group. The control group is provided with rice that does not contain any flavour enhancements and the treatment group is provided with rice that does contain the flavour enhancer. In the event that more than one enhancer is being evaluated, experimental designs, such as a completely randomised selection can be applied successfully, but due to the intricacy of measuring taste, a Likert scale can be used to evaluate the taste. As a result, this makes the study quantitative, but the taste can also be assessed qualitatively.

This research study used the mixed methods research method which is discussed further in the following sections. Quantitative and qualitative data were collected and provided different types of information. In the case of the qualitative data, open-ended data was collected and in the case of quantitative, closed-ended data was collected for this study. A comparison was drawn between the different perspectives of individuals drawn from both types of data, by creating a greater understanding of the necessary changes that are required for a marginalised group by integrating qualitative and quantitative data (Creswell & Creswell, 2017).

4.3.2 RESEARCH DESIGN

Traditionally, qualitative and quantitative research methodologies are frequently contrasted and presented as two opposing entities (Granikov *et al.*, 2020). The integration of qualitative and quantitative approaches can be traced back to the 1950s and its inception in Psychology (Creswell & Creswell, 2017; Creswell & Plano Clark, 2018; Pluye, 2012). In general terms, in quantitative research approaches, researchers are concerned with the collection and measurement of numerical data, and the analysis thereof, using statistical methods, whereas qualitative approaches are concerned with the collection of textual data, and the description or

interpretation thereof using inductive thematic approaches (Bazeley, 2018; Clark & Ivankova, 2015). In mixed methods research, the researcher utilises both quantitative and qualitative approaches for data collection and analysis, integration of the findings, and drawing inferences (Creswell & Tashakkori, 2007; Johnson *et al.*, 2007).

Many different terms refer to the mixed-methods approach, namely integration, synthesis, multimethod, quantitative and qualitative methodology, and mixed methodology”, but in recent times it is more commonly known by the term “mixed methods” (Bryman, 2006). Creswell and Creswell (2017) state that in the last decade with the growing development and recognised legitimacy of both qualitative and quantitative research, mixed methods research using a combination of qualitative and quantitative research approaches has gained renewed interest as well as popularity. It is a research methodology that is continuously developing and evolving due to its prominence by utilising the strengths of quantitative and qualitative research. Many researchers are interested in this combination in order to increase the scope of understanding a phenomenon and to deepen their insights from their studies (Bryman, 2006; Creswell *et al.*, 2011; Sandelowski, 2000). The complexity of human phenomena mandates more complex research designs to capture them as advocated by mixed-methods researchers because the use of either quantitative or qualitative approaches exclusively is inadequate to address this complexity (Creswell & Creswell, 2017; Hong *et al.*, 2020; Sandelowski, 2000).

More researchers are exposed to this multimethod of research in recent years, and since the start of the 21st century, this research methodology has grown substantially with the creation of international associations such as the Mixed Methods International Research Association (MMIRA), Methodes Mixtes Francophonie (MMF), and the JSMMR (Japan Society for Mixed Methods Research), regional and global conferences, graduate courses and workshops development, book publications plus specialised peer-reviewed academic journals such as the Journal of Mixed Methods Research (JMMR) (Hong *et al.*, 2020).

Conducting mixed-methods research is challenging, since it requires the combination of both qualitative and quantitative research which demands extensive data collection. A great amount of time is needed for the analysis of both textual and numerical data, and it requires the researcher to be familiar with both qualitative and quantitative research (Creswell & Creswell, 2017; McCrudden & Marchand, 2020). The integration challenge associated with this combination of

research methodologies is due to “producing a whole that is greater than the sum of each individual part” (Fetters & Freshwater, 2015).

Furthermore, Fetters and Freshwater (2015) describe that this integration of qualitative and quantitative methods can be achieved in three different ways in a mixed-methods study: (1) research design (basic designs, and advanced frameworks), (2) data collection methods and analysis (connecting, building, merging and embedding), and (3) the interpretation and reporting of the results (narrative, data transformation, joint display). As attested by Creswell (2013), research design can be classified into six types: (1) sequential explanatory design, (2) sequential exploratory design, (3) sequential transformative design, (4) concurrent triangulation design, (5) concurrent nested design, and (6) concurrent transformative design. It is of the utmost importance for a researcher to consider that the mixed-methods research approach has its own challenges, mainly in relation to the incompatibility of both qualitative and quantitative research approaches. Each of these research designs has its own advantages and disadvantages linked to the research questions and the aim of the research study (Conrad *et al.*, 2014; Creamer, 2018; Ramírez-Montoya & Lugo-Ocando, 2020).

A mixed-methods research design was used for this research study using an explanatory sequential design (QUAN → qual) (see Figure 4.1). This study used the convergent (concurrent) design to address the posed research question(s), as the quantitative and qualitative data were collected and analysed separately, with the former occurring prior to the latter. The quantitative investigation of the research study used a simple research design. Similarly, the qualitative investigation process used a basic research design with the aim of assisting the researcher to gain a better understanding of the perceptions of minority students in Information Technology experiencing programming and pair programming (Liebenberg *et al.*, 2012).

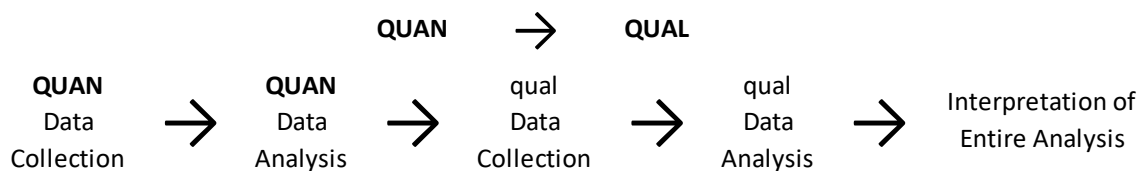


Figure 4.1 Sequential Explanatory Design

The qualitative investigation was based on textual narratives rather than numbers. Previous studies linked to a similar research problem investigated in my study, applied the quantitative approach. The main reason for this research study (underrepresentation of minorities in IT) led to a quantitative and qualitative investigation with pair programming as an intervention, as the researcher wished to describe the meaning of lived experiences and perceptions of minorities in IT in-depth (Bailey & Mentz, 2017; Liebenberg *et al.*, 2012). In agreement with the aim of the research study, the mixed methods approach used in this study was adequate in the sense that the objective is to not only better understand the particular phenomenon, but also to know what the reality (truth) is of these minority students and their views of pair programming (see Table 4.1 below).

Table 4.1 – An Adapted Sequential Explanatory Design used for this Research Study

Stages	Procedures	Outcomes
Quantitative Data Collection	Collection of data	Demographics of participants
↓	Measurement of numerical data	Numerical data
Quantitative Data Analysis	SPSS Data Analysis	Descriptive Statistics
↓	Frequencies	Examination of relationships among variables
	Correlations	Associations between and within group differences
	Factor Analysis	
Qualitative Data Collection	Develop structured, self-administered questionnaire	Close-ended responses
↓	Distribute questionnaire to the 284 participants	Open-ended responses
		Summarize responses
Qualitative Data Analysis	ATLAS.ti	Conceptual model of emergent themes
↓	Coding and thematic analysis	Codes and themes
	Theme development	Coding matrix
Integration of Quantitative and Qualitative Findings	Integration and explanation of the quantitative and qualitative findings	Discussion
		Implications
		Future research

4.3.3 PARTICIPANTS

According to Rahi (2017), a population can be defined as “all people or items that a researcher wishes to understand while sampling is the process of selecting a segment of the population for investigation”. Simply put, it is the process of selecting a sample of units for a single data set so that one may measure the beliefs, characteristics, and attitudes of the people involved (Hair Jr et al., 2019). The participants in this research study were (n=284) first-year BSc IT students in the Graphical User Interface Programming (second-semester class) at the Potchefstroom campus of the North-West University. The programming class had students from different genders, ethnic groups, cultures, and socio-economic backgrounds. The researcher grouped the participants in such a way that the students who identified as either Black, Coloured, Indian and/or Asian belong to the minority and those who identified as White belong to the majority for the context of this study. The participants were students who had experience with IT subjects in high school and students who were completely new to programming with no prior programming experience. (Table 4.2 provides a summary of the biographic data of the students.)

Table 4.2 – Profile of Students (n=284)

Criteria	Categories	Number of students (%)
Gender	Male	200 (70.4%)
	Female	84 (29.6%)
Ethnic origin	Majority	196 (69.0%)
	Minority	88 (31.0%)
Computer access since Grade 1	Yes	150 (52.8%)
	No	134 (47.2%)
Internet access since Grade 1	Yes	190 (66.9%)
	No	94 (33.1%)
IT as a subject in high school	Yes	203 (71.5%)
	No	81 (28.5%)

In terms of ethnic origin, only 31% of the students identified themselves as minorities who in reality belong to the majority population of South Africa. Only 29.6% of the students were women which is to be expected from the typical gender profile of most IT classes which indicates that they are a minority within a minority.

There is a digital divide amongst the students as a whole with 33.1% of the students, being the minorities, who do not have access to the internet at home. Furthermore, 28.5% of the students are in the minority with regard to having prior knowledge of the subject in high school. At the time of the study, the students had a female educator and the language of instruction was Afrikaans and/or English. This group of first-year students was selected because they had gained the experience of solo-programming in previous modules in their first semester of their first year. The students were taught the principles of pair programming by the researcher and were given the option to partner with a friend of their choosing. The pairs programmed together by using one computer to complete their practical assignments every week throughout the course of the semester. Section 4.3.4 contains a discussion of the pair programming principles that the students were taught.

4.3.4 PAIR PROGRAMMING TRAINING

Students are not familiar with the principles of pair programming, since it is a new way of active learning. It is important to have a consistent pair programming environment, hence several guidelines must be used as a framework by educators interested in adopting pair programming (Bevan *et al.*, 2002; Zarb *et al.*, 2014). To carry out pair programming efficiently, educators are required to introduce the main aspects of pair programming, such as pair compatibility, as well as the factors that impact the pairing, such as communication and teamwork to the classroom or computer laboratory (Dou *et al.*, 2017). (See section 3.3.5 for the pair programming guidelines).

At the time of the study, the pairs were formed by the students through them choosing a partner who most likely was a friend. This was done to avoid incompatibility and conflict. The researcher administered pair programming during the students' practical sessions each week. The students were constantly reminded to work together closely, to communicate with each other as a team and to clearly describe the requirements of the practical assignment. Students were prompted to exchange roles at regular intervals throughout the duration of the practical session. In the first pair programming practical session, the students were given the following guidelines:

1. The driver operates the keyboard and mouse.

2. The navigator follows what the driver is executing on screen and looks out for errors and proposes corrections. The navigator is in charge of programming language reference materials.
3. Positions: The driver is in front of the keyboard with the mouse in hand. The navigator sits on the opposite side of the mouse. The screen is angled so the navigator can clearly view it. The navigator positions his/her chair so that he/she can point at the screen. The navigator has reference materials organised on the available table space to ensure easy access.
4. The navigator ensures that he/she clearly understands the aims of the program.
5. Partners physically get up and move positions when switching roles.
6. Partners pay close attention to each other when pair programming: they look at each other when talking and listen carefully, they ask for clarification, and confirm understanding.
7. Partners work hard to ensure each person understands what is being created. The driver must check for agreement on operations before execution. The partners point at the screen to support clear communication and the driver, while he/she is working, describes what he/she is doing. The navigator shows notes recorded in the design notebook to the driver to check for agreement.
8. Partners are respectful of each other: navigators do not handle the mouse or keyboard; drivers do not grab for reference materials; disagreement is natural and should be resolved respectfully.
9. Partners share ownership of the project.
10. Partners help each other, create opportunities for each other to learn, promote trading off of pair-programming roles, and share the creation of their project.
11. Partners must assess each other and must be honest in their evaluation.

12. Partners take time to reflect on how well they are functioning and they must plan to improve.
13. Partners use the words “we” and “us” to talk about their decisions and work.
14. Partners must trust each other's judgment and each other's loyalty to the team.
15. Problems with a partner must be surfaced immediately to give the facilitator an opportunity to correct the situation.

The students followed the aforementioned pair programming guidelines diligently at each pair programming practical session under observation of the researcher throughout the course of the semester.

4.3.5 DATA COLLECTION METHODS AND INSTRUMENTS

4.3.5.1 INSTRUMENT

Instruments can be used in combination studies to fulfil a variety of purposes. Instruments can be used as an elicitation device in the questionnaire concerning both the target phenomena and the instrument itself. In this case, the target phenomena are the underrepresented students in the programming course and the instrument is the questionnaire. For example, the researcher used the participant's responses from the open-ended questions to draw out more thoughts and feelings about pair programming.

According to Sandelowski (2000) these instruments are used to “make generalizations about each participant involved in the study not to make nomothetic generalizations, nor generalizations from the study sample to populations”. This use of instruments in combination exemplifies the development purpose for mixed-methods research, in the sense that the results of using quantitative data collection inform or guide the use of the qualitative data collection technique (Sandelowski, 2000). The questionnaire developed posed questions based on the UTAUT constructs using a five-point Likert scale. The discussion of the UTAUT model will follow in Section 4.3.5.2.

4.3.5.2 UTAUT

There are various cultural and psychological factors that can affect the behaviour of users towards the use and acceptance of pair programming. One of the methods to measure the factors that result in the acceptance of pair programming is the UTAUT model. At present, the advancement of ICT at universities has changed the traditional education methods into technology-based methods due to a generation of young people who grew up with the Internet and digital technologies (Ameri *et al.*, 2020; Liebenberg *et al.*, 2018). Venkatesh *et al.* (2003) designed the UTAUT model by incorporating the eight IT acceptance models which, as a result, made it one of the most comprehensive models of technology acceptance. The UTAUT model by Venkatesh *et al.* (2003) had four primary constructs namely; performance expectancy (PE), effort expectancy (EE), social influence (SI), and facilitating conditions (FC), which have an influence on the behavioural intention (BI) and usage behaviours (UB) of a user (Nikolopoulou *et al.*, 2020; Raman & Don, 2013; Venkatesh *et al.*, 2012).

Venkatesh *et al.* (2003) described these relevant constructs under the UTAUT model as follows:

- Performance Expectancy: can be seen as the degree to which an individual believes that using the system will help him/her to attain gains in job performance.
- Effort Expectancy: this construct represents the individual's beliefs regarding the ease or effort associated with the use of the system.
- Social Influence: is the extent to which an individual perceives that which is important to others such as friends and family, believe they should use a particular new technology.
- Facilitating Conditions: can be seen as the extent to which an individual believes that an organizational and technical infrastructure exists to support the use of the system.
- Behavioural Intention: this construct refers to the motivational factors that influence a given behaviour where the stronger the intention to perform the behaviour, the more likely the behaviour will be performed by an individual.
- Use behaviour: refers to the extent to which an individual displays continuous commitment to a system.

Moreover, individual differences like gender, age and experience are considered as the moderators of the four original UTAUT constructs in the UTAUT model (see Figure 4.2 for more detailed relations) (Raman & Don, 2013; Sharif *et al.*, 2019; Venkatesh *et al.*, 2012).

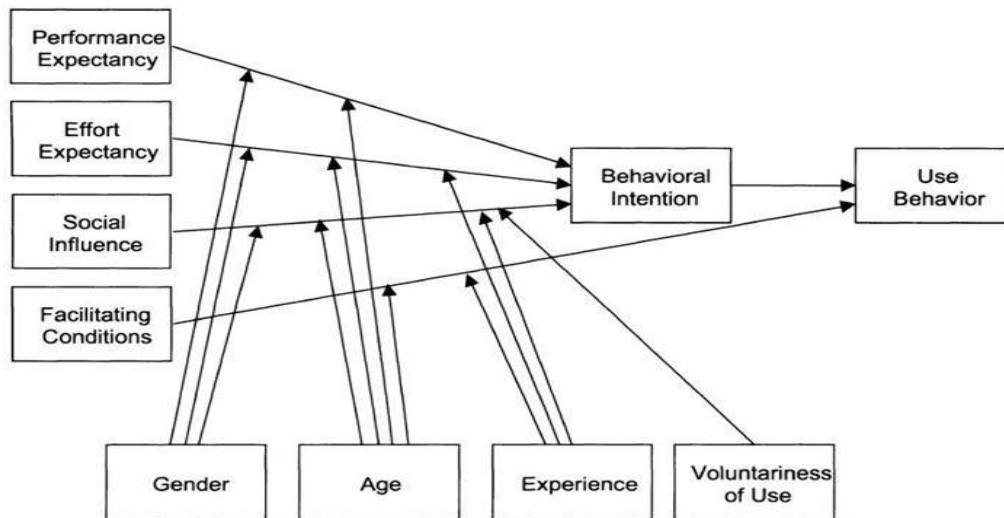


Figure 4.2 The Original UTAUT Model (Venkatesh *et al.*, 2003)

Venkatesh *et al.* (2012) proposed the UTAUT2 model by means of extending the original UTAUT model with three additional consumer constructs namely; hedonic motivation (HM), price value (PV), and habit (HT) with the objective to enhance its explanatory power and scope up the restrictions in the original model (Ameri *et al.*, 2020; El-Masri & Tarhini, 2017; Raman & Don, 2013; Sharif *et al.*, 2019).

The UTAUT2 model incorporated three other variables (see Figure 4.3), Venkatesh *et al.* (2012) explained the additional constructs as follows:

- Hedonic Motivation: can be as seen as the fun, or pleasure derived from using the new technology.
- Price Value: the cost and the pricing structure may significantly impact the individual's use of the technology, since it is they who bear the monetary cost of such use.
- Habit: the extent to which an individual performs the behaviour to be automatically as a result of learning.

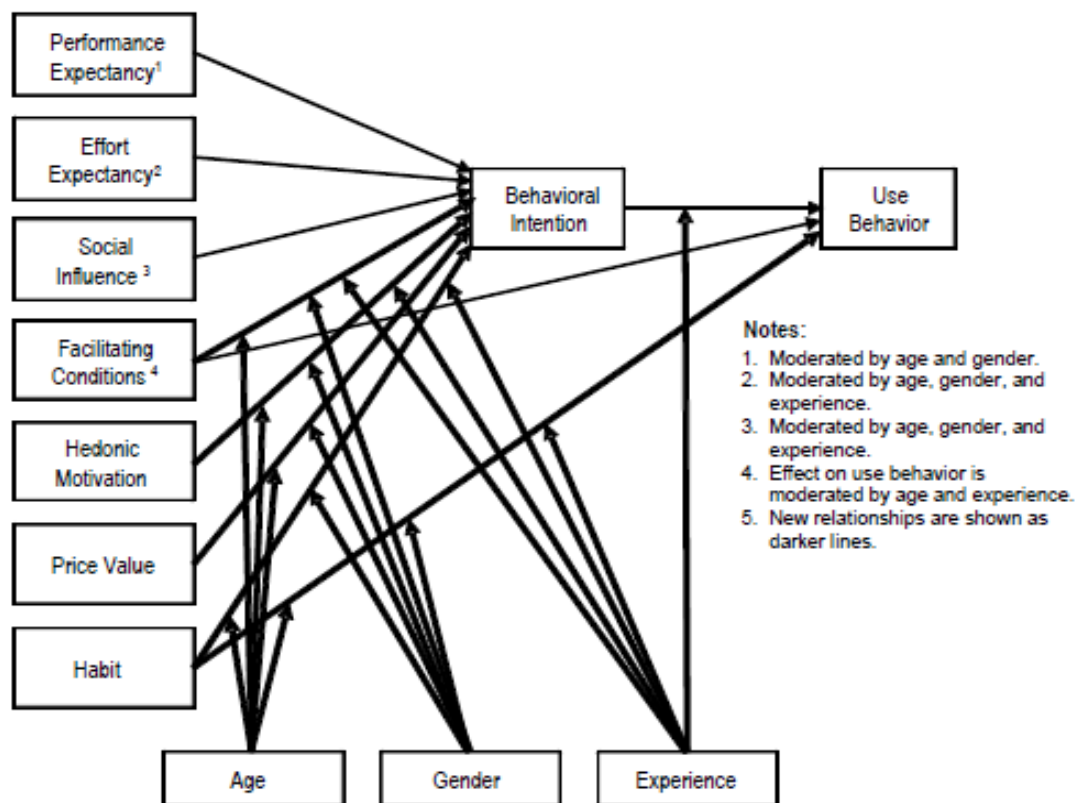


Figure 4.3 The UTAUT2 Model with Moderators and Additional Constructs (Venkatesh *et al.*, 2012)

4.3.5.3 DATA COLLECTION METHOD

Surveys as a data collection strategy, are commonly used in social and health sciences and it is associated with deductive research approaches (Creswell & Hirose, 2019). In this research study, a survey in the form of a questionnaire was used to collect information by means of a pre-designed questionnaire. There are three conditions for when conducting research using the survey strategy, (1) Data of the research is based on quantitative methods, (2) Pre-defined instruments are used in the research, (3) research work that needs to analyse a finding of a sample to a whole population as stated by Pinsonneault and Kraemer (1993). Surveys vary in degrees of complexity due to surveys gathering data at a particular point in time with the intention to examine individuals, groups, institutions, methods in order to describe the nature of existing conditions, or identifying standards against which pre-existing conditions can be compared, contrasted, classified, analysed and interpreted (Cohen *et al.*, 2013).

