

850472

rb
A LB1028.3
T559
2011

**THE EFFECTS OF METAPHORS AND PAIR PROGRAMMING ON
RECALL AND RETENTION AMONGST STUDENTS WITH DIFFERENT
LEARNING STYLES AND SELF-REGULATED LEARNING LEVELS**

by

TIE HUI HUI

**Thesis submitted in fulfilment of the requirements
for the degree of
Doctor of Philosophy**

February 2011

ACKNOWLEDGEMENTS

First and foremost, I wish to express my sincere gratitude and appreciation to my main thesis supervisor, Associate Professor Dr Irfan Naufal Umar for his invaluable assistance, guidance, unconditional support, insightful ideas, suggestions and continuous encouragement throughout the duration of completing this research study. Many thanks to him for being patiently moulding and guiding this fresh blood who was not that strong at the beginning of the journey to become a better person through the newfound orientated strength and skills that pull her through.

I deeply appreciate Dr Irfan, Associate Professor Dr Nor Azilah Ngah and Universiti Sains Malaysia for providing me with the opportunity to further my graduate education and for their continuous effort in providing quality education. I would also like to thank Associate Professor Dr Wan Mohd Fauzy Wan Ismail, Associate Professor Dr Merza Abbas, Associate Professor Dr Toh Seong Chong, Dr Balakrishnan Muniandy, Dr Mona Masood and Dr Aswati Hamzah for their valuable suggestions, ideas, advice and assistance throughout the graduate study.

My sincere thanks and appreciation to the lecturers for their assistance and participation in the research; and I would like to extend my indebtedness to Mr Tan Teong Guan for prove-reading this write-up. Without them, the study could not have been completed. The students who participated in this study have my gratitude. Your cooperation, efforts and assistance in data collection and analysis have made this research a success. I am also grateful to the management of the private college for supporting this study.

To my parents and friends, thank you for the prayers, love and care given throughout this journey of completing this thesis. To a special friend, Philip Skeane who had been a walking dictionary, counsellor and an advisor constantly standing by me never losing hope in whatever situation. With his encouragement and positive stroke, he eventually is observing from above and looking down at me with my tam.

Last but not least, I would also like to take this opportunity to express my special thanks and extend my gratitude to providence for providing me with strength, courage, perseverance, good health and soul during this difficult period without which this study will not be possible. These have made the journey a valuable life long learning experience. I look forward to life with confidence by knowing that providence is guiding and protecting me. I open up myself to all the advice and help given and I receive them in joy.

This marks the end of one chapter and the beginning of another; and we need to remember and treasure those who have guided us this far – William Shakespeare. This success is a tribute to Associate Professor Dr Irfan Naufal Umar, Philip Skeane and Mr Tan Teong Guan for making the impossible possible.

TABLE OF CONTENTS

Acknowledgements	ii
Table of Contents	iv
List of Tables	x
List of Figures	xiv
List of Abbreviations	xvi
Abstrak	xvii
Abstract	xix

CHAPTER 1: INTRODUCTION

1.0 Introduction	1
1.1 Background of Study	1
1.2 Problem Statement	5
1.3 Research Objective	19
1.4 Theoretical Framework	21
1.5 Research Questions	23
1.6 Hypotheses	25
1.7 Significance of the Study	27
1.8 Operational Definitions	29
1.9 Limitations and Delimitations	33
1.10 Summary	35

CHAPTER 2: LITERATURE REVIEW

2.0 Introduction	37
2.1 Skills in Teaching and Learning Basic Programming Concepts	37
2.1.1 Programming Concepts	43
2.1.2 Process of Learning Programming	43
2.2 Models of Learning Programming	46
2.2.1 McGill and Volet's Conceptual Framework	46
2.2.2 Mayer's Theory in Learning to Programme	49
2.2.3 Programming Knowledge and Challenges	51

2.3	Instructional Strategy	52
2.3.1	Metaphors as an Instructional Strategy	53
2.3.1.1	Research Findings on Metaphors	55
2.3.1.2	Metaphors and Computational Concepts	56
2.3.1.3	Veale and Keane's Metaphorical Framework	58
2.3.1.4	Conceptual Metaphor	61
2.3.1.5	System Metaphors	63
2.3.2	Pair Programming as an Instructional Strategy	64
2.3.2.1	Factors to Consider in Implementing Pair Programming	66
2.3.3	Metaphors with Pair Programming as an Instructional Strategy for Teaching Basic Programming Concepts	68
2.4	Learning Styles	73
2.4.1	The Verbal-Imager Dimension of Learning	75
2.4.2	The Holist-Analytic Dimension of Learning	76
2.4.3	Style Differences in Learning and Behaviour	76
2.4.3.1	Visual Learning Style	78
2.4.3.2	Verbal Learning Style	78
2.4.4	Instrument Measuring Learning Styles	79
2.4.4.1	Felder-Silverman Model of Learning Styles	80
2.5	Self-Regulated Learning	82
2.5.1	Models of Self-Regulated Learning	82
2.5.1.1	Pintrich's Model of Self-Regulated Learning	83
2.5.1.2	Zimmerman's Dimensions of Self-Regulated