The survey strategy has several characteristics; this strategy was chosen to:

- collect standardised data all in one go,
- gather large scale data from a wide target population in order to enable generalisations about the target in focus,
- generate numerical data that can be processed statistically,
- provides descriptive, inferential and explanatory information, and assert correlations,
- capture data from multiple choices and/or closed questions,
- support or refute hypotheses about the target population (Cohen *et al.*, 2013).

As reported by Nemoto and Beglar (2014), educational research in this context has three primary objectives: (1) contribute to the development of theory, (2) investigate phenomena believed to play an important role in the educational process, and (3) develop a more effective pedagogy. In order to complete these objectives, the use of data-gathering tools such as a questionnaire form is required. Due to the intricacy that comes with the measurement of the students' attitude towards pair programming, a rating scale was in order to quantify subjective preferential thinking, feeling and action in a validated and reliable way. In 1932, the Likert Scale was constructed to measure one's 'attitude' in a manner which was scientifically accepted and validated (Joshi *et al.*, 2015; Krosnick, 2018). A Likert scale is comprised of a set of statements (items) offered to a participant for a real or hypothetical situation under study. The students were asked to represent their degree of agreement (ranging from strongly disagree to strongly agree) (see Table 4.3 below) with the provided statement on a metric scale (Joshi *et al.*, 2015; Nemoto & Beglar, 2014). All of the given statements were used in combination so that a researcher may reveal the specific dimension of the attitude towards the issue (Singh, 2006). For this reason, the Likert scale is regarded as one of the most fundamental and generally used psychometric tools in educational and social sciences research (Joshi *et al.*, 2015).

Table 4.3 – A Five-Point Likert Scale

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	2	3	4	5

The questionnaire designed for this research study used the UTAUT2 model (Venkatesh *et al.*, 2012) and is an adaptation of the questionnaire from the Benadé and Liebenberg (2017) study (see Table 4.4 below). The questionnaire was divided into three distinct parts. Part A, the first part, was designed to collect the demographic and general information of the students (gender, ethnicity, prior experience with the computers, the internet and IT as a high school subject). Part B, the second part, comprised of statements that asked the students questions based on the following constructs: performance expectancy, hedonic motivation, effort expectancy, social influence, habits, facilitating conditions, and behavioural intention. The third part, Part C, consisted of statements that asked the students about their challenges and difficulties experienced throughout the semester implementing pair programming to complete their practical assignments and concluded with two open-ended questions. Due to the educational nature of this research study, the “Price Value (PV)” and “Use Behaviour (UB)” constructs were inapplicable to the study and were excluded as a result.

Out of the 23 questions in Part B, 4 statements for performance expectancy, 3 statements for hedonic motivation, 4 statements for effort expectancy, 3 statements for social influence, 3 statements for habits, 3 statements for facilitating conditions, and 3 statements for behavioural intention, were presented respectively. The five-point Likert scale (quantitative with a range 1 = strongly disagree to 5 = strongly agree) was used to answer the questions in parts B and C.

Immediately after the final pair programming practical session, the students were given a week to complete the questionnaire form online, using Google Forms. The questionnaire consisted of 38 questions and took approximately 15 minutes to complete.

Table 4.4 – Questionnaire

Factors		Questions	Answer Type	
Demographics	1	What is your gender?	Multiple Choice	Part A
	2	Please specify your ethnic origin:	Multiple Choice	
	3	I have a computer (laptop/PC) at home.	Yes/No	
	4	I had access to a computer since Grade 1.	Yes/No	
	5	I had access to the Internet since Grade 1.	Yes/No	
	6	In high school I had IT as a subject.	Yes/No	
	7	In your programming module(s) overall, please provide a grade for your academic performance.	Multiple Choice	
Performance Expectancy	8	I find pair programming useful.	Likert Scale	Part B
	9	The use of pair programming allows me to increase my chances of improving my marks.	Likert Scale	
	10	Using pair programming helps me to complete tasks more quickly.	Likert Scale	
	11	Using pair programming increases my productivity.	Likert Scale	
Hedonic Motivation	12	Using pair programming is fun.	Likert Scale	
	13	Using pair programming is enjoyable.	Likert Scale	
	14	Using pair programming is entertaining.	Likert Scale	
Effort Expectancy	15	Learning to use pair programming is easy for me.	Likert Scale	
	16	My interaction using pair programming is clear and understandable.	Likert Scale	
	17	I find pair programming easy to use.	Likert Scale	
	18	It is easy for me to become a skillful pair programmer.	Likert Scale	
Social Influence	19	People who are important to me think that I should use pair programming.	Likert Scale	
	20	My lecturer thinks that I should use pair programming.	Likert Scale	
	21	My fellow students whose opinions I value think I should use pair programming to complete tasks.	Likert Scale	
Habit	22	The use of pair programming has become a habit for me.	Likert Scale	
	23	The use of pair programming has become natural to me.	Likert Scale	
	24	I rely on pair programming to complete programming tasks.	Likert Scale	
Facilitating Condition	25	I have the resources necessary to use pair programming.	Likert Scale	
	26	I have the knowledge necessary to use pair programming.	Likert Scale	
	27	Pair programming is compatible with the way I write programs.	Likert Scale	
Behavioral Intention	28	I intend to use the principles of pair programming in the future.	Likert Scale	
	29	I will always try to use the principles of pair programming to complete programming tasks.	Likert Scale	
	30	I plan to continue using the principles of pair programming elsewhere in my studies.	Likert Scale	
Pair Compatibility	31	My programming partner and I are incompatible.	Likert Scale	
	32	It is difficult to use pair programming if my partner is on a different programming skill level as I am.	Likert Scale	
	33	The lack of communication between my partner and I makes using pair programming difficult.	Likert Scale	
	34	Language was a barrier when we were programming in pairs.	Likert Scale	
Loose Items	35	Pair programming taught me teamwork.	Likert Scale	
	36	I prefer pair programming over solo programming.	Likert Scale	
Open-Ended	37	What are the advantages of pair programming?	Open-ended	Part C
	38	What are the disadvantages of pair programming?	Open-ended	

4.3.5.4 FACTOR ANALYSIS

Kline (2014) defines factor analysis as a number of statistical techniques that aim to simplify large complex data sets. It is more often used in psychology, but in recent years it has been used in education, social sciences, economics, geography, and in fields that relate to health sciences (Kline, 2014; Williams *et al.*, 2010; Yong & Pearce, 2013). The aim of factor analysis is to discover the simplest method of interpretation of variables (or factors) in an observed data set (Yong & Pearce, 2013). A factor is a construct which is a compressed statement of the relationships that exist between a set of variables. It is defined by its factor loadings which are the correlations of a variable with a factor (Kline, 2014). In factor analysis there are two techniques, namely Confirmatory Factor Analysis (CFA) and Exploratory Factor Analysis (EFA). In CFA, the researcher has assumptions and expectations and tests a proposed theory based on priori theory regarding the number of variables in order to determine which factor theories are the best fit (Williams *et al.*, 2010). In contrast, EFA just as the name suggests is exploratory in nature meaning that the researcher uses this approach in order to discover the main factors influencing the variables in order to generate a theory or model from a set of latent variables in order to analyse which of them account for the correlations (Yong & Pearce, 2013).

For this study, the researcher used the CFA approach to reduce a large number of variables to a smaller number of factors while retaining as much variance as possible. In order to display the relationships between the individual variables, the researcher used a correlation matrix in the CFA process. Upon inspection of the correlation matrix, the researcher found that the determinant = 2.663E-9. The correlation matrix tested if the variables share too much variance. The determinant was found to be >0.00001. This indicated that there was no severe multicollinearity which made it practical to determine the correlations between the variables. Before the extraction process of the factors was performed, the researcher used the Kaiser-Meyer-Olkin (KMO) Measure of Sampling Adequacy and Bartlett's Test of Sphericity in order to assess the appropriateness of the respondent data for factor analysis (Banu & Duraipandian, 2014; Yong & Pearce, 2013). The KMO index ranges from zero to one, with < 0.5: not acceptable; 0.5 –0.7: medium, 0.7 –0.8: good; 0.8 –0.9: very good; and >0.9: superb. The Bartlett's Test of Sphericity should be significant ($p < 0.5$) for the factor analysis to be appropriate. Table 4.5 below shows the SPSS output for the KMO and Bartlett's Test of Sphericity for this study.

Table 4.5 – Kaiser-Meyer-Olkin Measure and Bartlett's Test of Sphericity

Kaiser-Meyer-Olkin	0.940	
Bartlett's Test of Sphericity	Approx. Chi-Square	5419.717
	df	253
	Sig.	0.000

The reported KMO value of 0.940 showed that the test for sample adequacy was a superb sample adequacy, which indicated that the sample size was indeed suitable to perform a factor analysis on the data. With regard to the Bartlett's Test of Sphericity, the 0.000 value indicated that the correlations between the factors were indeed sufficiently high.

The results from the Total Variance Explained table (Table 4.4 below) was used to determine the number of significant factors. In this table only the extracted and rotated values are deemed meaningful for interpretation (Yong & Pearce, 2013). The Initial Eigenvalues and the Extraction Sums of Squared Loadings columns show the eigenvalues and the variance before the rotation takes place. The Rotation Sums of Squared Loadings show the eigenvalues and the variance after the rotation takes place. The Cumulative percentage of variance of at least 50% is demonstrated in Table 4.6 below with a total of 7 factors having an initial eigenvalue > 0.7.

Table 4.6 – Total Variance Explained

Factors	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings ^a
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total
Performance Expectancy	11.963	52.014	52.014	11.717	50.946	50.946	2.798
Hedonic Motivation	2.031	8.830	60.844	1.743	7.578	58.524	3.717
Effort Expectancy	1.302	5.660	66.504	0.997	4.335	62.859	5.908
Social Influence	1.006	4.376	70.880	0.744	3.233	66.093	8.869
Habit	0.864	3.757	74.636	0.624	2.713	68.807	6.600
Facilitating Condition	0.764	3.320	77.957	0.501	2.182	70.989	8.381
Behavioral Intention	0.726	3.156	81.113	0.339	1.472	72.461	7.866

The researcher calculated the average extracted communalities in order to determine the eigenvalue cut-off based on The Kaiser criterion (Field, 2009; Yong & Pearce, 2013). The SPSS output for Communalities (as seen in Table 4.7 below) shows that the averaged extracted

communalities is above .60 and the sample size for this study being 284 is above 250 cases (Field, 2009).

Table 4.7 – Communalities

Questions	Initial	Extraction
Q8	0.795	0.796
Q9	0.773	0.784
Q10	0.714	0.764
Q11	0.790	0.830
Q12	0.815	0.850
Q13	0.848	0.915
Q14	0.736	0.763
Q15	0.702	0.762
Q16	0.610	0.611
Q17	0.749	0.797
Q18	0.617	0.622
Q19	0.637	0.934
Q20	0.254	0.205
Q21	0.682	0.694
Q22	0.723	0.816
Q23	0.651	0.657
Q24	0.659	0.705
Q25	0.524	0.622
Q26	0.559	0.740
Q27	0.576	0.573
Q28	0.708	0.748
Q29	0.670	0.729
Q30	0.673	0.748

Since this study used CFA for factor analysis, the extraction method applied was the Principal Axis Factoring method which was used due to its factorial suitability to best extract factors from the data set. According to Williams *et al.* (2010) the selection of the rotation method maximises high item loadings and it minimises low item loads which, as a result, produces a solution that is simplified and more interpretable. In this study the researcher selected the direct oblique oblimin rotation method which produces factors that are correlated. This method produced accurate results for this research due to the fact that it involves human behaviour. The results in Table 4.8 reported that the majority of the questions did indeed correspond with the theoretical UTAUT2 factors. However, both questions Q8: “I find pair programming useful” and Q23: “The use of pair programming has become natural to me” had double loadings.

The researcher decided that the best fit would be to group Q8 with the Performance Expectancy factor and Q23 with the Habit factor as indicated by the UTAUT2 model literature. It is important to note that only questions Q20: “My lecturer thinks that I should use pair programming” and Q27: “Pair programming is compatible with the way I write programs” did not correspond with the theoretical factors, but the researcher decided to group them accordingly anyway.

Table 4.8 – Pattern Matrix

	Pattern Matrix						
	Factors						
Questions	Performance Expectancy	Hedonic Motivation	Effort Expectancy	Social Influence	Habit	Facilitating Conditions	Behavioural Intention
Q8	0.325			0.346			
Q9	0.451						
Q10	0.523						
Q11	0.468						
Q12				0.938			
Q13				0.964			
Q14				0.794			
Q15					-0.792		
Q16					-0.619		
Q17					-0.68		
Q18					-0.58		
Q19			0.979				
Q20	-0.242						
Q21			0.507				
Q22							0.812
Q23					-0.324		0.534
Q24							0.734
Q25		0.768					
Q26		0.832					
Q27						-0.484	
Q28						-0.736	
Q29						-0.733	
Q30						-0.812	

The reliability of the factors in Part B and C of the questionnaire were assessed by the calculation of the Cronbach alpha value of each UTAUT2 construct. Measuring the reliability of a questionnaire provides an indication of how closely related a set of items are as a group. The calculated Cronbach alpha coefficient for the items are $\alpha \geq 0.60$, which suggests that the items have a relatively high internal consistency (as shown in Table 4.9). This indicates that the multiple Likert-Scale questions in the survey are reliable for all eight factors.

Table 4.9 – The Reliability Coefficients of the Factors

Factors	Cronbach Alpha (α)	Adequacy
Performance Expectancy	0.929	Superb
Hedonic Motivation	0.936	Superb
Effort Expectancy	0.885	Very Good
Social Influence	0.716	Good
Habit	0.861	Very Good
Facilitating Condition	0.661	Medium
Behavioral Intention	0.885	Very Good
Pair Compatibility	0.653	Medium

4.3.6 DATA ANALYSIS

The Statistical Consultation Services of the North-West University (NWU) Potchefstroom Campus analysed the data. The analysis of the data was done in SPSS Statistics. First, CFA was performed and then the reliability coefficients were obtained by calculating the Cronbach's alpha (α). The relationships of the factors with behavioural intention were determined by means of a correlation analysis. Frequencies were calculated and basic analysis was performed by calculating the mean values and the standard deviation for each of the factors and the single items. T-tests were used to test for the significant differences between the means in the different UTAUT2 factors for all of the groupings. The p-values will be presented in the results and discussion chapter, but will not be interpreted due to the use of Analysis of variance (ANOVA) for all of the groupings in order to obtain more information about the relationships between dependent and independent factors.

Part C of the questionnaire contained two open-ended questions, Q37 and Q38 in Table 4.2. The qualitative data was analysed, using ATLAS.ti 9. Content analysis was performed on each participant's data from the two open-ended questions, and these responses were coded in order to categorise them into themes. The qualitative analysis provided richly descriptive results which aided in gaining better insight and understanding of the perceptions, experiences, underlying reasons and motivations of minorities that used pair programming. The significant results from the students are presented in quotes in the next chapter where they are discussed in-depth.

4.4 SUMMARY

In Chapter 4, the data collection instrument was developed in order to evaluate the factors that influence the perceptions of pair programming of first year students. The research paradigm, design and research methodology of the quantitative and qualitative investigations were reported in this chapter. The study used the mixed-method approach and this chapter also discussed the data collection, and the factor analysis that was performed on the respondent data in order to address the objectives that were set. In Chapter 5, the results of the analysed data are presented, followed by a discussion and a graphical presentation thereof.

CHAPTER 5 RESULTS AND DISCUSSION

5.1 INTRODUCTION

In Chapter 5, the researcher reports the results from the questionnaire, Part B (quantitative) and Part C (qualitative open-ended questions) that were discussed in the previous chapter. The results that are presented in this chapter address the questionnaire results by means of a discussion on the descriptive and inferential statistics, frequency distribution analysis on the demographic data from the students and the relationships between the demographic variables.

5.2 DESCRIPTIVE STATISTICS

Descriptive statistics are used in this study to present the quantitative descriptions and a simplification of the collection of responses in Table 5.1. This was done to describe the characteristics of these responses and this simplification allows the researcher to analyse the responses in order to understand them in a sensible manner. Survey research was conducted in order to collect data by means of Likert scale type questions to easily operationalise the perceptions of the students of this study. This questionnaire used a five-point Likert scale so that the data can be analysed quantitatively which allows the researcher to gain deeper insights about the students of this study. The students chose a level of agreement or disagreement based on the questions from the questionnaire in order to see if they are pleased or have issues with pair programming in particular.

- *Performance Expectancy* - it is evident that 33.8% of the students agreed that pair programming is useful and 37.3% of them also agreed that pair programming allows them to improve their marks. 30.6% of the students agreed that using pair programming helps them to complete their programming tasks quickly and 28.9% of the students agreed that pair programming increases their productivity. There is a mostly positive feeling with regards to the degree to which the students believe that using pair programming helps them to increase their chances of improving their marks.
- *Hedonic Motivation* - 33.8 percent of the students agreed that pair programming is fun and 38% of the students agreed that pair programming is enjoyable. The researcher found that 30.6% of the students agree that pair programming as a learning technique is entertaining.

It can be deduced that most of the students do derive fun from using pair programming and this plays a major role in their acceptance of using this learning technique.

- *Effort Expectancy* - 37.7 percent of the students agreed that learning how to use pair programming was an easy task that required minimal effort. The researcher found that 45.4% of the students found the guidelines and principles of pair programming to be clear and understandable. With regards to ease of use, 36.3% of the students found pair programming easy and 39.8% of them agreed that it is easier for them to become a skilful programmer. It is clear that almost half of the students felt positive about the degree of ease associated with the use of pair programming and, as a result this may have an impact on the overall use intention and perceptions of programming.
- *Social Influence* - the researcher found that 41.5% of the students were neutral and felt indifferent about the people that think that they should make use of pair programming. 37.7% of the students felt neutral about their lecturer recommending the use of pair programming and 36.6% of them agreed that their lecturer thinks that they should make use of pair programming. The researcher found that 33.8% of the students felt indifferent when asked about what their fellow students thought on the use of pair programming in order to complete programming tasks. It is evident that the students felt indifferent about the degree to which they perceive that significant people believe pair programming use to be important.
- *Habit* - the researcher found that 27.5% of the students disagreed that the use of pair programming has become a habit for them. In contrast, 34.5% of the students agreed and felt that pair programming was a habit that came naturally to them. 29.2% of the students did not rely on pair programming to complete programming tasks which indicated to the researcher that this group may prefer solo programming over pair programming. It is clear that the students overall felt neutral about the extent to which they perform pair programming automatically, but they also believed that pair programming as a behaviour can be automatic the more they use it. With more experience using pair programming it can be deduced that they will eventually rely on it in the future.

Table 5.1 – Questions, Factors and Measure of Frequency (n=284)

	Questions	Factors	Strongly Disagree (%)	Disagree (%)	Neutral (%)	Agree (%)	Strongly Agree (%)
8	I find pair programming useful.	Performance Expectancy	7.0	8.5	22.9	33.8	27.8
9	The use of pair programming allows me to increase my chances of improving my marks.		9.2	10.6	18.3	37.3	24.6
10	Using pair programming helps me to complete tasks more quickly.		10.2	14.8	19.4	30.6	25.0
11	Using pair programming increases my productivity.		11.6	14.8	23.9	28.9	20.8
12	Using pair programming is fun.	Hedonic Motivation	7.7	7.0	23.2	33.8	28.2
13	Using pair programming is enjoyable.		7.0	7.7	23.2	38.0	23.9
14	Using pair programming is entertaining.		7.4	12.3	27.1	30.6	22.5
15	Learning to use pair programming is easy for me.	Effort Expectancy	5.3	10.2	21.5	37.7	25.4
16	My interaction using pair programming is clear and understandable.		2.1	3.9	22.2	45.4	26.4
17	I find pair programming easy to use.		5.3	8.8	23.6	36.3	26.1
18	It is easy for me to become a skillful pair programmer.		6.3	9.2	26.4	39.8	18.3
19	People who are important to me think that I should use pair programming.	Social Influence	10.2	16.5	41.5	21.8	9.9
20	My lecturer thinks that I should use pair programming.		3.2	1.8	37.7	36.6	20.8
21	My fellow students whose opinions I value think I should use pair programming to complete tasks.		9.9	16.9	33.8	26.1	13.4
22	The use of pair programming has become a habit for me.	Habit	16.2	27.5	23.2	21.5	11.6
23	The use of pair programming has become natural to me.		11.3	17.3	24.6	34.5	12.3
24	I rely on pair programming to complete programming tasks.		22.2	29.2	20.4	18.3	9.9
25	I have the resources necessary to use pair programming.	Facilitating Condition	2.5	3.5	18.3	51.4	24.3
26	I have the knowledge necessary to use pair programming.		1.4	2.5	18.7	51.8	25.7
27	Pair programming is compatible with the way I write programs.		7.7	12.7	29.6	33.8	16.2
28	I intend to continue using pair programming in the future.	Behavioral Intention	6.3	9.2	24.3	41.9	18.3
29	I will always try to use the principles of pair programming to complete programming tasks.		7.7	15.8	33.1	30.3	13.0
30	I plan to continue to use the principles of pair programming elsewhere in my studies.		8.5	10.9	27.8	37.7	15.1
31	My programming partner and I are incompatible.	Pair Compatibility	21.1	33.1	23.6	13.4	8.8
32	It is difficult to use pair programming if my partner is on a different programming skill level as I am.		5.6	14.1	22.5	33.1	24.6
33	The lack of communication between my partner and I makes using pair programming difficult.		16.2	25.0	21.5	25.0	12.3
34	Language was a barrier when we were programming in pairs.		47.9	28.2	12.3	7.7	3.9

- *Facilitating Conditions* - 51.4 percent of the students agreed with regards to that they had the necessary knowledge to use pair programming, and 51.8% of the students agreed that they had the necessary resources to adequately use pair programming. With regards to pair programming being compatible with the way in which they write programs, 29.6% of the students felt neutral and 33.8% of them agreed that it is indeed compatible with their preferred writing style. It is evident that the majority of the

students know that they have the organization and technical infrastructure to support the use of pair programming. This is mainly because their use of pair programming occurred during their practical sessions in the computer laboratory.

- *Behavioural Intention* - it was encouraging to see that 41.9% of the students agreed that they intend to continue using pair programming in the future. However, 33.1% of the students felt indifferent when asked if they always attempt to make use of pair programming principles to complete programming tasks. The researcher also saw that 30.3% of the students agreed that they always try to use pair programming principles to complete programming tasks. 37.7% of the students agreed that they plan on using the pair programming principles elsewhere in their studies.
- *Pair Compatibility* - the researcher noted that 33.1% of the students disagreed with regard to pair incompatibility with their partner which makes sense because they were allowed to pick their partners for pair programming. It is important to note that 33.1% of the students agreed that they indeed experienced difficulties using pair programming due to their partner having a higher or lower programming skill than them. 25% of the students disagreed that the lack of communication between them and their partner made the use of pair programming difficult, and in contrast 25% of the students agreed that the lack of communication made it difficult for them to use pair programming. It is unclear what the cause of this could be, but one can assume that absence/tardiness could possibly be a reason for a lack of communication because the student had to program alone. From this quantitative analysis it can be assumed that overall the students prefer pair programming over solo programming in spite of there being a small group that is opposed to pair programming.

The single items in Figure 5.1 show that the students involved in this study had a positive experience with pair programming.

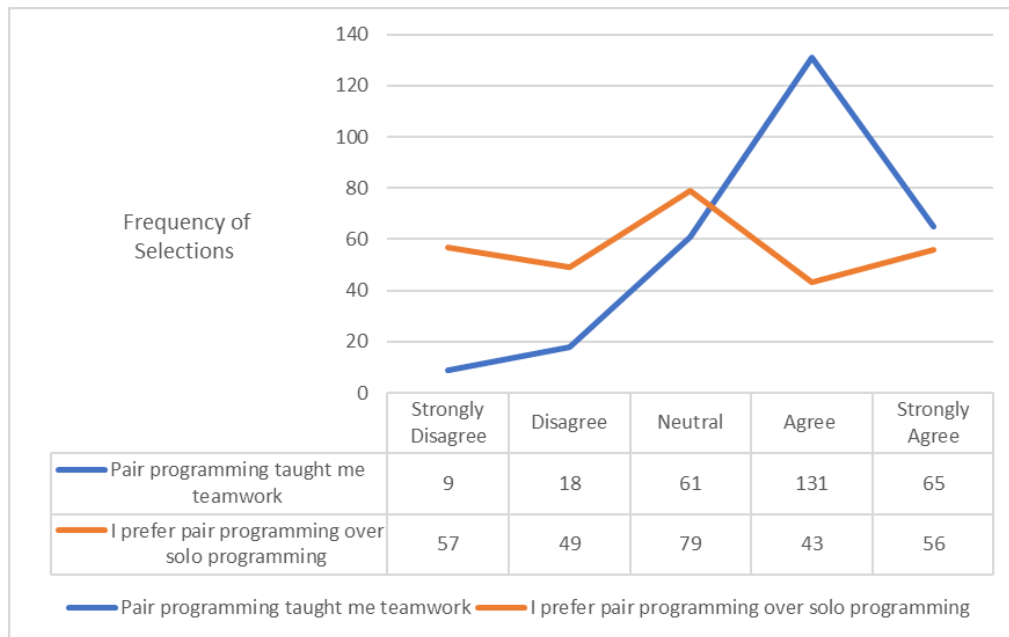


Figure 5.1 Frequencies of Single Items

The statistical results of this study are summarised in Table 5.2. This shows that the mean values of all the items displayed below are relatively high (with a score of 3 being neutral) which simply indicates that the students felt neutral or agreed with the different statements.

- *Performance Expectancy* - has a mean score of 3.51 which indicates that the students agreed that they indeed found pair programming useful and it allows them to not only improve their marks and increase productivity, but also helped them to complete tasks quickly. The majority of the students believe that the usage of pair programming can provide them with benefits when they are completing their programming tasks.
- *Hedonic Motivation* - the mean was 3.60 which is one of the highest mean scores among all of the factors. This shows that the students found pair programming as a fun, and entertaining way to learn how to program. It is clear that many of the students feel motivated to use pair programming due to the internal satisfaction that it brings them.

Table 5.2 – Factors, Questions and Measures of Tendency and Variation (n=284)

Factors	Questions	Mean*	Standard Deviation
Performance Expectancy	I find pair programming useful.	3.51	1.12
	The use of pair programming allows me to increase my chances of improving my marks.		
	Using pair programming helps me to complete tasks more quickly.		
	Using pair programming increases my productivity.		
Hedonic Motivation	Using pair programming is fun.	3.60	1.09
	Using pair programming is enjoyable.		
	Using pair programming is entertaining.		
Effort Expectancy	Learning to use pair programming is easy for me.	3.70	0.91
	My interaction using pair programming is clear and understandable.		
	I find pair programming easy to use.		
	It is easy for me to become a skillful pair programmer.		
Social Influence	People who are important to me think that I should use pair programming.	3.30	0.85
	My lecturer thinks that I should use pair programming.		
	My fellow students whose opinions I value think I should use pair programming to complete tasks.		
Habit	The use of pair programming has become a habit for me.	2.90	1.10
	The use of pair programming has become natural to me.		
	I rely on pair programming to complete programming tasks.		
Facilitating Condition	I have the resources necessary to use pair programming.	3.76	0.74
	I have the knowledge necessary to use pair programming.		
	Pair programming is compatible with the way I write programs.		
Behavioural Intention	I intend to continue using pair programming in the future.	3.41	1.00
	I will always try to use the principles of pair programming to complete programming tasks.		
	I plan to continue to use the principles of pair programming elsewhere in my studies.		
Pair Compatibility	My programming partner and I are incompatible.	2.74	0.84
	It is difficult to use pair programming if my partner is on a different programming skill level as I am.		
	The lack of communication between my partner and I makes using pair programming difficult.		
	Language was a barrier when we were programming in pairs.		
Loose Items	Pair programming taught me teamwork.	3.79	0.97
	I prefer pair programming over solo programming.	2.97	1.39

- *Effort Expectancy* - with the highest mean of 3.70 showed that the majority of the students in the introductory programming course found the pair programming guidelines clear and understandable which, as a result made it easy for them to learn programming and they believe it will make it easy for them to become skilful programmers. It is clear that most of the students found pair programming easy to use. It also makes analysing of the usage of pair programming easy as it is one of the most important determinants, as result the acceptance of pair programming had a positive impact on the students and their perception of programming overall.

- *Social Influence* - with a mean score of 3.30 showed that the lecturer, and the students around them recommend the use of pair programming as a learning technique. The students' decision making process of adopting pair programming is influenced by the social notion beyond their own decision thinking. As a result, this enhances their perception of pair programming because of the social factor that is associated with this determinant.
- *Habit* - with one of the lowest mean 2.90 indicates a neutrality among the students. Some of the students struggled to make it a habit of using pair programming to complete programming tasks. Learning to use pair programming has an impact on the degree to which the students tend to automatically use this learning technique. This habitual usage can strongly predict if they are willing to use pair programming in the future. Their neutral stance towards pair programming indicates that some may intend to use pair programming again elsewhere in the future, and some students may not.
- *Facilitating Conditions* – this factor had the highest mean of all with 3.76, where majority of the students agreed that they do have the knowledge, and access to the necessary resources required to adequately use pair programming and they feel that it is compatible with the way that they write programs. Provided that pair programming takes place in the computer laboratory and a partner is present, it can be predicted that the students will continue to make use of pair programming with confidence.
- *Behavioural Intention* - with a mean of 3.41 suggests that the students intend to continue implement and use pair programming and its principles to complete programming tasks in the future even though some of them are not majoring in a Computer Science degree. In the literature it was evident that the students understood the pair programming guidelines, had the necessary resources to implement pair programming, and in doing so, enjoyed making use of pair programming with minimal effort. It is clear that most of the students consciously plan to make use of pair programming in the future, but it is difficult to predict if they will actually use pair programming elsewhere in their studies.
- *Pair Compatibility* - with the lowest mean of 2.74 thereby indicates that some of the students found using pair programming difficult due to skill incompatibility, absence and/or tardiness and also communication being an issue due to a language barrier. The

researcher recommends that students in the future should be paired together based on their programming skill levels in order to mitigate the conflict that could arise from incompatibility. This can have a negative impact on the perception that a student may have on programming with a partner and programming overall.

With a mean of 3.79 with regards to the statement “Pair programming taught me teamwork”, it is evident that two minds are better than one. Majority of the students understand that pair programming does increase teamwork and engagement with the task at hand. The benefits of pair programming entails building more resilient code, providing alternating viewpoints from the driver and navigator. It encourages them to share knowledge, and as a result, helps them both to grow their individual skill sets.

From Table 5.2 above, the researcher reported a mean of 2.79 with regards to the statement “I prefer pair programming over solo programming” which indicates that most of the students prefer programming in a pair over programming alone. There is a small group of students that dislike pair programming for various reasons that will be discussed and analysed further in section 5.6.

In summation, the factors with the highest means showed that the majority students are aware of the benefits of pair programming and they are willing to make use of its principles in future courses and in the workplace.

In contrast, the factors with the lowest means showed that there is a minority that dislikes pair programming, and prefers solo programming over pair programming. It is evident that the students have a positive opinion on pair programming and it appears that the majority of the students in this introductory programming course prefer pair programming over solo programming. The frequency of selection of factors for this study are shown below in Figure 5.2 and it is clear that the students in this study had an overall positive experience with pair programming with most factors showing a positive inclination.

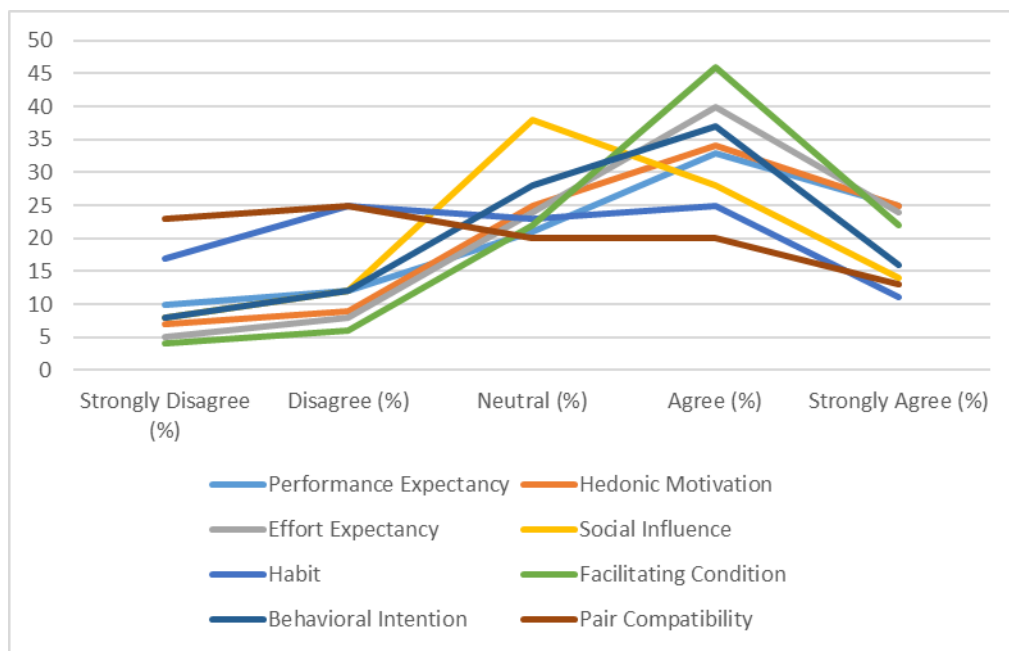


Figure 5.2 Frequency of Selections of Constructs

In Section 5.3 and Section 5.4 the results of the qualitative data are presented and discussed. Since the product of the qualitative research is richly descriptive, the results are presented in the form of a selection of quotes from the students' comments.

5.3 ADVANTAGES OF PAIR PROGRAMMING

In this section the responses to the qualitative question "What are the advantages of pair programming?" are presented and discussed. In concurrence with the quantitative data presented above, the following themes emerged from the responses:

- "fast and efficient"
- "social skills"
- "communication"
- "helping"
- "sharing knowledge"

- “problem-solving”
- “perspectives”
- “teamwork”
- “error detection”
- “ease of use”
- “confidence”
- “none”
- “workload”

The following sub-sections present and discuss the aforementioned pair programming advantages.

5.3.1 FAST AND EFFICIENT

Respondent #03 [Black Male]: *“Gets the job done faster and with accurate codes unlike coding and being unable to spot your logical errors.”*

Respondent #20 [White Male]: *“You can complete tasks quicker.”*

Respondent #44 [White Male]: *“Can complete tasks at a faster pace. Get to learn what you don't know from your partner. Can learn from each other's mistakes.”*

Respondent #272 [Black Male]: *“Pair programming equips students to finish programming tasks quickly and efficiently. It also provides easier means to learn how to code.”*

From the quotes listed above, it is quite clear that pair programming allows the students to complete their programming assignments faster and with higher efficiency than when programming solo. Working in a pair forces the students to use their time wisely because they are forced to focus on the task at hand. This serves as an explanation for the skewed perception

of time that they experience when they are programming in pairs, and they end up feeling that they worked for less time than they actually did in reality.

5.3.2 SOCIAL SKILLS AND COMMUNICATION

Respondent #09 [White Male]: *“You learn to work with people.”*

Respondent #11 [White Female]: *“Communication and programming skills.”*

Respondent #160 [Coloured Male]: *“It helps me to work with others and improves my social skills.”*

Respondent #169 [Black Male]: *“You get to work on your communication and teamwork skills which are essential for you to have in the industry.”*

It is indeed obvious that pair programming is a communication-intensive process and it is of the utmost importance when it comes to working with a partner. In the literature, Williams and Kessler (2003) describe a silent pair as a signal of danger. In order to be a good programming partner, one must be a good listener, articulate properly, debate and discuss code. This constant communication is of paramount importance for success in academia, and in the industry (see Sections 3.3.4 and 3.5.4).

5.3.3 HELPING AND SHARING KNOWLEDGE

Respondent #29 [White Female]: *“Stuff I don’t know my partner knows. She helps me with the parts of programming I struggle with and vice versa. Spotting errors in the code is also easier with two pairs of eyes.”*

Respondent #47 [White Male]: *“Shared knowledge.”*

Respondent #83 [White Male]: *“Makes you focus more on the task than doing it on your own. It is helpful to have a partner to talk with about your ideas and see if he thought the same or knows a better way.”*

Respondent #116 [White Female]: *“you can share your knowledge with your partner and they can also teach you some things.”*

Without a doubt there is a transfer of knowledge that takes place between students during the pair programming process. The two students can answer each other's questions and constant communication makes this knowledge transfer occur quickly which is widely acknowledged as a benefit by a range of studies in the industry (see Section 3.5.3).

5.3.4 PROBLEM SOLVING AND DIFFERENT PERSPECTIVES/SOLUTIONS

Respondent #45 [White Female]: *"You understand a problem more efficiently if you work in pairs, because the other person brings another perspective to a problem. You learn to work in a team and learn more of programming by working with another person (in my case my partner was a more experienced programmer than I am, so I learned a lot from her)."*

Respondent #79 [Black Female]: *"Through the sharing of ideas, we see how various problems have various solutions which all lead to the same answer."*

Respondent #181 [White Female]: *"Your partner might do something differently to how you would, which can sometimes open you up to different ways of doing something."*

Many beginner programmers, unfortunately, have insufficient computational thinking, and programming knowledge, and the expertise to solve a problem. Beginners are required to focus on a range of skills, such as problem-solving and problem design. It is clear that having a programming partner is quite beneficial in the sense that he/she can help his/her partner to solve a problem that is troubling them by presenting a different perspective with regard to a solution. As a result, the partner gains knowledge and learns from his/her partner (see Section 3.5.3).

5.3.5 TEAMWORK

Respondent #131 [Black Male]: *"We work together, team work."*

Respondent #134 [White Female]: *"1. Get to know each other. 2. Team work. 3. Get things done faster. 4. See things from your partner's side too, maybe wouldn't have noticed if partner didn't bring it up."*

Respondent #135 [Coloured Male]: *"You learn how to work together with someone else, which can help in your future job."*

Respondent #146 [Black Male]: *“It is effective and promotes teamwork.”*

Respondent #151 [Black Male]: *“It teaches importance of team work and communication.”*

It is clear from these responses relating to teamwork that most underrepresented students prefer teamwork by means of pair programming when it comes to learning programming languages. In the IT industry, multifaceted software systems are only built by a large team of software developers. It is evident from these responses reported above that the students are aware of the fact that pair programming is mostly used in the IT industry and they can see the great value that this specific pair programming benefit brings forth.

5.3.6 ERROR DETECTION/EASE OF USE

Respondent #33 [White Male]: *“Your partner can easily detect errors while you are typing rather than you debugging them after unsuccessfully running the program.”*

Respondent #69 [White Female]: *“It is easier to program in pairs, because it makes the room for errors smaller as two people are working together, making sure it is right. It also helped me, because I am not very good at programming, I learned a lot from my partner.”*

Respondent #132 [White Male]: *“Faults are found faster and easier.”*

Respondent #215 [White Male]: *“Learning to spot mistakes in code easier.”*

Pair programming is effective in the work environment, and the classroom setting, due to developers or students constantly exchanging ideas, changing driver-navigator roles and in doing so, reviewing their code in order to reduce the number of errors which as a result produces a product or assignment without errors (see Section 3.5.2).

5.3.7 CONFIDENCE

Respondent #22 [White Female]: *“Dit het my baie meer selfvertroue gegee.”*
(Translation: *“It gave me a lot more self-confidence.”*)

Out of the 284 students, only one mentioned self-confidence as an advantage of pair programming in their response. A self-confident student in any aspect of life can handle any kind

of setback with minimal effort. The affirmation received from a partner builds courage which as a result gives their partner the necessary confidence to tackle a problem head-on with no fear compared to when programming solo (Section 3.5.6).

5.3.8 NONE

Respondent #242 [White Female]: *“I do not think there is any disadvantages of pair programming.”*

Respondent #275 [Indian Male]: *“Not a problem.”*

Respondent #281 [Black Female]: *“None that I can think of.”*

It can be assumed that the students above that had no issues with pair programming. These students implemented and used pair programming correctly according to the set pair programming principles and guidelines. It is clear that they prefer pair programming over solo programming and will most likely accept and use this methodology in the future.

5.3.9 WORKLOAD

Respondent #172 [White Male]: *“Less work for each individual, divides the workload.”*

Respondent #195 [White Male]: *“Less mark work for the lecturer.”*

With regard to Workload, it is apparent that working in a team together on a software system or programming assignment, it is crucial for the partners to follow the pair programming principles and guidelines in order to effectively divide the workload equally by frequently changing roles and remaining in constant communication with each other (see Section 3.5.8). Pair programming not only benefits students and developers, but educators as well.

Pair programming significantly reduces the amount of time spent grading each individual assignment and it also makes an educator's job easier during class due to a student having a partner with whom to discuss and solve problems (see Section 3.5.8).

5.4 DISADVANTAGES OF PAIR PROGRAMMING

In this section the responses to the qualitative question “What are the disadvantages of pair programming?” are presented and discussed. In concurrence with the quantitative data presented in Section 5.3, the following themes emerged from the responses:

- “conflict”
- “time-consuming”
- “incompatible personalities”
- “skill incompatibility”
- “lack of knowledge and understanding”
- “absence”
- “tardiness”
- “communication”
- “language barrier”
- “ease of use”
- “anxiety”

The following sub-sections present and discuss the aforementioned pair programming disadvantages.

5.4.1 CONFLICT

Respondent #12 [White Male]: *“I have to admit that it wasn't all good and fun. A simple reason is that if you disagree with your partner you can't continue with what you have in mind. This leads to conflict and by the time when you get your chance to code again you have long forgotten what it was you wanted to do.”*

Respondent #64 [White Male]: *“There are often clashes in terms of how to program and I feel that having someone program with you can bring you down and affect performance.”*

Respondent #149 [Black Male]: *“Clash of ideas.”*

Respondent #160 [Coloured Male]: *“You and your pair don’t usually agree on certain things and results in quarrels and delays you from finishing in time.”*

Respondent #250 [Black Female]: *“The disadvantages of pair programming are that when you can’t find a common ground with your partner and disagreements ensue from that. Another thing is being prepared for a practical and having a partner that’s underprepared is frustrating because it feels like the work is being done by one person. The last thing is having a partner that can’t thoroughly explain their logic to their ideas of programming certain questions. This causes a rift because you are not really sure what the program intends to do and when there are errors, it leads to tension because of lack of communication.”*

The use and implementation of pair programming can inadvertently lead to conflict between two people. Conflicts and/or disagreements always come as a surprise and cause upset to a person because he/she do not expect to get into a disagreement with a friend, colleague or someone with whom he/she do not share interests. In this situation, it is better to expect conflict to occur even before it actually happens. It is important for both parties involved in the conflict to stay calm and start the conflict resolution by not making any kind of judgmental statements because these words can make the situation even worse. Be respectful and simply give the other person the benefit of the doubt and exercise positive reinforcement after the conflict has been resolved. In doing so, less time is spent on arguing, and that time can be used to complete the task at hand. The educator has to address conflict resolution during their presentation of the pair programming guidelines in order to avoid unnecessary conflict (see Section 3.6.1).

5.4.2 TIME-CONSUMING

Respondent #07 [White Female]: *“It sometimes takes longer to come to a solution because each has their own approach.”*

Respondent #09 [White Male]: *“Takes a long time because my knowledge in programming is higher than my partner’s knowledge in programming.”*

Respondent #27 [White Male]: *“Waste of time in general, typically both programmers know what has to be done, and is able to do it due to sufficient education. If they do not know what to do they can ask someone who can then give them a brief explanation as to what to do instead of literally watching over them as they write the program, which effectively cuts productivity in half.”*

Respondent #34 [White Male]: *“Was time-consuming for me since it took longer to finish a practical. Had to continue programming where my partner stopped, which could be confusing. Struggled to communicate my thought process.”*

Time is of the essence and with that being said, Benadé and Liebenberg (2017), in their study about pair programming as a learning method found that pair programming with a partner at a lower skill level consumes more time compared to programming alone. As a result, this time-wasting can completely discourage a person from using pair programming (see Section 3.6.4).

5.4.3 INCOMPATIBLE PERSONALITIES

Respondent #79 [Black Female]: *“Lack of dedication in one of the partners. When partners don’t get on.”*

Respondent #94 [White Female]: *“You and your partner do not always get on.”*

Respondent #114 [White Male]: *“Dit kan jou geduld toets. Dit leer jou dat jy nie altyd met mense goed oor die weg gaan kom by die werk nie.” (Translation: “It can test your patience. It teaches you that you can’t always get along with people at work.”)*

Respondent #164 [White Male]: *“Sometimes can be a little bit frustrating when your partner has habits you’re not used to.”*

Respondent #166 [White Male]: *“If you’re not compatible with your partner and/or their skill is below your skill, it will impact the assignment negatively.”*

Respondent #223 [Black Male]: *“I believe that it’s just about understanding the personality of your partner.”*

From the responses above, it is quite obvious that pair programming is most problematic when the two people involved are incompatible. Students who have a negative pair programming experience with a partner with whom they are incompatible can cause them to detest working in a pair. This psycho-social mismatch affects the effectiveness of pair programming which can be due to personality differences and gender combinations, and programming skill levels (see Sections 3.6.2 and 3.6.3). A person’s personality has a great impact on pair compatibility and can make it very difficult to work with someone.

5.4.4 SKILL INCOMPATIBILITY

Respondent #29 [White Male]: *“The time that we have to switch is sometimes irritating, because the one who is coding wants to do the program a specific way and the other partner has to take over and is maybe not so skilled in this type of method.”*

Respondent #43 [Black Male]: *“Working with someone with a higher level of programming, might make one feel bad about themselves.”*

Respondent #45 [White Female]: *“You can get frustrated with your partner, especially if you are less or more experienced in programming than they are. You have to adapt to another person’s programming style while also trying to incorporate your own.”*

Respondent #49 [Black Male]: *“If your partner thinks he/she is better than you then communication between you two it will be hard as he/she will not want to use your strategies of programming.”*

Respondent #223 [Black Female]: *“If your partner knows more than you, you won’t get up to date.”*

Students face challenges when they are programming in a pair and this can drastically slow down the programming process due to incompatible personalities and differing programming skill levels. As in the case of incompatible personality traits, incompatible skill levels can be problematic in the sense that a beginner programmer would not be able to keep up with their more skilled and

experienced partner. It is possible that the more experienced partner may not have the time to be patient working with their lesser-skilled beginner programmer. They may also feel that pair programming is useless because they are unable to learn anything from the novice due to the large skill gap between them. This can ultimately lead to dissatisfaction from both parties and make them less successful in their programming endeavours. The preference for experienced programmers to work alone or to do all the work by themselves is because of their partner holding them back, therefore, causes frustration, guilt and mistrust (see Section 3.6.2).

5.4.5 LACK OF KNOWLEDGE AND UNDERSTANDING

Respondent #03 [Black Male]: *“Lack of understanding the work makes one rely on another person.”*

Respondent #68 [White Female]: *“When I was busy with an idea and we had to switch places it interrupted the idea I was busy working on with and my partner didn't always understand at first what I was doing.”*

Respondent #106 [Black Male]: *“The inability of the other programmer to prepare and understand concepts of programming.”*

Respondent #150 [Indian Male]: *“Sometimes it's hard to understand what the other person is thinking.”*

Respondent #181 [White Female]: *“My way of understanding code and how certain things work is by doing an assignment and figuring out the solutions myself. Pair programming made it so I don't fully take in what we're doing because I wasn't necessarily the one to be thinking what code to write or solving the problem (my partner would for example think about a solution faster than me which doesn't give me time to come up with a solution on my own).”*

It is challenging for someone with a high programming skill level and vast experience to work with a person who is completely clueless about programming concepts and who is unprepared to solve a problem in order to complete a task. This lack of knowledge can be time-consuming with regard to the entire development process. People think and reason differently; some struggle to articulate

themselves adequately which can cause a misunderstanding amongst the pair (see Sections 3.4, 3.5.3, 3.5.7 and 3.6.1).

5.4.6 ABSENCE/TARDINESS

Respondent #01 [Black Male]: *“My partner is absent sometimes.”*

Respondent #69 [White Female]: *“My partner was very unreliable, so I ended up working with three different partners.”*

Respondent #109 [White Female]: *“If your partner does not show, you need to adapt with someone else.”*

Respondent #158 [Black Female]: *“When a team mate is absent or late it becomes a problem.”*

Students who fail to attend classes not only have a negative impact on their own ability to contribute to the joint assignment, but also on their partner’s ability to work productively. Tardiness puts their partner at a great disadvantage because they are forced to work individually which defeats the whole purpose of pair programming and prohibits them from reaping the benefits of this teaching methodology (see Section 3.6.4).

5.4.7 COMMUNICATION/LANGUAGE BARRIER

Respondent #81 [White Male]: *“Skill level and language create difficulties in communication and takes up extra time and effort.”*

Respondent #90 [White Male]: *“A lack of communication makes programming difficult.”*

Respondent #118 [Black Male]: *“Language might be a barrier and skill levels might differ.”*

Respondent #186 [Coloured Male]: *“Lack of communication. Finding a partner that makes you do all the work.”*

South Africa is a diverse nation that boasts 11 official languages, which makes communication one of the main difficulties that contribute to programming pairs being unsuccessful. It is therefore

not surprising that the students quoted above pointed out the language barrier being a major pair programming disadvantage. In order for pair programming to be executed correctly, communication is of the utmost importance and it makes the programming progress impossible because an Afrikaans speaking student to discuss, debate, and provide a solution for a problem with an isiXhosa speaking student. As a result, this inability for two people to talk to each other can lead to conflict and deter both students from making use of pair programming in the future (see Section 3.6.1).

5.4.8 ANXIETY

Respondent #137 [White Male]: “Social interaction I have anxiety.”

It is true that some people have a social anxiety disorder which is an extreme fear of engaging in social interactions. This mental illness is quite common and can have a long-term impact on a person's life. Any mental illness can make the simplest of tasks complex.

In the classroom setting it can be quite difficult for someone who has social anxiety to work productively with a partner due to his/her crippling fear of being judged or watched by others which can really make the driver-navigator experience quite unpleasant. For those students that have been battling with this phobia since their teenage years, there is hope because this overwhelming fear of social settings can be treated and it does get better as one gets older.

5.5 DIFFERENCES AMONG STUDENTS

When conducting the mixed-methods research, the researcher may find contradictory conclusions from the quantitative, as well as the qualitative strands, however these contradicting findings are valuable in the sense that they enrich the understanding of a phenomenon, in doing so opens new avenues for further research (Venkatesh *et al.*, 2013). In the reported quantitative results presented, it showed that the students felt that pair programming does have a positive influence on the experiences of underrepresented groups. In addition, some of the qualitative results re-affirmed that notion. However, there are comments from the students overall that contradicted the notion that pair programming has a positive influence on their programming experience.

The students who participated in this study were all placed in categories based on their gender, ethnicity, computer and Internet access, IT as a high school subject and their academic performances. The majority of the students were males (70.4%) and the minority were females (29.6%). The gender distribution (Figure 5.3) is depicted in the chart below:

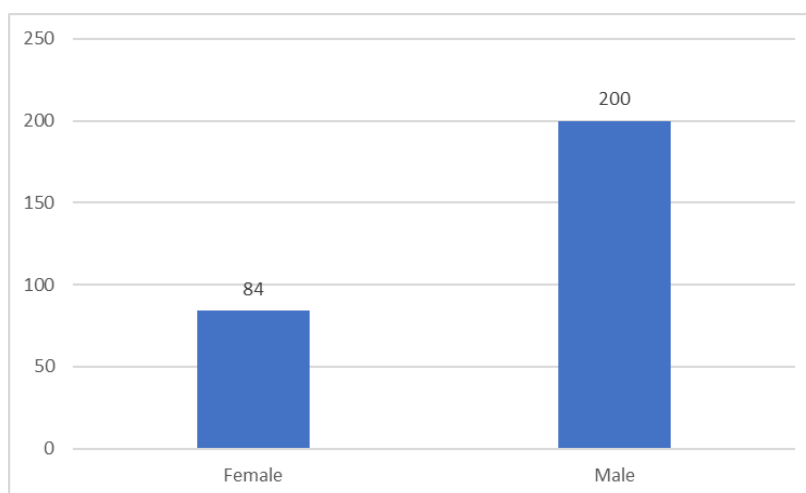


Figure 5.3 Gender Distribution Chart

As mentioned before, the students who participated in this study come from different ethnic backgrounds. The chart (Figure 5.4) below displays the ethnic diversity among the students.

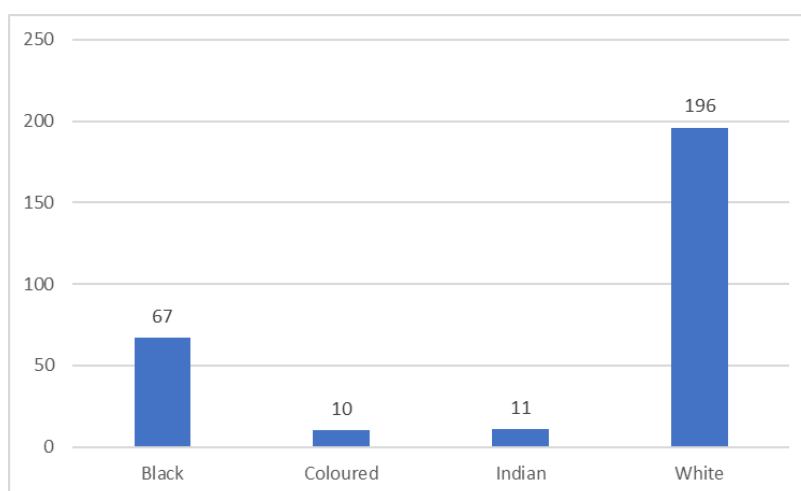


Figure 5.4 Ethnicity Distribution Chart

In the context of this study, the students who identified as Black, Coloured and/or Indian are the minorities (non-white 31%) and the White students make up 69% of this population.

Due to this study only focusing on access to computers and the internet at educational institutions, the statistical analysis for the “I have a computer at home” category was omitted. By means of using T-tests, all the categories previously mentioned, were tested for any practically visible differences or significant differences between the eight UTAUT2 factors used for this study. The Bartlett p-values are only presented for the sake of completeness, and will not be interpreted, since a convenience sample is used instead of a random sample.

5.5.1 GENDER

Based on the reported effect size ($d = 0.52$) in Table 5.3 below, there is a practically significant difference between the means of the Habit factor of the female students and the male students. This means that the female students (mean = 3.29) agreed more with regard to the statements within the Habit factor whereas their male counterparts (mean = 2.73) agreed significantly less. This indicates that female students are more inclined to make use of pair programming principles because they believe that the implementation of the specified principles is an automatic behaviour (a habit), in contrast to the male students who believe otherwise.

Table 5.3 – Differences based on Gender

Gender	Male (n=200)		Female (n=84)		Effect Size (d)	p-value
Factors/Statements	Mean	St. Dev	Mean	St. Dev		
Performance Expectancy	3.351	1.136	3.875	1.023	0.46	<0.001
Hedonic Motivation	3.485	1.101	3.877	1.045	0.36	<0.050
Habit	2.728	1.083	3.293	1.045	0.52	<0.001
Pair programming taught me teamwork.	3.670	0.998	4.080	0.839	0.41	<0.001
I prefer pair programming over solo programming.	2.780	1.297	3.440	1.484	0.45	<0.001

5.5.2 ETHNICITY

Based on the reported effect size ($d = 0.29$) as seen in Table 5.4 below, there is a small difference between the Compatibility of the majority students and the minority students. This suggests that the majority students with (mean = 2.660) on average felt more negative with regard to the statements within this factor whereas the minority students (mean = 2.920) on average felt neutral. This indicates that compatibility in terms of ethnic origin was not a problem due to the fact that the students were given the liberty to choose their partner who more often than not shared the same ethnic origin. It is interesting to note that there is a small to medium practical significant

difference ($d = 0.35$) between the ethnic minority students ($n = 88$), and the ethnic majority students with regard to their programming preferences. The minority students prefer pair programming (mean = 3.32) in comparison with the majority students who prefer solo programming (mean = 2.820).

Table 5.4 – Differences based on Ethnicity

Ethnicity	Majority (n=196)		Minority (n=88)		Effect Size (d)	p-value
Factors/Statements	Mean	St. Dev	Mean	St. Dev		
Performance Expectancy	3.367	1.193	3.815	0.898	0.38	<0.001
Behavioral Intention	3.284	1.016	3.678	0.907	0.39	0.084
Pair Compatibility	2.660	0.796	2.920	0.903	0.29	<0.050
I prefer pair programming over solo programming.	2.820	1.424	3.320	1.237	0.35	<0.050

5.5.3 COMPUTER ACCESS SINCE GRADE 1

Based on the reported effect size ($d = 0.03$) shown in Table 5.5 below, there is no practically significant difference between the mean scores of the students who had access to a computer since Grade 1 and the students who had no access to a computer since Grade 1. This simply means that the students who had access to a computer with (mean = 2.95) on average felt somewhat neutral. Similarly, the students who had no access to a computer since Grade 1 felt rather neutral with regards to their programming preference.

Table 5.5 – Differences based on Computer Access since Grade 1

Computer Access since Grade 1	Access (n=134)		No-access (n=150)		Effect Size (d)	p-value
Factors/Statements	Mean	St. Dev	Mean	St. Dev		
Performance Expectancy	3.438	1.198	3.566	1.062	0.11	0.343
Behavioral Intention	3.343	1.016	3.462	0.983	0.12	0.319
Pair Compatibility	2.712	0.817	2.766	0.858	0.06	0.588
I prefer pair programming over solo programming.	2.947	1.394	2.993	1.383	0.03	0.783

5.5.4 INTERNET ACCESS SINCE GRADE 1

The reported effect size ($d = 0.28$) as presented in Table 5.6 below, shows that there is a small, practically no difference between the Facilitating Conditions of the students who had access to the Internet since Grade 1 and the students who had no access to the Internet. This means that the students with access to the Internet with (mean = 3.69) on average felt neutral. The students

with no access to the Internet with (mean = 3.90) on average completely agree with regard to the Facilitating Conditions factor. This simply means that the students with no Internet access believe to a great extent that they have the technical infrastructure at their disposal to support the use of pair programming in an introductory programming course.

Table 5.6 – Differences based on Computer Access since Grade 1

Internet Access since Grade 1	Access (n=190)		No-access (n=94)		Effect Size (d)	p-value
Factors/Statements	Mean	St. Dev	Mean	St. Dev		
Social Influence	3.225	0.837	3.461	0.851	0.28	<0.050
Facilitating Conditions	3.687	0.766	3.901	0.655	0.28	<0.050
I prefer pair programming over solo programming.	2.830	1.319	3.260	1.481	0.29	<0.050

5.5.5 IT AS A HIGH SCHOOL SUBJECT

In Table 5.7 below, the reported effect size of ($d = 0.76$) displays a large, practically significant difference between the means of the students who had IT as a subject in high school with (mean = 2.25) and those who did not have IT as a high school subject (mean = 3.26) with regard to their programming preferences. This means that the students who had IT as a high school subject on average feel negative towards preferring pair programming over solo programming, whereas the students who did not have IT in high school are on average neutral, and felt a bit positive towards their programming preference. With regards to pair programming teaching teamwork, the non-IT students feel neutral and are leaning towards strongly agree with a mean score of 3.89.

Furthermore, there is another large practically significant difference between the IT and non-IT students with a reported effect size of ($d = 0.73$). The former with (mean = 2.34) on average feel negative towards the Habit factor item compared to the latter with a reported mean of (mean = 3.12), indicating that the non-IT students feel completely neutral to almost agree. Consequently, this positive stance has a direct and indirect effect through Behavioural Intention. With regards to Behavioural Intention, the mean for the non-IT students (mean = 3.6) indicates that this group feels more positive. It is also interesting to see that this same group of students believes to a high degree that using pair programming helped them to better their academic performances and as a result enhanced the quality of their work (mean = 3.77). This is a clear indication that the students who did not have IT as a high school subject prefer pair programming over solo programming due to the benefits of pair programming.

Table 5.7 – Differences based on IT as a High School Subject

IT as High School subject	IT (n=81)		Non-IT (n=203)		Effect Size (d)	p-value
	Mean	St. Dev	Mean	St. Dev		
Performance Expectancy	2.843	1.242	3.770	0.961	0.75	<0.001
Hedonic Motivation	3.218	1.151	3.754	1.039	0.47	<0.001
Social Influence	3.045	0.854	3.406	0.825	0.42	<0.001
Habit	2.342	1.058	3.116	1.042	0.73	<0.001
Behavioral Intention	2.996	1.033	3.569	0.939	0.56	<0.001
Pair programming taught me teamwork.	3.540	1.013	3.890	0.938	0.34	<0.050
I prefer pair programming over solo programming.	2.250	1.250	3.260	1.333	0.76	<0.001

5.5.6 ACADEMIC PERFORMANCE

Table 5.8 displayed below, with the reported effect size of ($d = 0.48$) with regard to the Performance Expectancy factor indicates a medium, practically visible difference between the students who achieved a distinction in previous programming modules (mean = 3.08) and the students who achieved a pass with a (mean of 3.72). The students with a distinction on average feel neutral, and felt negative with regard to Performance Expectancy. On the other hand, the pass students are neutral and felt more positive. The students who achieved a pass believe that pair programming can help them improve their academic performances.

With regard to the Hedonic Motivation factor, the effect size ($d = 0.31$) which displays a small, practically significant difference between the distinction students (mean = 3.36) and the pass students with a (mean = 3.74). Both respondent groups are neutral, and felt positive which means that both groups to a certain extent feel a degree of happiness as a result of programming in a pair.

It is interesting to note that the students with a distinction (mean = 3.5) with regard to the 'Pair programming taught me teamwork' on average agree that pair programming indeed does teach teamwork, but these same students prefer solo programming over pair programming due to their mean of 2.55 suggesting that they are neutral, and felt negative with regards to their programming preference. On the contrary, the students who achieved a pass feel neutral, and were positive with regard to their programming preference (mean = 3.15). This means that they prefer pair programming over solo programming. To further support this finding, this same group of students feel positive almost strongly agree with the 'Pair programming taught me teamwork' item with (mean = 3.94).

Table 5.8 – Differences based on Academic Performance

Academic Performance		Distinction (n=101)		Pass (n=175)		Effect Size (d)	p-value
Factors/Statements		Mean	St. Dev	Mean	St. Dev		
Performance Expectancy		3.082	1.345	3.721	0.916	0.48	<0.001
Hedonic Motivation		3.356	1.265	3.743	0.979	0.31	<0.050
Habit		2.508	1.121	3.080	0.963	0.47	<0.001
Behavioral Intention		3.158	1.171	3.531	0.855	0.32	<0.050
Pair programming taught me teamwork.		3.495	1.101	3.942	0.835	0.41	<0.001
I prefer pair programming over solo programming.		2.545	1.389	3.150	1.324	0.44	<0.001

In summary, the results above show that Performance Expectancy, Hedonic Motivation, Facilitating Conditions, and the Habit factor had a positive effect with regard to the Behavioural intention to implement and use pair programming. It is clear from this current study and prior findings in the literature that the students strongly believe that their academic performances, and their quality of work will improve, if they correctly use pair programming and reap the benefits thereof. Minority students are more likely to pursue IT-related courses if the supporting technical infrastructure is available to them (Liebenberg *et al.*, 2018; Liebenberg *et al.*, 2012; Nawrocki & Wojciechowski, 2001; Venkatesh *et al.*, 2012; Venkatesh *et al.*, 2016; Williams & Kessler, 2000a; Williams & Kessler, 2001).

5.6 RELATIONSHIPS BETWEEN RACE AND GENDER AND THE UTAUT2 FACTORS

According to Kenton (2021), analysis of variance (ANOVA) is an analysis tool that is used in statistics which divides an observed aggregate variability found inside a data set into two parts, systematic factors and random factors. This is done in order to determine the statistical influence on a given data set. The researcher used ANOVA to determine what influence independent variables have on the dependent variable. (Kenton, 2021; Lakens, 2013) Effect sizes are used to determine the sample size to examine effects across studies, therefore it is the most important outcome of empirical studies due to the fact that it emphasises and communicates the practical significance of results (Lakens, 2013). For the purpose of this study, performance expectancy is the degree to which an individual believes that his/her performance would increase using pair programming (Izuagbe, 2021).

5.6.1 PERFORMANCE EXPECTANCY

The results of the ANOVA test between Female White with Female Non-White ($d = 0.21$) indicates a small effect which means that the females in the class regardless of their ethnic origin, believe that pair programming is useful. However, when comparing the effect size between White Males and White Females ($d = 0.72$), the indication is that there is a large impact and it can be interpreted as the group of White Males not finding pair programming useful or productive. The performance expectancy between White Males and Non-White females has an effect size of 0.51 which shows that there is a large effect between the two groups and it also shows that Non-White Females believe that pair programming increases their chances of improving their marks. There is a small effect when comparing Non-White Males with Non-White Females ($d = 0.16$) and White Females ($d = 0.08$), which indicates that the minorities overall believe that pair programming allows them to complete tasks quicker, and it also improves their productivity. There was also a significant difference when comparing White Males and Non-White Males with an effect size of 0.64, indicating that White Males do not find pair programming useful as they prefer programming solo (see Table 5.9).

Table 5.9 ANOVA – Performance Expectancy

					Effect Size		
					Female White with	Female NonWhite with	Male White with
Performance Expectancy	Female White	58	3.948	1.111			
	Female Non-White	26	3.711	0.786	0.212		
	Male White	138	3.123	1.145	0.720	0.513	
	Male Non-White	62	3.858	0.943	0.080	0.156	0.642
	Total	284	3.506	1.128			

5.6.2 HEDONIC MOTIVATION

The ANOVA test conducted on Hedonic Motivation, points to a large effect with an effect size of 0.57 when comparing White Males and White Females. The result between Non-White Males and White Females has a small effect ($d = 0.17$) which points to the two groups finding pair programming fun and enjoyable.

Table 5.10 ANOVA – Hedonic Motivation

		N	Mean	Std. Deviation	Effect Size		
					Female White with	Female NonWhite with	Male White with
Hedonic Motivation	Female White	58	3.994	1.078			
	Female NonWhite	26	3.615	0.931	0.351		
	Male White	138	3.338	1.154	0.568	0.240	
	Male NonWhite	62	3.811	0.896	0.169	0.210	0.410
	Total	284	3.600	1.097			

Similarly, Non-White Males and Non-White Females believe that pair programming is fun and enjoyable yielding an effect size of 0.21 which indicated a small effect. However, when comparing White Males and Non-White Males with ($d = 0.41$), it is clear that with this large effect, White Males do not find pair programming fun nor enjoyable.

5.6.3 EFFORT EXPECTANCY

As seen in the table below (see Table 5.11), Effort Expectancy has a large effect on Non-White Males and Non-White Females with an effect size of 0.50 which indicates a significant difference between both these groups. This means that the minorities find pair programming easy to use and it can make them become skilful programmers. In contrast, there is a small effect between White Males and Non-White Males ($d = 0.26$), which suggests that White Males do not find pair programming easy to use and they are reluctant to utilise it in order to become more skilful programmers.

Table 5.11 ANOVA – Effort Expectancy

		N	Mean	Std. Deviation	Effect Size		
					Female White with	Female NonWhite with	Male White with
Effort Expectancy	Female White	58	3.905	0.861			
	Female NonWhite	26	3.403	0.818	0.581		
	Male White	138	3.606	0.938	0.317	0.216	
	Male NonWhite	62	3.854	0.898	0.056	0.502	0.264
	Total	284	3.703	0.913			

5.6.4 SOCIAL INFLUENCE

With regard to Social Influence, there is a large effect between the White Males and the Non-White Males. Between these two groups ($d = 0.43$) lies a significant difference in the sense that Non-White Males who prefer pair programming value each other's opinions and they believe that they should use pair programming to complete programming tasks. There was also a small effect

between White Females and Non-White Females ($d = 0.14$) and they both agree that the people who are important to them also think that they should use pair programming to complete programming tasks. When comparing the Social Influence between White Males and Females in general both White ($d = 0.30$) and Non-White ($d = 0.17$) indicates a significant difference in the sense that White Males in the classroom setting do not use pair programming and so not see it as useful to complete tasks efficiently.

Table 5.12 ANOVA – Social Influence

					Effect Size		
		N	Mean	Std. Deviation	Female White with	Female NonWhite with	Male White with
Social Influence	Female White	58	3.425	0.889			
	Female NonWhite	26	3.294	0.576	0.146		
	Male White	138	3.149	0.824	0.309	0.176	
	Male NonWhite	62	3.532	0.902	0.118	0.263	0.423
	Total	284	3.302	0.848			

5.6.5 HABIT

As shown in the table below (see Table 5.13), there is indeed a significant difference between Non-White Males and White Males with the effect size of 0.47. This large effect, points to the fact that White Males do not rely on pair programming to complete programming tasks and the use thereof feels unnatural to them. However, there is a small effect between Non-White Males and Non-White Females ($d = 0.01$) and White Females ($d = 0.28$) which implies that these minority groups together agree that using pair programming has become natural and is a habit for them, and they rely on it to complete all their programming tasks. Strangely enough, there is a large effect between White Males and White Females ($d = 0.76$). This significant difference between the groups suggests that White Males do not rely on their partners to complete programming tasks.

Table 5.13 ANOVA – Habit

					Effect Size		
		N	Mean	Std. Deviation	Female White with	Female NonWhite with	Male White with
Habit	Female White	58	3.385	1.075			
	Female NonWhite	26	3.089	0.964	0.274		
	Male White	138	2.572	1.057	0.755	0.488	
	Male NonWhite	62	3.075	1.068	0.287	0.013	0.470
	Total	284	2.895	1.101			

5.6.6 FACILITATING CONDITIONS

Despite having access to the necessary resources and knowledge to use pair programming it is evident that White Males find pair programming incompatible with their preferred programming style when paired together compared to White Females with an effect size of 0.54. In contrast, the effect size of 0.10 between Non-White Males and White Males and the effect size of 0.32 between Non-White Females, indicates that these minority groups, in terms of facilitating conditions, believe that pair programming is compatible with their programming style.

Table 5.14 ANOVA – Facilitating Conditions

		N	Mean	Std. Deviation	Effect Size		
					Female White with	Female NonWhite with	Male White with
Facilitating Conditions	Female White	58	3.879	0.697			
	Female NonWhite	26	3.50	0.464	0.544		
	Male White	138	3.743	0.725	0.186	0.336	
	Male NonWhite	62	3.784	0.868	0.108	0.328	0.047
	Total	284	3.758	0.737			

5.6.7 BEHAVIOURAL INTENTION

In terms of Behavioural Intention, there is a statistical significance when comparing Non-White Males and White Males. The large effect size of 0.56 between these two groups, points to the White Males not intending to use pair programming, even in the future elsewhere in their studies to complete tasks. There is a small effect that exists between Non-White Females and Non-White Males ($d = 0.2$) which points to their belief that they will continue using pair programming to complete programming tasks and they intend to use the principles of pair programming in the future. Similarly, White Females also intend to use pair programming in the future to complete programming tasks when compared to Non-White Females ($d = 0.08$) and Non-White Males ($d = 0.13$). These small differences between the minority groups point to them favouring pair programming, unlike the White Male population in the class.

Table 5.15 ANOVA – Behavioural Intention

		N	Mean	Std. Deviation	Effect Size		
					Female White with	Female NonWhite with	Male White with
Behavioural Intention	Female White	58	3.609	0.851			
	Female NonWhite	26	3.538	0.718	0.083		
	Male White	138	3.147	1.051	0.439	0.371	
	Male NonWhite	62	3.736	0.975	0.130	0.203	0.560
	Total	284	3.406	0.999			

5.6.8 PAIR COMPATIBILITY

With respect to Compatibility, there is a small effect that is evident between Non-White Males and Non-White Females ($d = 0.06$). This statistical significance implies that these two groups are compatible with their partners and do not find pair programming difficult to use even if their partner was on a different programming skill level. Furthermore, there is a medium effect between Non-White Females and Non-White Males ($d = 0.26$) which indicates that these two groups believe that pair programming taught them how to work as a team in order to complete programming tasks. The medium difference ($d = 0.31$) that exists between White Males and Non-White Males points to White Males agreeing that they were incompatible with their partner and it clearly indicates that they prefer working alone to working with a partner on a programming task.

However, a major factor, such as the lack of communication could be the reason for this difference which can make pair programming challenging.

Table 5.16 ANOVA – Pair Compatibility

		N	Mean	Std. Deviation	Effect Size		
					Female White with	Female NonWhite with	Male White with
Pair Compatibility	Female White	58	2.698	0.860			
	Female NonWhite	26	2.875	0.831	0.205		
	Male White	138	2.644	0.769	0.061	0.276	
	Male NonWhite	62	2.939	0.938	0.257	0.068	0.313
	Total	284	2.741	0.837			

5.6.9 TEAMWORK

In view of pair programming teaching the students Teamwork, it is evident that there is a significant statistical difference between White Males and Non-White Males. This large effect ($d = 0.44$) between the two groups, indicates that White Males did not learn how to work as a team throughout the course of the semester. The same can be said for the difference between White

Males and White Females where the effect size was 0.64. Furthermore, there was also a medium effect between those same White Males and Non-White Females ($d = 0.30$) which is in accordance with their dislike of pair programming. The Non-White Males and Females showed a small effect ($d = 0.13$) when compared to each other which shows that they both learnt teamwork by using pair programming.

Table 5.17 ANOVA – Teamwork

					Effect size		
					Female White with	Female NonWhite with	Male White with
Teamwork	Female White	58	4.189	0.712			
	Female NonWhite	26	3.846	1.046	0.328		
	Male White	138	3.528	1.026	0.643	0.303	
	Male NonWhite	62	3.983	0.858	0.239	0.131	0.443
	Total	284	3.792	0.970			

5.6.10 PAIR PROGRAMMING OVER SOLO PROGRAMMING

With regard to programming preferences, it is clear that White Females and Non-White Females have a small difference of 0.12 which suggests that both groups prefer pair programming over solo programming. In contrast, when comparing the effect size between White Males and White Females, it is evident that White Males prefer solo programming over pair programming due to the large effect size ($d = 0.63$). The same can also be said about the effect size between White Males and Non-White Females ($d = 0.58$), which further implies their preference for solo programming. In addition, the effect sizes between Non-White Males and Non-White Females is insignificant with a value of ($d = 0.11$). Similarly, there is also a small effect size between Non-White Males and White Females ($d = 0.01$).

On the other hand, a large difference between Non-White Males and White Males ($d = 0.63$). It is remarkable to note that the effect size between White Males and White Females is exactly the same as the effect size ($d = 0.63$) between White Males and Non-White Males ($d = 0.63$). The results from this ANOVA analysis proves that in general, White Males do indeed have a strong preference for solo programming over pair programming. When taking into account all of the aforementioned findings, the entire group have a positive opinion on pair programming which means that the White Males lower the mean which, as a result, indicates that the ethnic minorities feel even more positive about pair programming. Thus, this significant finding from this study answers the research question, and substantiates the great impact that pair programming has on

the underrepresented students' experiences and perceptions of Information Technology in South Africa.

Table 5.18 ANOVA – Pair programming over Solo programming

		N	Mean	Std. Deviation	Effect size		
					Female White with	Female NonWhite with	Male White with
Pair programming over Solo Programming	Female White	58	3.50	1.547			
	Female NonWhite	26	3.307	1.349	0.124		
	Male White	138	2.528	1.268	0.627	0.576	
	Male NonWhite	62	3.322	1.198	0.114	0.011	0.625
	Total	284	2.971	1.386			

5.7 CONTRADICTIONARY CONCLUSIONS

When conducting the mixed-methods research, the researcher may find contradictory conclusions from the quantitative, as well as the qualitative strands, however these contradicting findings are valuable in the sense that they enrich the understanding of a phenomenon, in doing so opens new avenues for further research (Venkatesh *et al.*, 2013). In the reported quantitative results presented above, it showed that the students felt positive about pair programming over solo programming, and some of the qualitative results re-affirmed that notion. However, the comments of the white male below contradicted the notion that the entire group prefers pair programming over solo programming, when there are a few that dislike pair programming due to their complaints about pair programming becoming a habit and compatibility.

Respondent #81 [White Male]: *“Due to the nature of people and their group dynamic as a whole being forced into pair programming can be compared to communism, even though it sounds great in theory with equal work distribution, the work distribution will always be off-balance one way or another and at the end of the day those involved in the group losses their passion for a wonderful subject as great as C#. Above all that, there is no pride in the work, the piece of work that is completed is always "our" work, sure pride can be taken in the result of a collective effort, but the work tend to be little fun projects given in a stressful environment (noisy & unhygienic labs, slow computers, multiple parts to complete that tend to be more just because you're in a group), on a deadline and it kind of ruins the whole experience; and even if you do enjoy the piece of work despite the bad environment you can't call it your own, you can't make any project your own; even if you're the one that made the project a success you can't take full recognition because then you're*

doing something that is frowned upon. Pair programming is a necessity in many cases, but small things like this make it an awful experience.”

It can be deduced that this respondent prefers solo programming over pair programming. The respondent is probably not the only one to feel this way towards collaborative learning. For him, he prefers having a complete control over whatever he is developing because it allows him to solve problems in his own way. However, he is definitely the only one in this group of respondents to compare pair programming to communism.

5.8 SUMMARY

In this chapter, the results of the statistical analysis were presented and discussed. In Chapter 6, based on the analysed results in previous chapters, recommendations will be made and it will also conclude this study.

CHAPTER 6 CONCLUSION AND RECOMMENDATIONS

6.1 SYNOPSIS OF THIS STUDY

This research study on pair programming and underrepresented groups in Information Technology in South Africa consists of six chapters. In Chapter 1, the problem, research aim and objectives, methods of investigation and design, chapter division and core literature references were stated. In this chapter the requirement of adequately qualified ICT employees who traditionally belong to the underrepresented groups in IT in South Africa's IT industry, in addition to the shortage of said minorities in IT-related courses in universities and other institutions of higher learning were highlighted. Pair programming was suggested as an intervention tool in order to address the factors that potentially limit underrepresented groups' participation in IT-related fields. Pair programming in the context of this research study is quite promising in the sense that it shows that it might attract and increase the retention rate of underrepresented groups in IT in South Africa.

In Chapter 2, a literature review of underrepresented groups in IT was undertaken. This chapter further provided a discussion on the historical view of STEM fields in South Africa, the digital divide among minorities as well as the closing thereof and women in IT. In addition, the various efforts made to increase the participation of racial, and gender-based minorities in STEM fields were discussed.

In Chapter 3, a literature review of pair programming was given. In this chapter, the use of pair programming in the industry in terms of the development methodologies that are implemented with regard to software development as a whole was discussed. The discussion also focused on pair programming in the context of the classroom, highlighting the guidelines, the educational advantages of implementing pair programming and its challenges in an introductory programming course.

In Chapter 4, the research methodology of this study was detailed. A discussion was provided of the positivistic research paradigm used, the demographic information of the students involved, the research design methodology in terms of data collection methods and instruments, and the description of how the data was analysed for this study.

In Chapter 5, the descriptive statistical results were presented and discussed, followed by the themes that emerged in the qualitative data extracted from the open-ended questions in the questionnaire, and the results of variance with regard to the students' demographic data.

In this final chapter, a discussion and interpretation of the research results in light of the research question set for this study are provided. The conclusions arising from the results are drawn, followed by recommendations resulting from the conclusions.

6.2 DISCUSSION OF THE FINDINGS OF THIS STUDY

The interpretation and discussion of the research results occur in light of the study objectives:

- To examine the literature on the underrepresented groups in Information Technology;
- To examine the literature on pair programming; and
- To investigate the impact of pair programming on the students' experiences and perceptions.

The discussion focuses more explicitly on the empirical findings than on the literature, because the literature studied in Chapters 2 and 3 formed a foundation for data collection and analysis in the empirical study in Chapter 4 and Chapter 5. The literature is therefore implicitly and intrinsically part of the study and its findings.

6.2.1 SYNOPSIS OF THE LITERATURE STUDY

Firstly, a synopsis of the findings of the literature study is given.

6.2.1.1 UNDERREPRESENTED GROUPS IN IT

In Africa, there is a critical shortage of graduates with adequate computing skills. Regardless of the various efforts made in the industry on a global scale to increase the representation of ethnic minorities in STEM fields, these groups remain underrepresented. Women, people of colour, and people with disabilities are essential in closing this perceived gap in STEM fields. In South Africa, historical and political policies are responsible for the production of scientific knowledge and the education of the future scientists of this country.

Marred by the governance of the Apartheid regime, people of colour in South Africa were subjected to inferior levels of education established by the Bantu Education Act of 1953. The 1997 Higher Education Act was instated in order to address the past inequalities and to overcome racism and sexism in institutions of higher education. There exists a digital divide in this country due to the high level of poverty. This divide is an area of great concern in South Africa in the sense that it has a great impact on the millions of households in the country. These countless households have limited or no access to educational programs that involve computing.

Over the decades, the use of computers, as well as the internet has grown at a slow rate. Due to poverty and racial segregation, the majority of South Africans who are black are mostly uneducated and those who are educated received their education from low-income schools. These schools have inadequately trained educators who are unable to teach their students the basic technical skills that are required to use a computer and the internet, whereas their affluent counterparts who are the minority in this country have the luxury of learning these skills in the comfort of their homes. The low level of education of these individuals who live in impoverished communities is limited to low-skill jobs that do not require the use of computers and the internet.

In order to close this digital divide, it is important to acknowledge that adequate educational infrastructure plays a major role in society. It enhances the character and self-belief of these individuals when they are young so that they can set high expectations and goals for themselves. It is absolutely imperative that the government provides them with the necessary assistance in the form of technological equipment and coding initiatives in order for them to attain a high academic standing and, as a result, get employed in the IT world and achieve a high standard of living.

6.2.1.2 PAIR PROGRAMMING

Pair programming is an example of an Extreme Programming practice, where two programmers collaborate at the same computer to complete a task. In the process, both of these programmers gain knowledge from each other.

In the midst of the forthcoming Fourth Industrial Revolution, software plays a significant role in our daily lives and software development has always been regarded as a notoriously difficult task

to perform. In the software development industry, the delivery of a high-quality software program is the primary objective for all the developers involved in the process.

The quality of the software is dictated by the software development approach used in order to deliver the final product. A Software Development Life Cycle is a layout of the framework for planning, controlling, in addition to the methodologies and agile software development models used in the development of the software program. Pair programming as an Extreme Programming practice ensures that every individual involved in a project works together as a team and strives to write simple code that saves time and effort. This aims to produce a final product of high quality by reducing the number of defects, development costs and in doing so accentuates customer satisfaction.

Pair programming focuses on the principle of sharing expertise across the teams involved in a project which is essential in such a knowledge-based activity. As beneficial as this practice is, it can be detrimental to the development process of a project if it is implemented in the wrong environment. Overall pair programming is effective in the work environment because the developers are constantly brainstorming, and changing roles by reviewing their code which reduces the number of defects to a great extent.

Pair programming has risen to popularity alongside Extreme Programming in both the IT industry and educational fields. Solving problems is a sought-after skill in the pursuit of being successful in learning. Typically, introductory programming classes for undergraduates require the students to individually complete assignments that are content-specific. In pair programming, one student assumes the role of the “driver” who is tasked with coding by means of using the keyboard and mouse. The other student assumes the role of the “navigator” and he/she is responsible for searching for possible solutions to a coding problem by observing and analysing the code written by the driver.

In the academic environment, pair programming has been regarded as a promising learning technique. Several students in the past have confirmed it improves their comprehension of programming concepts. It is important to note that the collaborative aspect of pair programming exposes students to teamwork and it improves their communication skills that are required in the IT industry. It is crucial that the students are familiarised with the principles of pair programming

and that they correctly implement the guidelines in order to reap all the benefits that are associated with pair programming.

A discussion of the findings arising from the empirical study follows.

6.2.2 DISCUSSION OF THE FINDINGS OF THE EMPIRICAL STUDY

The empirical study aimed to investigate the impact of pair programming on the experiences and perceptions of underrepresented groups in Information Technology in South Africa

The participants in this research study were first-year BSc IT students (n=284) from different genders, ethnic groups, cultures, and socio-economic backgrounds. The students who identified as Black, Coloured, Indian and/or Asian were categorised as the minority and those who identified as White as the majority. In terms of ethnic origin, only 31% of the students identified themselves as minorities who in reality belong to the majority population of South Africa. Only 29.6% of the students were women which is to be expected from the typical gender profile of most IT classes which indicates that they are a minority within a minority. There is a digital divide amongst the participants as a whole with 33.1% of the students, being the minorities, who do not have access to the internet at home. Furthermore, 28.5% of the students are in the minority with regard to having prior knowledge of the subject in high school (see Section 4.3.3).

The study found that the factors with the highest means (Facilitating Conditions, Effort Expectancy, Hedonic Motivation, Performance Expectancy, Social Influence, and Behavioural Intention) showed that the majority of students are aware of the benefits of pair programming and they are willing to make use of its principles in future courses and in the workplace (see Section 5.2).

Facilitating Conditions – The students agreed that they had the necessary knowledge and resources to adequately use pair programming. The students felt that pair programming is compatible with their preferred writing style. It is evident that the majority of the students know that they have the organisation and technical infrastructure to support the use of pair programming.

Effort Expectancy - The students agreed that learning how to use pair programming was an easy task that required minimal effort. The students found the guidelines and principles

of pair programming to be clear and understandable and they could easily become skilful programmers.

Hedonic Motivation - The students reported that pair programming is fun, enjoyable and entertaining. It can be assumed that most of the students do derive fun from using pair programming and this plays a major role in their acceptance of using this learning technique.

Performance Expectancy - There is a mostly positive feeling with regards to the degree to which the students believe that using pair programming will aid in increasing their overall performance and marks. It is evident that students agree that pair programming is useful and that pair programming allows them to improve their marks. Pair programming helps them to complete their programming tasks rapidly and increases their productivity.

Social Influence - The students felt somewhat indifferent about the beliefs of significant people in their lives and their fellow students who think that they should make use of pair programming. Not surprisingly, students agreed that their lecturer thinks that they should make use of pair programming.

Behavioural Intention – The data suggested that the students intend to continue implementing and using pair programming and its principles in the completion of programming tasks even though some of them are not majoring in a Computer Science degree.

In contrast, the factors with the lowest means (Habit and Pair Compatibility) showed that pair programming might not come naturally to all the students and the compatibility of pair programming partners might be an obstacle in the successful use of pair programming.

Habit - Some of the students struggled to make it a habit of using pair programming to complete programming tasks. Learning to use pair programming has an impact on the degree to which the students tend to automatically use this learning technique. This habitual usage can strongly predict if they are willing to use pair programming in the future.

Pair Compatibility - Some of the students found using pair programming difficult due to skill incompatibility, absence and/or tardiness of their partner and also communication being a concern due to a language barrier.

Preference of programming method - It is evident that the students have a positive opinion on pair programming and it appears that the majority of the students in this introductory programming course prefer pair programming over solo programming.

Teamwork - It is evident that two minds are better than one. The majority of the students understand that pair programming does increase teamwork and engagement with the task at hand.

The following advantages of pair programming emerged from the qualitative data:

Fast and efficient - Working in a pair, forces the students to use their time wisely because they are forced to focus on the task at hand. This serves as an explanation for the skewed perception of time that they experience when they are programming in pairs, and they end up feeling that they worked for less time than they actually did in reality (see Section 5.3.1).

Social skills and communication - To be a good programming partner, one must be a good listener, articulate properly, debate and discuss code. This constant communication is of paramount importance for success in academia, and in the industry (see Sections 3.3.4 and 5.3.2).

Helping and sharing knowledge - There is a transfer of knowledge that takes place between students during the pair programming process. The two students can answer each other's questions and constant communication makes this knowledge transfer occur quickly which is widely acknowledged as a benefit by a range of studies in the industry (see Sections 3.5.3 and 5.3.3).

Problem-solving and different perspectives/solutions- Many novice programmers have insufficient computational thinking, and programming knowledge, in addition to the expertise to solve a problem. It is clear that having a programming partner is quite beneficial in the sense that he/she can help his/her partner to solve a problem that is

troubling them by presenting a different perspective with regard to a solution (see Section 3.3.1 and 5.3.4).

Teamwork - It is clear from the responses relating to teamwork that most underrepresented students prefer teamwork by means of pair programming when it comes to learning programming languages (see Sections 5.3.5).

Error detection/Ease of use - Pair programming is effective in the work environment, and the classroom setting, due to developers or students constantly reviewing their code in order to reduce the number of errors (see Sections 3.5.2 and 5.3.6).

Confidence - The affirmation received from a partner builds courage which as a result gives their partner the necessary confidence to tackle a problem head-on with no fear compared to when programming solo (see Sections 3.5.6 and 5.3.7).

Workload - It is crucial for the partners to follow the pair programming principles and guidelines in order to effectively divide the workload equally by frequently changing roles and remaining in constant communication with each other. Pair programming significantly reduces the amount of time spent grading each individual assignment (see Sections 3.5.8 and 5.3.9).

None - The respondent saw no disadvantage to pair programming, and most likely implemented and used pair programming correctly according to the set pair programming principles and guidelines (see Sections 3.3.5, 4.3.4 and 5.3.8).

The following disadvantages of pair programming emerged from the qualitative data:

Conflict - The use and implementation of pair programming can inadvertently lead to conflict between two people. Conflicts and/or disagreements always come as a surprise and cause upset to a person because you do not expect to get into a disagreement with a friend, colleague or someone with whom you do not share interests (see Sections 5.4.1).

Time-consuming - Pair programming with a partner at a lower skill level consumes more time compared to programming alone. As a result, this timewasting can completely discourage a person from using pair programming (see Sections 3.6.4 and 5.4.2).

Incompatible personalities - Students who have a negative pair programming experience with a partner with whom they are incompatible can cause them to detest working in a pair (see Sections 3.6.3 and 5.4.3).

Skill incompatibility - Students face challenges when they are programming in a pair and this can drastically slow down the programming process due to incompatible personalities and differing programming skill levels. They may also feel that pair programming is useless because they are unable to learn anything from the novice due to the large skill gap between them. The preference for experienced programmers to work alone or to do all the work by themselves is because of their perception that their partner is holding them back which causes frustration, guilt and mistrust (see Sections 3.6.2 and 5.4.4).

Lack of knowledge and understanding - It is challenging for someone with a high programming skill level and vast experience to work with a person who is completely clueless about programming concepts and who is unprepared to solve a problem in order to complete a task. This lack of knowledge can be time-consuming with regard to the entire development process (see Sections 5.4.5).

Absence/Tardiness - Tardiness in a partner results in a great disadvantage because one member of the team is then forced to work individually which defeats the whole purpose of pair programming and prohibits them from reaping the benefits of pair programming (see Sections 3.6.4 and 5.4.6).

Communication/Language barrier - The inability of two people to communicate with each other can lead to conflict and deter both students from making use of pair programming in the future (see Sections 3.6.1 and 5.4.7).

Anxiety - In the classroom setting, it can be challenging for someone who has social anxiety to work productively with a partner due to his/her crippling fear of being judged or watched by others which can really make the driver-navigator experience quite unpleasant (see Sections 5.4.8).

The researcher found contradictory conclusions from the quantitative, as well as the qualitative strands. These contradicting findings were of great value to the analysis of these results in the sense that they enriched the understanding of the phenomena in question.

The students who participated in this study were all placed in categories, based on their gender, ethnicity, computer and Internet access and IT as a high school subject.

With regard to differences based on gender, the female students were more inclined to make use of pair programming and make it a habit in contrast to the male students.

The students of this study were given the liberty to choose their partner and more often than not shared the same ethnic origin which indicated that compatibility in terms of ethnic origin was not a major problem.

By comparing the preferences of solo and pair programming, the researcher found that the students who had no access to a computer since Grade 1 felt somewhat neutral with regard to their programming preference. Correspondingly, the students who did have access to a computer also felt rather neutral with regard to their programming preference.

When comparing the Facilitating Conditions between students who had access to the Internet and those who did not have access to the internet. The researcher found that the students with no access to the Internet believe to a great extent that they have the technical infrastructure at their disposal to support the use of pair programming in an introductory programming course.

The researcher compared and found a significant difference between the students who had IT as a subject in high school and those who did not have IT as a high school subject with regard to their programming preferences. A positive stance emerged from the Non-IT students and it possibly had a direct effect on Behavioural Intention. It was also interesting to see that this same group of students believed to a high degree that using pair programming helped them to better their academic performances and as a result enhanced the quality of their work. This indicated that the students who did not have IT as a high school subject preferred pair programming over solo programming due to the benefits of pair programming.

The results from the ANOVA analysis showed that in general, White Males do indeed have a strong preference for solo programming over pair programming. When taking into account all the aforementioned findings, the entire group of programming students had a positive opinion on pair programming which meant that the White Males lowered the mean which, as a result, indicates that the ethnic minorities and females feel even more positive about pair programming. This significant finding therefore answered the research question and substantiated the great impact

that pair programming has on the underrepresented students' experiences and perceptions of Information Technology in South Africa. Pair programming may consequently attract and increase the retention rate of underrepresented groups in IT and encourage them to further pursue IT careers in South Africa.

6.3 STUDY CONTRIBUTIONS

In this study, the gap that exists in the literature with regard to minorities and software development, especially in South Africa was identified. The study made a contribution to the literature by proposing pair programming as a different approach to the learning experience of programming that may encourage underrepresented groups to participate in IT-related degrees and pursue ICT careers. The valuable information acquired from this study's results has indeed helped in gaining a better understanding of the underrepresented groups' perceptions of IT in South Africa. With this newly found insight, it is possible to develop strategies to confront the dominant stereotype that ICT career paths are only for white males.

Representation is important in the sense that it can change the underrepresented groups' entire perception of IT courses by seeing people that look like them pursuing careers in STEM. The positive influence may change their attitudes towards STEM which hopefully will provide them with the necessary confidence and motivation to enrol and complete their studies in Information Technology.

6.4 STUDY LIMITATIONS

This study used the mixed-methods approach and a set of open-ended questions in the questionnaire. Some of the students in this study did not provide sufficient information with regards to the open-ended set of questions. Perhaps in the future in addition to the use of the mixed-methods approach the researcher could gather more detailed information by means of interviews. This would greatly benefit future research in order to gain a better understanding of the perceptions of underrepresented groups in IT in South Africa. The generalisation of the findings from this study is limited in the sense that it focused and gathered information only from one campus of a university in South Africa.

6.5 CONCLUSION AND RECOMMENDATIONS

It can be concluded that overall, the majority of the students had a positive experience with pair programming in the introductory programming course, based on the findings from the empirical study. In accordance with the results of the seminal studies, pair programming was found to be beneficial for students who are learning how to program. Consequently, pair programming has the potential to attract and increase the retention rate of underrepresented groups in IT. The experience of learning coding by means of pair programming also changed the underrepresented groups' perception of IT and it also opened their eyes to the importance and need for IT specialists of colour in South Africa. In addition, it also provided them with potential career paths in the ICT industry.

Pair programming improves the quality of the code produced by the students in terms of functionality and readability. The immediate help and support afforded to them allows the students to solve problems quickly and easily. This form of collaborative learning significantly increased their confidence and motivation which subsequently increased their retention rate, as well as their enjoyment of learning to program. The mutual responsibility shared between the students instils an enhanced work ethic which can be fundamental when working in the software development industry in the future.

Both quantitative and qualitative yielded results that provided a better understanding of the impact of pair programming on the students' experiences. Based on the qualitative findings of the study, the underrepresented students provided several reasons for suggesting that pair programming is favourable for female and ethnic minority students. In an educational context, when implemented correctly, not only does pair programming benefit learners, but educators as well in the sense that their time spent grading assignments is halved and it reduces the number of teaching cases that they need to deal with during a lecture.

With regard to closing the digital divide, it is clear that the relationship between underrepresented groups and technology is impacted by access and affordability. From the literature, it is evident that the availability of technology does not ensure that minorities have access to it nor does it guarantee an increase in the use of technology. It is of the utmost importance that the digital divide is closed because it can have a negative impact on education and future employment. In the long run, this can harm the country's economy. The majority of South Africa's population who

are currently the minorities in the ICT sector will become the majority of the ICT workforce in the future. However, if the digital divide is not narrowed soon, and they remain at the short end of the digital divide, the nation risks losing the largest ICT market in all of Africa. It is absolutely crucial for this nation's economic health that we reap the potential benefits associated with the ICT sector in the midst of the forthcoming Fourth Industrial Revolution.

6.5.1 RECOMMENDATIONS ARISING FROM THIS STUDY

It is evident from the problem statement of this study that in South Africa there is an increased demand for ICT specialists, especially for ICT professionals of colour. This study aimed to investigate the impact of pair programming on the experiences and perceptions of underrepresented groups in Information Technology in South Africa. The ICT skills shortage in South Africa is indeed of national concern in the sense that the majority of the population is in fact the minority when it comes to the make-up of the IT workforce in the country. Prior studies on South African ICT graduate skills requirements have provided figures relating to this great lack of basic IT skills, and the difficulty of IT companies in South Africa to find adequate candidates of colour during the recruitment process. This need for qualified Black graduates in South Africa is furthermore affected by the limited number of enrolments in IT courses at a tertiary level.

This is quite alarming because with the imminent Fourth Industrial Revolution, South Africa's ICT sector, as the continent's largest ICT market, will be a crucial contributor to the nation's Gross Domestic Product. With the rise of the Fourth Industrial Revolution, there has been a great need for individuals who are able to solve problems by using logic, critical thinking and teamwork. The South African Department of Education aims to solve this issue by giving South Africa's future IT specialists the opportunity to fulfil their potential. The Department of Education intends to introduce coding and robotics in its curricula in order to prepare for the Fourth Industrial Revolution. Pair programming has been known to be a beneficial code-learning technique in any setting, whether it be in the classroom at a primary or secondary level or at a tertiary level.

The influence of pair programming may address the underlying factors that impact the participation of underrepresented groups in IT-related fields. This study investigated the impact of pair programming on the experiences, and perceptions of these underrepresented groups in ICT in South Africa.

This study provides beneficial insight into the dominant determinants of the students' intentions to use pair programming. The results from this study can aid educators in introducing pair programming as a learning technique in their classrooms and coding workshops. Institutions at a tertiary level can use these results to address the skills shortage by incorporating pair programming in their curricula.

In order to reap the full benefits of pair programming, it is recommended that government promotes the importance of coding and robotics by including these subjects in curricula, as well as spreading awareness that Computer Science is a crucial part of the Fourth Industrial Revolution. For this reason, coding initiatives should be supported across the country. It is recommended that from an educational perspective, awareness is promoted at secondary and tertiary levels regarding the career opportunities in IT in South Africa. Companies in the ICT sector should be encouraged to form stronger relationships with tertiary institutions in order to improve the recruitment process and by doing so, to address the IT skills shortage in the country by employing qualified candidates of colour.

6.5.2 RECOMMENDATIONS FOR FURTHER RESEARCH

The researcher recommends that a similar study should be conducted with a much larger number of students. For the purpose of generalisation, a quantitative study on a sample of students of colour, using the same research questions can be very helpful. A similar comparative study could be carried out with a control group of black students who have and have not used pair programming in order to understand their experiences, as well as perceptions of computer programming. The development of a model to promote the importance of coding and robotics in less fortunate communities in South Africa could also be undertaken.

The research reported in this study will hopefully make a difference to the number of people of colour and women who enrol in Computer Science courses and increase their retention rate and also motivate them to pursue careers in ICT. Hopefully this will encourage the use of pair programming as a learning technique in order to benefit everyone and South Africa as well.

BIBLIOGRAPHY

Aguirre-Urreta, M., Marakas, G. & Sun, W. 2015. Effectiveness of Pair and Solo Programming Methods: A Survey and an Analytical Approach.

Aish, N., Asare, P. & Miskioğlu, E.E. 2018. People like me: Providing relatable and realistic role models for underrepresented minorities in STEM to increase their motivation and likelihood of success. (In. 2018 IEEE Integrated STEM Education Conference (ISEC) organised by: IEEE. p. 83-89).

Akala, B. & Divala, J. 2016. Gender equity tensions in South Africa's post-apartheid higher education: in defence of differentiation. *South African Journal of Higher Education*, 30(1):1-16.

Alharahsheh, H. & Pius, A. 2020. A Review of key paradigms: positivism VS interpretivism. *Global Academic Journal of Humanities and Social Sciences*, 2(3):39-43.

Aliyu, A.A., Bello, M.U., Kasim, R. & Martin, D. 2014. Positivist and non-positivist paradigm in social science research: Conflicting paradigms or perfect partners. *J. Mgmt. & Sustainability*, 4:79.

Ally, M., Darroch, F. & Toleman, M. 2005. A framework for understanding the factors influencing pair programming success. (In. International Conference on Extreme Programming and Agile Processes in Software Engineering organised by: Springer. p. 82-91).

Ameri, A., Khajouei, R., Ameri, A. & Jahani, Y. 2020. Acceptance of a mobile-based educational application (LabSafety) by pharmacy students: An application of the UTAUT2 model. *Education and Information Technologies*, 25(1):419-435.

Anwer, F., Aftab, S. & Ali, I. 2017a. Proposal of Tailored Extreme Programming Model for Small Projects. *International Journal of Computer Applications*, 171(7):23-27.

Anwer, F., Aftab, S., Shah, S.M. & Waheed, U. 2017b. Comparative Analysis of Two Popular Agile Process Models: Extreme Programming and Scrum. *International Journal of Computer Science and Telecommunications*, 8(2):1-7.

Arora, S.K. 2021. What is Data Analysis? Methods, Techniques & Tools. <https://hackr.io/blog/what-is-data-analysis-methods-techniques-tools> Date of access: 15 Oct. 2021.

Bailey, R. & Mentz, E. 2017. The value of pair programming in the IT classroom. *The Independent Journal of Teaching and Learning*, 12(1):90-103.

- Bantham, B. 2017. NWU academic presents university history at Africa Day symposium. <https://news.nwu.ac.za/nwu-academic-presents-university-history-africa-day-symposium> Date of access: 10 Feb. 2022.
- Banu, R. & Duraipandian, K. 2014. Development of an instrument to measure work life balance of it professionals in Chennai. *International Journal of management*, 5(11):21-33.
- Baser, M. 2013. Attitude, gender and achievement in computer programming. *Middle East Journal of Scientific Research*, 14(2):248-255.
- Bazeley, P. 2018. Mixed methods in my bones”: transcending the qualitative-quantitative divide. *International Journal of Multiple Research Approaches*, 10(1):334-341.
- Beck, K. 1999. Embracing Change With Extreme Programming *Computer*, 32(10):70-77.
- Beck, K. 2000. Extreme programming explained: embrace change: addison-wesley professional.
- Begel, A. & Nagappan, N. 2008. Pair programming: what's in it for me? (In. Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement organised by: ACM. p. 120-128).
- Benadé, T. & Liebenberg, J. 2017. Pair Programming as a Learning Method Beyond the Context of Programming. (In. Proceedings of the 6th Computer Science Education Research Conference organised by: ACM. p. 48-55).
- Bernadita, C. 2021. Your Modern Business Guide To Data Analysis Methods And Techniques. <https://www.datapine.com/blog/data-analysis-methods-and-techniques/> Date of access: 12 Oct. 2021.
- Bevan, J., Werner, L. & McDowell, C. 2002. Guidelines for the use of pair programming in a freshman programming class. (In. Proceedings 15th Conference on Software Engineering Education and Training (CSEE&T 2002) organised by: IEEE. p. 100-107).
- Bock, S.J., Taylor, L.J., Phillips, Z.E. & Sun, W. 2013. Women and minorities in computer science majors: Results on barriers from interviews and a survey. *Issues in Information Systems*, 14(1):143-152.
- Boone, M., Hendricks, M.L. & Waller, R. 2014. Closing the digital divide and its impact on minorities. *Global E Learning*, 3:1-6.
- Bosch, N., Huang, E., Angrave, L. & Perry, M. 2019. Modeling Improvement for Underrepresented Minorities in Online STEM Education. Paper presented at the Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization, Larnaca, Cyprus. <https://doi.org/10.1145/3320435.3320463> Date of access: 18 Mar. 2019.

- Brought, G., Eby, L.M. & Wahls, T. 2008. The effects of pair-programming on individual programming skill. (In. Proceedings of the 39th SIGCSE technical symposium on Computer science education organised by. p. 200-204).
- Brought, G., Wahls, T. & Eby, L.M. 2011. The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education (TOCE)*, 11(1):2.
- Brown, K. 2006. "New" educational injustices in the "new" South Africa. *Journal of Educational Administration*.
- Bryman, A. 2006. Integrating quantitative and qualitative research: how is it done? *Qualitative research*, 6(1):97-113.
- Burrell, D.N. 2019. Developing more women in managerial roles in information technology and cybersecurity.
- BusinessTech. 2019a. South African schools will soon get these 3 new subjects. <https://businesstech.co.za/news/government/304684/south-african-schools-will-soon-get-these-3-new-subjects> Date of access: 18 Mar. 2019.
- BusinessTech. 2019b. These are the most in-demand job skills in South Africa right now. <https://businesstech.co.za/news/business/305606/these-are-the-most-in-demand-job-skills-in-south-africa-right-now-6/> Date of access: 18 Mar. 2019.
- BusinessTech. 2021. Big change planned for schools in South Africa. <https://businesstech.co.za/news/government/528056/big-change-planned-for-schools-in-south-africa/> Date of access: 12 Oct. 2021.
- Buzzetto-More, N.A., Ukoha, O. & Rustagi, N. 2010. Unlocking the barriers to women and minorities in computer science and information systems studies: Results from a multi-methodological study conducted at two minority serving institutions. *Journal of Information Technology Education: Research*, 9(1):115-131.
- Calitz, A.P., Greyling, J.H. & Cullen, M.D. 2014. South African industry ICT graduate skills requirements. *Southern African Computer Lecturers' Association (SACLA)*, 1:25-26.
- Campe, S., Denner, J., Green, E. & Werner, L. 2018. Pair Programming Interactions in Middle School: Collaborative, Constructive, Dismissive, or Disengaged? (In. Proceedings of the 49th ACM Technical Symposium on Computer Science Education organised by: ACM. p. 1093-1093).
- Campos-Castillo, C. 2015. Revisiting the first-level digital divide in the United States: Gender and race/ethnicity patterns, 2007–2012. *Social Science Computer Review*, 33(4):423-439.

Castagna, R. 2021. Information Technology.
<https://searchdatacenter.techtarget.com/definition/IT> Dare of Access: 25 Nov. 2021.

Celepku, M. & Boyer, K.E. 2018. Thematic analysis of students' reflections on pair programming in cs1. (In. Proceedings of the 49th ACM Technical Symposium on Computer Science Education organised by: ACM. p. 771-776).

Chaparro, E.A., Yuksel, A., Romero, P. & Bryant, S. 2005. Factors Affecting the Perceived Effectiveness of Pair Programming in Higher Education. (In. PPIG organised by: Citeseer. p. 2).

Choi, K.S., Deek, F.P. & Im, I. 2009. Pair dynamics in team collaboration. *Computers in Human Behavior*, 25(4):844-852.

Chong, J. & Siino, R. 2006. Interruptions on software teams: a comparison of paired and solo programmers. (In. Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work organised by: ACM. p. 29-38).

Chutel, L. 2016. South Africa doesn't have enough developers to build a digital economy. *Quartz Africa*.

Chutel, L. 2017. Post-apartheid South Africa is failing the very people it liberated. *Quartz Africa*.

Clark, V.L.P. & Ivankova, N.V. 2015. Mixed methods research: A guide to the field. Vol. 3: Sage publications.

Cockburn, A. & Williams, L. 2000. The costs and benefits of pair programming. *Extreme programming examined*, 8:223-247.

Coger, R.N., Cuny, J., Klawe, M., McGann, M. & Purcell, K.D. 2012. Why STEM fields still don't draw more women. *Chronicle of Higher Education*, 59(10):B24-B27.

Cohen, L., Manion, L. & Morrison, K. 2013. Research methods in education: Routledge.

Conrad, C., Serlin, R. & Harwell, M. 2014. Research Design in Qualitative Quantitative/Mixed Methods. The SAGE Handbook for Research in Education: Pursuing Ideas as the Keystone of Exemplary Inquiry. p. 147.

Creamer, E.G. 2018. Enlarging the conceptualization of mixed method approaches to grounded theory with intervention research. *American Behavioral Scientist*, 62(7):919-934.

Creswell, J.W. 2013. Steps in conducting a scholarly mixed methods study.

Creswell, J.W. & Creswell, J.D. 2017. Research design: Qualitative, quantitative, and mixed methods approaches: Sage publications.

Creswell, J.W. & Hirose, M. 2019. Mixed methods and survey research in family medicine and community health. *Family Medicine and Community Health*, 7(2).

Creswell, J.W., Klassen, A.C., Plano Clark, V.L. & Smith, K.C. 2011. Best practices for mixed methods research in the health sciences. *Bethesda (Maryland): National Institutes of Health*, 2013:541-545.

Creswell, J.W. & Plano Clark, V. 2018. Designing and conducting mixed methods research. Third: SAGE Publications, Inc.

Creswell, J.W. & Tashakkori, A. 2007. Differing perspectives on mixed methods research: Sage Publications Sage CA: Los Angeles, CA.

Dalalah, A. 2014. Extreme Programming: Strengths and Weaknesses. *Computer Technology and Application*, 5(1).

Dalvit, L. & De Klerk, V. 2005. Attitudes of Xhosa-speaking students at the University of Fort Hare towards the use of Xhosa as a language of learning and teaching (LOLT). *Southern African Linguistics and Applied Language Studies*, 23(1):1-18.

de Lima Salge, C.A. & Berente, N. 2016. Pair Programming vs. Solo Programming: What Do We Know After 15 Years of Research? (In. 2016 49th Hawaii International Conference on System Sciences (HICSS) organised by: IEEE. p. 5398-5406).

Demir, Ö. & Seferoglu, S.S. 2021. A Comparison of solo and pair programming in terms of flow experience, coding quality, and coding achievement. *Journal of Educational Computing Research*, 58(8):1448-1466.

Demirşahin, I., Yalçınkaya, İ. & Zeyrek, D. 2012. Pair annotation: Adaption of pair programming to corpus annotation. (In. Proceedings of the sixth linguistic annotation workshop organised by: Association for Computational Linguistics. p. 31-39).

Denner, J., Werner, L., Campe, S. & Ortiz, E. 2014. Pair programming: Under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education*, 46(3):277-296.

Department of Higher Education & Training South Africa. 2014. White paper for post-school education and training: Building an expanded, effective and integrated post-school system.

Dewi, F.R., Ekohariadi, E., Soeryanto, S. & Rijanto, T.R.T. 2021. The Effect of Pair Programming Learning Model on Learning Outcomes of Basic Programming of Vocational School Students. *International Journal for Educational and Vocational Studies*, 3(3):236-242.

Dhingra, M. & Dhingra, V. 2011. Perception: Scriptures' Perspective. *Journal of Human Values*, 17(1):63-72.

Dhir, S., Kumar, D. & Singh, V. 2019. Success and failure factors that impact on project implementation using agile software development methodology. *Software engineering*. Springer. p. 647-654).

Di Bella, E., Fronza, I., Phaphoom, N., Sillitti, A., Succi, G. & Vlasenko, J. 2012. Pair Programming and Software Defects--A Large, Industrial Case Study. *IEEE Transactions on Software Engineering*, 39(7):930-953.

Dickey, M.R. 2017. Dispatches on diversity: Uber, sexual harassment and venture capital.

Dingsøyr, T., Nerur, S., Balijepally, V. & Moe, N.B. 2012. A decade of agile methodologies: Towards explaining agile software development (Vol. 85. pp. 1213-1221): Elsevier.

Dou, W.-f., Bai, X.-d., Zhang, Y., Yang, Y.-b. & Yang, Y. 2017. Implementation and Practice of Pair Learning Based on Pair Programming. *DEStech Transactions on Social Science, Education and Human Science*(meit).

El-Masri, M. & Tarhini, A. 2017. Factors affecting the adoption of e-learning systems in Qatar and USA: Extending the Unified Theory of Acceptance and Use of Technology 2 (UTAUT2). *Educational Technology Research and Development*, 65(3):743-763.

English, A. 2002. Extreme programming, it's worth a look. *IT professional*, 4(3):48-50.

Erickson, J., Lyytinen, K. & Siau, K. 2005. Agile modeling, agile software development, and extreme programming: the state of research. *Journal of Database Management (JDM)*, 16(4):88-100.

Ferreira, O. 2006. Die Trotse en Soms Omstrede Geskiedenis van die PUK" In U Lig": Die PU vir CHO van selfstandigwording tot samesmelting, 1951-2004, Elize S. van Eeden (redakteur): book review. *Historia*, 51(2):242-245.

Fetters, M.D. & Freshwater, D. 2015. The 1+ 1= 3 Integration Challenge: SAGE Publications Sage CA: Los Angeles, CA.

Fiegerman, S. 2017. Labor Department goes after big tech for discrimination. <http://money.cnn.com/2017/04/10/technology/labor-department-tech/index.html> Date of access: 27 Apr. 2020.

Field, A. 2009. Discovering statistics using SPSS: Sage publications.

Fihlani, P. & Dominic, B. 2019. South Africa elections: Charting divides 25 years after apartheid. <https://www.bbc.com/news/world-africa-48050197> Date of access: 15 Oct. 2021.

Filstead, W.J. 1979. Qualitative methods: A needed perspective in evaluation research. *Qualitative and quantitative methods in evaluation research*, 1.

Freeman, S., Eddy, S.L., McDonough, M., Smith, M.K., Okoroafor, N., Jordt, H. & Wenderoth, M.P. 2014. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, 111(23):8410-8415.

Gallis, H., Arisholm, E. & Dybå, T. 2003. An Initial Framework for Research on Pair Programming. Paper presented at the Proceedings of the 2003 International Symposium on Empirical Software Engineering.

Ghobadi, S., Campbell, J. & Clegg, S. 2017. Pair programming teams and high-quality knowledge sharing: A comparative study of cooperative reward structures. *Information Systems Frontiers*, 19(2):397-409.

Gómez, O.S., Batún, J.L. & Aguilar, R.A. 2013. Pair versus Solo Programming--An Experience Report from a Course on Design of Experiments in Software Engineering. *arXiv preprint arXiv:1306.4245*.

Gool, K. 2018a. Coding: An essential skill of the future. <https://www.itweb.co.za/content/mQwkoM6K36eq3r9A> Date of access: 12 Aug 2019.

Gool, K. 2018b. SA schools need to embrace coding. <https://www.itweb.co.za/content/KPNG8v8dod6v4mwD> Date of access: 12 Aug. 2019.

Govender, R.G. & Govender, D.W. 2021. A Physical Computing Approach to the Introduction of Computer Programming among a Group of Pre-service Teachers. *African Journal of Research in Mathematics, Science and Technology Education*, 25(1):54-65.

Granikov, V., Hong, Q.N., Crist, E. & Pluye, P. 2020. Mixed methods research in library and information science: A methodological review. *Library & Information Science Research*, 42(1):101003.

Gringeri, C., Barusch, A. & Cambron, C. 2013. Examining foundations of qualitative research: A review of social work dissertations, 2008–2010. *Journal of Social Work Education*, 49(4):760-773.

Grover, S., Bienkowski, M., Tamrakar, A., Siddiquie, B., Salter, D. & Divakaran, A. 2016. Multimodal analytics to study collaborative problem solving in pair programming. (In. Proceedings of the Sixth International Conference on Learning Analytics & Knowledge organised by: ACM. p. 516-517).

Guba, E.G. & Lincoln, Y.S. 1994. Competing paradigms in qualitative research. *Handbook of qualitative research*, 2(163-194):105.

- Hair Jr, J.F., Page, M. & Brunsveld, N. 2019. Essentials of business research methods: Routledge.
- Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L. & Zander, C. 2011. Pair programming in education: a literature review. *Computer Science Education*, 21(2):135-173.
- Hannay, J.E., Arisholm, E., Engvik, H. & Sjoberg, D.I. 2009. Effects of personality on pair programming. *IEEE Transactions on Software Engineering*, 36(1):61-80.
- Harloe, M. 2008. The people's home?: social rented housing in Europe and America: John Wiley & Sons.
- Harrison, F. & Stokes, M. 2017. Towards the Synthesis of Extreme Programming. *Software engineering and CS Journal*, 12(1).
- He, X. & Chen, Y. 2015. Analyzing the efficiency of pair programming in education.
- Hodgetts, P. 2004. Refactoring the development process: Experiences with the incremental adoption of agile practices. (In. Agile Development Conference organised by: IEEE. p. 106-113).
- Hong, Q.N., Rees, R., Sutcliffe, K. & Thomas, J. 2020. Variations of mixed methods reviews approaches: A case study. *Research Synthesis Methods*.
- Huchzermeyer, M. 2003. Housing rights in South Africa: Invasions, evictions, the media, and the courts in the cases of Grootboom, Alexandra, and Bredell. (In. Urban Forum organised by: Springer. p. 80-107).
- Hurlbut, W.B. 2018. Overcoming poverty and inequality in South Africa: An assessment of drivers, constraints and opportunities: World Bank.
- Igwenagu, C. 2016. Fundamentals of research methodology and data collection: LAP LAMBERT Academic Publishing.
- International Trade Administration, U.S. 2021. South Africa - Country Commerical Guide. <https://www.trade.gov/knowledge-product/south-africa-information-technology> Date of access: 12 Oct. 2021.
- Inventado, P.S., Scupelli, P., Heffernan, C. & Heffernan, N. 2017. Feedback design patterns for math online learning systems. Paper presented at the Proceedings of the 22nd European Conference on Pattern Languages of Programs.
- Iqbal, S. & Harsh, O.K. 2013. A self review and external review model for teaching and assessing novice programmers. *International Journal of Information and Education Technology*, 3(2):120.

- Isong, B., Moemi, T., Dladlu, N., Motlhabane, N., Ifeoma, O. & Gasela, N. 2016. Empirical confirmation of pair programming effectiveness in the teaching of computer programming. (In. 2016 International Conference on Computational Science and Computational Intelligence (CSCI) organised by: IEEE. p. 276-281).
- Izuagbe, R. 2021. Faculty research performance expectancy of online databases: system design characteristics as facilitating conditions. *The Journal of Academic Librarianship*, 47(2):102318.
- Jansen, D. & Warren, K. 2020. Quantitative Data Analysis 101: The lingo, methods and techniques, explained simply: GradCoach.
- Johnson, R.B., Onwuegbuzie, A.J. & Turner, L.A. 2007. Toward a definition of mixed methods research. *Journal of mixed methods research*, 1(2):112-133.
- Jones, D.L. & Fleming, S.D. 2013. What use is a backseat driver? A qualitative investigation of pair programming. (In. 2013 IEEE Symposium on Visual Languages and Human Centric Computing organised by: IEEE. p. 103-110).
- Jooste, J.P. 1957. Die geskiedenis van die PU vir CHO (tot Inkorporasie). *Koers-Bulletin for Christian Scholarship*, 25(1):63-110.
- Joshi, A., Kale, S., Chandel, S. & Pal, D.K. 2015. Likert scale: Explored and explained. *Current Journal of Applied Science and Technology*:396-403.
- Kaila, E., Rajala, T., Laakso, M.-J., Lindén, R., Kurvinen, E., Karavirta, V. & Salakoski, T. 2015. Comparing student performance between traditional and technologically enhanced programming course. *ACE*, 160:147-154.
- Kallaway, P. 1984. Apartheid and education: The education of black South Africans: Ravan Press of South Africa.
- Karthikeyan, K., Ahmed, I. & Jayalakshmi, J. 2018. Pair programming for software engineering education: an empirical study. *Int. Arab J. Inf. Technol.*, 15(2):246-255.
- Kavitha, R. & Ahmed, M.I. 2015. Knowledge sharing through pair programming in learning environments: An empirical study. *Education and Information Technologies*, 20(2):319-333.
- Kenton, W. 2021. Analysis of Variance (ANOVA).
- Khomokhoana, P.J. & Nel, L. 2020. Decoding the underlying cognitive processes and related support strategies utilised by expert instructors during source code comprehension. *South African Computer Journal*, 32(2):232-257.
- Kline, P. 2014. An easy guide to factor analysis: Routledge.

Krosnick, J.A. 2018. Questionnaire design. The Palgrave handbook of survey research. Springer. p. 439-455).

Kuleshov, Y.A., Rada, E. & Lucietto, A.M. 2021. Minority Graduates in Engineering Technology: Trends in Choice of Major. *Minority Graduates in Engineering Technology: Trends in Choice of Major*.

Lakens, D. 2013. Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and ANOVAs. *Frontiers in psychology*, 4:863.

Layman, L. 2006. Changing students' perceptions: an analysis of the supplementary benefits of collaborative software development. (In. 19th Conference on Software Engineering Education & Training (CSEET'06) organised by: IEEE. p. 159-166).

Lee, R. & Ahlswede, T. 2007. Evaluating the usefulness of pair programming in a classroom setting. (In. 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007) organised by: IEEE. p. 302-308).

Leech, N.L., Dellinger, A.B., Brannagan, K.B. & Tanaka, H. 2010. Evaluating mixed research studies: A mixed methods approach. *Journal of mixed methods research*, 4(1):17-31.

Leung, L. 2014. Availability, access and affordability across' digital divides': Common experiences amongst minority groups. *Journal of Telecommunications and the Digital Economy*, 2(2):38.

Levine, D.B. & Walker, H.M. 2006. XP practices applied to grading. Paper presented at the Proceedings of the 37th SIGCSE technical symposium on Computer science education, Houston, Texas, USA. <https://doi.org/10.1145/1121341.1121396> Date of access: 12 Aug. 2019.

Lewis, C.M. 2011. Is pair programming more effective than other forms of collaboration for young students? *Computer Science Education*, 21(2):105-134.

Lewis, C.M. & Shah, N. 2015. How equity and inequity can emerge in pair programming. (In. Proceedings of the eleventh annual International Conference on International Computing Education Research organised by: ACM. p. 41-50).

Liccardo, S. & Bradbury, J. 2017. Black women scientists: Outliers in South African universities. *African Journal of Research in Mathematics, Science and Technology Education*, 21(3):282-292.

Liebenberg, J., Benade, T. & Ellis, S. 2018. Acceptance of ICT: applicability of the Unified Theory of Acceptance and Use of Technology (UTAUT) to South African students. *The African Journal of Information Systems*, 10(3):1.

- Liebenberg, J., Mentz, E. & Breed, B. 2012. Pair programming and secondary school girls' enjoyment of programming and the subject Information Technology (IT). *Computer Science Education*, 22(3):219-236.
- Mabokela, R.O. 2000. 'We cannot find qualified blacks': faculty diversification programmes at South African universities. *Comparative Education*, 36(1):95-112.
- Mafilika, T. 2021. The most in-demand careers in South Africa right now. <https://www.iol.co.za/education/the-most-in-demand-careers-in-south-africa-right-now-e7bc533b-39d5-4012-b978-fd6ea3040c78> Date of access: 25 Nov. 2021.
- Makombe, G. 2017. An expose of the relationship between paradigm, method and design in research. *The Qualitative Report*, 22(12):3363.
- Makura, A., Skead, M. & Nhundu, K. 2011. Academic development practices at Fort Hare University: An epitome of university access. *Research in Higher Education Journal*, 12:1.
- Malik, S.I. & Coldwell-Neilson, J. 2017. A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, 22(3):1089-1120.
- Malik, S.I., Shakir, M., Eldow, A. & Ashfaq, M.W. 2019. Promoting Algorithmic Thinking in an Introductory Programming Course. *International Journal of Emerging Technologies in Learning*, 14(1).
- Matharu, G.S., Mishra, A., Singh, H. & Upadhyay, P. 2015. Empirical study of agile software development methodologies: A comparative analysis. *ACM SIGSOFT Software Engineering Notes*, 40(1):1-6.
- McCrae, R.R. & John, O.P. 1992. An introduction to the five-factor model and its applications. *Journal of personality*, 60(2):175-215.
- McCrudden, M.T. & Marchand, G. 2020. Multilevel mixed methods research and educational psychology. *Educational Psychologist*, 55(4):197-207.
- McDowell, C., Werner, L., Bullock, H. & Fernald, J. 2002. The effects of pair-programming on performance in an introductory programming course. *ACM SIGCSE Bulletin*, 34(1):38-42.
- McDowell, C., Werner, L., Bullock, H.E. & Fernald, J. 2003. The impact of pair programming on student performance, perception and persistence. (In. *Software Engineering*, 2003. Proceedings. 25th International Conference on organised by: IEEE. p. 602-607).
- McGee, K. 2018. The influence of gender, and race/ethnicity on advancement in information technology (IT). *Information and Organization*, 28(1):1-36.

- McKane, J. 2021. Great news for IT job seekers in South Africa. <https://mybroadband.co.za/news/business/389948-great-news-for-it-job-seekers-in-south-africa.html> Date of access: 04 Apr. 2021.
- Merriam-Webster. 2021. Online dictionary. <https://www.merriam-webster.com/> Date of access: 12 Oct. 2021.
- Mossberger, K., Tolbert, C.J. & Gilbert, M. 2006. Race, place, and information technology. *Urban Affairs Review*, 41(5):583-620.
- Mossberger, K., Tolbert, C.J. & Stansbury, M. 2003. Virtual inequality: Beyond the digital divide: Georgetown University Press.
- Muller, J. 1991. South Africa, in: P.G. Altbach (Ed.). *International Higher Education an Encyclopedia*, 1:411-123.
- Murray, B.K. 1990. Wits as an 'open' university 1939–1959: black admissions to the university of the Witwatersrand. *Journal of Southern African Studies*, 16(4):649-676.
- MyBroadband. 2014. Facebook user numbers for South Africa. <https://mybroadband.co.za/news/internet/113277-facebook-user-numbers-for-south-africa-revealed.html> Date of access: 12 Aug. 2019.
- Mzekandaba, S. 2016. SA ranks bottom in global maths, science scores. <https://www.itweb.co.za/content/z5yONPvEgLQMXWrb> Date of access: 07 Oct. 2019.
- Mzekandaba, S. 2020. Upswing expected in SA's ICT spending in 2020. <https://www.itweb.co.za/content/nWJadMb83oGvbJO1> Date of access: 12 Oct. 2021.
- Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C. & Balik, S. 2003. Improving the CS1 experience with pair programming. *ACM SIGCSE Bulletin*, 35(1):359-362.
- Nassaji, H. 2021. Effect sizes in quantitative and qualitative research: SAGE Publications Sage UK: London, England.
- National Telecommunications & Administration, I. 2013. Exploring the digital nation: America's emerging online experience: US Department of Commerce Washington, DC.
- Nawrocki, J. & Wojciechowski, A. 2001. Experimental evaluation of pair programming. *European Software Control and Metrics (Escom)*:99-101.
- Nemoto, T. & Beglar, D. 2014. Likert-scale questionnaires. (In. JALT 2013 Conference Proceedings organised by. p. 1-8).

Nicolau, E.A. & Seymour, L.F. 2017. Mobile and Game Usage, Gender and Attitude Towards Computing Degrees. (In. Annual Conference of the Southern African Computer Lecturers' Association organised by: Springer. p. 67-82).

Nikolopoulou, K., Gialamas, V. & Lavidas, K. 2020. Acceptance of mobile phone by University students for their studies: An investigation applying UTAUT2 model. *Education and Information Technologies*:1-17.

North, D., Johnston, K. & Ophoff, J. 2014. The use of mobile phones by South African university students. *Issues in informing science and information technology*, 11:115-138.

NWU. 2022a. History of the NWU. <https://www.nwu.ac.za/welcome> Date of access: 10 Feb. 2022.

NWU. 2022b. NWU @ a glance. <https://www.nwu.ac.za/content/nwu-glance> Date of access: 10 Feb. 2022.

O'Donnell, C., Buckley, J., Mahdi, A., Nelson, J. & English, M. 2015. Evaluating Pair-Programming for Non-Computer Science Major Students. (In. Proceedings of the 46th ACM Technical Symposium on Computer Science Education organised by: ACM. p. 569-574).

Oblinger, D. 2003. Boomers gen-xers millennials. *EDUCAUSE review*, 500(4):37-47.

Ofosu, W.K. & Asgill, A.B. 2003. Impact of the digital divide on low-income and minority students. (In. ASEE Annual Conference Proceedings organised by. p. 6685-6691).

Papastergiou, M. 2008. Are computer science and information technology still masculine fields? High school students' perceptions and career choices. *Computers & education*, 51(2):594-608.

Parfrey, P.S. & Ravani, P. 2021. On Framing the Research Question and Choosing the Appropriate Research Design. *Clinical Epidemiology*. Springer. p. 1-16).

Parikh, D.V.S. 2019. EXTREME PROGRAMMING: VALUES, PRINCIPLES, PRACTICES AND USE. *Journal Current Science*, 20(1).

Park, Y.S., Konge, L. & Artino Jr, A.R. 2020. The positivism paradigm of research. *Academic Medicine*, 95(5):690-694.

Parry, D.A. & le Roux, D.B. 2018. Off-task media use in lectures: Towards a theory of determinants. (In. Annual Conference of the Southern African Computer Lecturers' Association organised by: Springer. p. 49-64).

PayScale.com. 2018. Average Software Engineer / Developer / Programmer Salary. https://www.payscale.com/research/ZA/Job=Software_Engineer_%2F_Developer_%2F_Programmer/Salary Date of access: 06 Aug. 2018.

- Pedrycz, W., Russo, B. & Succi, G. 2011. A model of job satisfaction for collaborative development processes. *Journal of Systems and Software*, 84(5):739-752.
- Pieterse, V., Thompson, L., Marshall, L. & Venter, D.M. 2012. Participation patterns in student teams. (In. Proceedings of the 43rd ACM technical symposium on Computer Science Education organised by. p. 265-270).
- Pinsonneault, A. & Kraemer, K. 1993. Survey research methodology in management information systems: an assessment. *Journal of management information systems*, 10(2):75-105.
- Plonka, L., Sharp, H., Van der Linden, J. & Dittrich, Y. 2015. Knowledge transfer in pair programming: An in-depth analysis. *International journal of human-computer studies*, 73:66-78.
- Pluye, P. 2012. Méthodes mixtes en santé mondiale. JC Suárez-Herrera et MJ Blain, *La recherche en santé mondiale: Perspectives socio-anthropologiques*, Québec, Association francophone pour le savoir:129-135.
- Poff, M.A. 2003. Pair programming to facilitate the training of newly-hired programmers.
- Ponterotto, J.G. 2005. Qualitative research in counseling psychology: A primer on research paradigms and philosophy of science. *Journal of counseling psychology*, 52(2):126.
- Poonam, R. & Yasser, C.M. 2018. An experimental study to investigate personality traits on pair programming efficiency in extreme programming. (In. 2018 5th International Conference on Industrial Engineering and Applications (ICIEA) organised by: IEEE. p. 95-99).
- Pretorius, H.W., Mawela, T., Strydom, I., Villiers, C.d. & Johnson, R.D. 2015. Continuing the discourse of women in information technology: a South African perspective. *Gender, Technology and Development*, 19(3):346-369.
- Rahi, S. 2017. Research design and methods: A systematic review of research paradigms, sampling issues and instruments development. *International Journal of Economics & Management Sciences*, 6(2):1-5.
- Raman, A. & Don, Y. 2013. Preservice teachers' acceptance of learning management software: An application of the UTAUT2 model. *International Education Studies*, 6(7):157-164.
- Ramírez-Montoya, M.-S. & Lugo-Ocando, J. 2020. Systematic Review of Mixed Methods in the Framework of Educational Innovation. *Comunicar: Media Education Research Journal*, 28(65):9-20.
- Ramrathan, L. 2016. Beyond counting the numbers: Shifting higher education transformation into curriculum spaces. *Transformation in Higher Education*, 1(1):1-8.

Rawlins, L.K. 2015. Coding should be compulsory at schools. <https://www.itweb.co.za/content/2j5alrMQ4Oe7pYQk> Date of access: 27 Apr. 2020.

Roebuck, K. 2012. Systems Development Life Cycle (SDLC): High-impact Strategies-What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors: Emereo Publishing.

Romani, L., Barmeyer, C., Primecz, H. & Pilhofer, K. 2018. Cross-cultural management studies: state of the field in the four research paradigms. *International Studies of Management & Organization*, 48(3):247-263.

Salahuddin, M. & Gow, J. 2016. The effects of Internet usage, financial development and trade openness on economic growth in South Africa: A time series analysis. *Telematics and Informatics*, 33(4):1141-1154.

Salleh, N., Mendes, E. & Grundy, J. 2010a. Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. *IEEE Transactions on Software Engineering*, 37(4):509-525.

Salleh, N., Mendes, E. & Grundy, J. 2014. Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments. *Empirical Software Engineering*, 19(3):714-752.

Salleh, N., Mendes, E., Grundy, J. & Burch, G.S.J. 2010b. An empirical study of the effects of conscientiousness in pair programming using the five-factor personality model. (In. 2010 ACM/IEEE 32nd International Conference on Software Engineering organised by: IEEE. p. 577-586).

Sandelowski, M. 2000. Focus on research methods combining qualitative and quantitative sampling, data collection, and analysis techniques. *Research in nursing & health*, 23(3):246-255.

Sanders, D. 2002. Student perceptions of the suitability of extreme and pair programming. *Extreme programming perspectives*:168-174.

Scotland, J. 2012. Exploring the philosophical underpinnings of research: Relating ontology and epistemology to the methodology and methods of the scientific, interpretive, and critical research paradigms. *English language teaching*, 5(9):9-16.

Sehoole, M. 2013. Democratizing higher education policy: Constraints of reform in post-apartheid South Africa: Routledge.

Sfetsos, P., Stamelos, I., Angelis, L. & Deligiannis, I. 2006. Investigating the impact of personality types on communication and collaboration-viability in pair programming—an empirical study. (In. International Conference on Extreme Programming and Agile Processes in Software Engineering organised by: Springer. p. 43-52).

Sharif, A., Afshan, S. & Qureshi, M.A. 2019. Acceptance of learning management system in university students: an integrating framework of modified UTAUT2 and TTF theories. *International Journal of Technology Enhanced Learning*, 11(2):201-229.

Sharp, H. & Robinson, H. 2010. Three 'C's of agile practice: collaboration, co-ordination and communication. *Agile Software Development*. Springer. p. 61-85).

Sharpe, H., Rogers, Y. & Preece, J. 2007. *Interaction design: beyond human-computer interaction* 2nd ed: John Wiley & Sons Ltd.

Shi, J., Shah, A., Hedman, G. & O'Rourke, E. 2019. Pyrus: Designing A Collaborative Programming Game to Support Problem-Solving Behaviors.

Sillitti, A., Succi, G. & Vlasenko, J. 2012. Understanding the impact of pair programming on developers attention: a case study on a large industrial experimentation. (In. 2012 34th International Conference on Software Engineering (ICSE) organised by: IEEE. p. 1094-1101).

Simon-Lewis, A. 2017. Sexism in Silicon Valley: The lawsuits that could bring the tech giants into disrepute.

Singh, Y.K. 2006. *Fundamental of research methodology and statistics*: New Age International.

Smith, M.O., Giugliano, A. & DeOrio, A. 2017. Long Term Effects of Pair Programming. *IEEE Transactions on Education*, 61(3):187-194.

Sohaib, O., Solanki, H., Dhaliwa, N., Hussain, W. & Asif, M. 2019. Integrating design thinking into extreme programming. *Journal of Ambient Intelligence and Humanized Computing*, 10(6):2485-2492.

Sooryamoorthy, R. 2015. *Transforming science in South Africa: Development, collaboration and productivity*: Springer.

Srikanth, H., Williams, L., Wiebe, E., Miller, C. & Balik, S. 2004. On pair rotation in the computer science course. (In. 17th Conference on Software Engineering Education and Training, 2004. Proceedings. organised by: IEEE. p. 144-149).

Stapel, K., Knauss, E., Schneider, K. & Becker, M. 2010. Towards understanding communication structure in pair programming. (In. International Conference on Agile Software Development organised by: Springer. p. 117-131).

Statistics South Africa. 2019. Youth graduate unemployment rate increases in Q1: 2019. <http://www.statssa.gov.za/?p=12121> 18 Mar. 2019.

Statistics South Africa. 2021. South African labour market is more favourable to men than women. <http://www.statssa.gov.za/?p=14606> Date of access: 15 Oct. 2021.

Stoilescu, D. & McDougall, D. 2011. Gender digital divide and challenges in undergraduate computer science programs. *Canadian Journal of Education*, 34(1):308-333.

Stross, R. 2008. What has driven women out of computer science. <http://classtap.pbworks.com/f/Women%20Driven%20From%20Computer%20Science%20Field.pdf> Date of access: 18 Mar. 2019.

Talukdar, D. & Gauri, D.K. 2011. Home Internet access and usage in the USA: Trends in the socio-economic digital divide. *Communications of the Association for Information Systems*, 28(1):7.

Tapia, A. & Kvasny, L. 2004. Recruitment is never enough: Retention of women and minorities in the IT workplace. *Conference*:89-91.

Teague, M.M. 2011. Pedagogy of introductory computer programming: a people-first approach. Queensland University of Technology.

Telecommunications, N. & Administration, I. 2013. Exploring the digital nation: America's emerging online experience: US Department of Commerce Washington, DC.

The Department of Communications and Digital Technologies, S.A. 2014. The Information Technology (IT) Sector.

Thompson, C. 2019. The Secret History of Women in Coding. http://www.science.smith.edu/dftwiki/images/a/af/SecretHistoryOfWomenInCoding_NYT_02032019.pdf Date of access: 27 Apr. 2020.

Umapathy, K. & Ritzhaupt, A.D. 2017. A meta-analysis of pair-programming in computer programming courses: Implications for educational practice. *ACM Transactions on Computing Education (TOCE)*, 17(4):16.

Uzzaman, A. 2019. Ten technology trends that will impact our lives in 2020. <https://venturebeat.com/2019/12/30/10-technology-trends-that-will-impact-our-lives-in-2020/> Date of access: 27 Apr. 2020.

Vandenberg, J., Rachmatullah, A., Lynch, C., Boyer, K.E. & Wiebe, E. 2021. The Relationship of CS Attitudes, Perceptions of Collaboration, and Pair Programming Strategies on Upper Elementary Students' CS Learning. (In. Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1 organised by. p. 46-52).

Vanhanen, J. 2011. Empirical assessment of the adoption, use and effects of pair programming.

Varma, R. 2002. Women in information technology: A case study of undergraduate students in a minority-serving institution. *Bulletin of Science, Technology & Society*, 22(4):274-282.

- Varma, R. 2018. US Science and Engineering Workforce: Underrepresentation of women and minorities. *American Behavioral Scientist*, 62(5):692-697.
- Varma, R. 2020. Indian Perspective on Women in Computing. (In. 2020 Gulf Southwest Section Conference organised by.
- Venkatesh, V. 2021. Adoption and use of AI tools: a research agenda grounded in UTAUT. *Annals of Operations Research*:1-12.
- Venkatesh, V., Brown, S.A. & Bala, H. 2013. Bridging the qualitative-quantitative divide: Guidelines for conducting mixed methods research in information systems. *MIS quarterly*:21-54.
- Venkatesh, V., Morris, M.G., Davis, G.B. & Davis, F.D. 2003. User acceptance of information technology: Toward a unified view. *MIS quarterly*:425-478.
- Venkatesh, V., Thong, J.Y. & Xu, X. 2012. Consumer acceptance and use of information technology: extending the unified theory of acceptance and use of technology. *MIS quarterly*, 36(1):157-178.
- Venkatesh, V., Thong, J.Y. & Xu, X. 2016. Unified theory of acceptance and use of technology: A synthesis and the road ahead. *Journal of the Association for Information Systems*, 17(5):328-376.
- Von Hausswolff, K. 2017. Hands-on in computer programming education. (In. Proceedings of the 2017 ACM Conference on International Computing Education Research organised by: ACM. p. 279-280).
- Wang, Y. 2020. Paradigm Debates in Education: Understanding Their Strengths and Weakness. (In. 2020 4th International Seminar on Education, Management and Social Sciences (ISEMSS 2020) organised by: Atlantis Press. p. 725-729).
- Werner, L., Denner, J., Campe, S., Ortiz, E., DeLay, D., Hartl, A.C. & Laursen, B. 2013. Pair programming for middle school students: does friendship influence academic outcomes? (In. Proceeding of the 44th ACM technical symposium on Computer science education organised by: ACM. p. 421-426).
- Whitney, T. & Taylor, V. 2018. Increasing Women and Underrepresented Minorities in Computing: The Landscape and What You Can Do. *Computer*, 51(10):24-31.
- Williams, B., Onsmann, A. & Brown, T. 2010. Exploratory factor analysis: A five-step guide for novices. *Australasian journal of paramedicine*, 8(3).

- Williams, L. 2001. Integrating pair programming into a software development process. (*In*. Proceedings 14th Conference on Software Engineering Education and Training.'In search of a software engineering profession'(Cat. No. PR01059) organised by: IEEE. p. 27-36).
- Williams, L. 2010a. Agile software development methodologies and practices. *Advances in Computers*. Elsevier. p. 1-44).
- Williams, L. & Kessler, R. 2002. *Pair programming illuminated*: Addison-Wesley Longman Publishing Co., Inc.
- Williams, L. & Kessler, R. 2003. *Pair Programming Illumi—nated* Addison Wesley: Pearson.
- Williams, L., Layman, L., Osborne, J. & Katira, N. 2006. Examining the compatibility of student pair programmers. (*In*. AGILE 2006 (AGILE'06) organised by: IEEE. p. 10 pp.-420).
- Williams, L., McCrickard, D.S., Layman, L. & Hussein, K. 2008. Eleven guidelines for implementing pair programming in the classroom. (*In*. Agile 2008 Conference organised by: IEEE. p. 445-452).
- Williams, L. & Upchurch, R.L. 2001. In support of student pair-programming. *ACM SIGCSE Bulletin*, 33(1):327-331.
- Williams, L.A. 2010b. Pair Programming. *Encyclopedia of software engineering*, 2.
- Williams, L.A. & Kessler, R.R. 2000a. All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM*, 43(5):108-114.
- Williams, L.A. & Kessler, R.R. 2000b. The effects of" pair-pressure" and" pair-learning" on software engineering education. (*In*. Thirteenth conference on software engineering education and training organised by: IEEE. p. 59-65).
- Williams, L.A. & Kessler, R.R. 2001. Experiments with industry's "pair-programming" model in the computer science classroom. *Computer Science Education*, 11(1):7-20.
- Wong, L. 2008. Data analysis in qualitative research: A brief guide to using NVivo. *Malaysian family physician: the official journal of the Academy of Family Physicians of Malaysia*, 3(1):14.
- Wood, K., Parsons, D., Gasson, J. & Haden, P. 2013. It's never too early: pair programming in CS1. (*In*. Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136 organised by: Australian Computer Society, Inc. p. 13-21).
- Worldometers. 2019. South Africa population. <https://www.worldometers.info/world-population/south-africa-population/> Date of access: 24 Jul. 2019.
- Wray, S. 2009. How pair programming really works. *IEEE software*, 27(1):50-55.

Ying, K.M., Pezzullo, L.G., Ahmed, M., Crompton, K., Blanchard, J. & Boyer, K.E. 2019. In Their Own Words: Gender Differences in Student Perceptions of Pair Programming. (In. Proceedings of the 50th ACM Technical Symposium on Computer Science Education organised by: ACM. p. 1053-1059).

Yong, A.G. & Pearce, S. 2013. A beginner's guide to factor analysis: Focusing on exploratory factor analysis. *Tutorials in quantitative methods for psychology*, 9(2):79-94.

Yusuf, M.O. 2005. Information and communication technology and education: Analysing the Nigerian national policy for information technology. *International education journal*, 6(3):316-321.

Zarb, M. & Hughes, J. 2015. Breaking the communication barrier: guidelines to aid communication within pair programming. *Computer science education*, 25(2):120-151.

Zarb, M., Hughes, J. & Richards, J. 2014. Evaluating industry-inspired pair programming communication guidelines with undergraduate students.

Zhong, B., Wang, Q. & Chen, J. 2016. The impact of social factors on pair programming in a primary school. *Computers in Human Behavior*, 64:423-431.

APPENDIX A – ETHICS APPROVAL LETTER



Private Bag X1290, Potchefstroom
South Africa 2520
50676
Tel: 018 299-1111/2222
Fax: 018 299-4910
Web: <http://www.nwu.ac.za>
50676
Senate Committee for Research Ethics
Tel: 018 299-4849
Email: nkosinathi.machine@nwu.ac.za

ETHICS APPROVAL LETTER OF STUDY

Based on approval by the Faculty of Natural and Agricultural Sciences Ethics Committee (FNASREC), the Faculty of Natural and Agricultural Sciences Ethics Committee hereby approves your study as indicated below. This implies that the North-West University Senate Committee for Research Ethics (NWU-SCRE) 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: Pair programming and underrepresented groups in Information Technology in South Africa															
Study Leader/Supervisor: Dr J Liebenberg															
Student: TD Litlhakanyane															
Ethics number:	N	W	U	-	0	1	4	7	1	-	2	0	-	A	9
	Institution		Study Number				Year		Status						
Status: S = Submission; R = Re-Submission; P = Provisional Authorisation; A = Authorisation															
Application type: Single				Risk Category:				Minimal							
Commencement date: 01/02/2020															
Expiry date: 01/04/2021															
Approval of the study is initially provided for a year, after which continuation of the study is dependent on receipt and review of the annual (or as otherwise stipulated) monitoring report and the concomitant issuing of a letter of continuation.															

Special in process conditions of the research for approval (if applicable):

- The following documentation are archived by FNASREC and should be complete and kept up to date:
 - Research proposal
 - Signed approval from the scientific committee indicating the proposed risk category
- All researchers involved in the study should submit signed NWU code of conduct statements annually.
- All researchers of low risk studies should submit proof of relevant ethics training every two years.
- All researchers that take part in activities that pose a safety and security threat to the researchers or the environment should submit a risk assessment form annually.
- All research involving human interaction should follow best ethical practise and keep documents as proof. This includes informed consent, questionnaires, incorporation of risk-benefit, and responsible data management.
- Any research at governmental or private institutions, permission must still be obtained from relevant authorities and provided to the FNASREC. Ethics approval is required BEFORE approval can be obtained from these authorities.

APPENDIX B – LANGUAGE EDITING CERTIFICATE

This serves to confirm that I, Isabella Johanna Swart, registered with and accredited as professional translator by the South African Translators' Institute, registration number 1001128, language edited the following mini-dissertation (excluding the Bibliography):

**Pair programming and underrepresented groups in Information
Technology in South Africa**

by

TD Litlhakanyane

A handwritten signature in black ink, appearing to read 'J Swart', with a large, stylized initial 'J'.

Dr Isabel J Swart

Date: 9 December 2021

23 Poinsettia Close
Van der Stel Park
Dormehlsdrift
GEORGE
6529
Tel: (044) 873 0111
Cell: 082 718 4210
e-mail: isaswart@telkomsa.net

APPENDIX C – INFORMED CONSENT

09/12/2021, 15:58

Pair programming and underrepresented groups in Information Technology in South Africa.

Pair programming and underrepresented groups in Information Technology in South Africa.

We would like to invite you to participate in our research project. The aim of the survey is to gather information regarding pair programming and minorities in Information Technology. Respondents' privacy as well as the confidentiality of their answers will be respected.

1. I voluntarily agree to participate in the project.
2. I give permission that the information that is obtained in this questionnaire may be used for research purposes.
3. I understand the procedures with regards to privacy and confidentiality.

* Required

1. By completing this questionnaire, you agree that you have read and understood the information about the research project. *

Check all that apply.

☐ I agree.

APPENDIX D – TURNITIN DIGITAL RECEIPT



Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	DAVID LITHAKANYANE
Assignment title:	David M
Submission title:	25064576:25064576_Final_Dissertation.docx
File name:	b80-4988-42d2-82fc-07da226ef0a1_25064576_Final_Dissertat...
File size:	1.89M
Page count:	163
Word count:	49,717
Character count:	275,459
Submission date:	09-Dec-2021 10:00PM (UTC-0800)
Submission ID:	1726326773



Pair programming and underrepresented groups in Information Technology in South Africa

D Lithakanyane
orcid.org 0000-0002-1268-2200

Dissertation submitted in partial fulfillment of the requirements for the degree *Master of Science in Computer Science* at the North-West University

Supervisor: Dr. JA Liebenberg

Examination November 2021
25064576

Copyright 2021 Turnitin. All rights reserved.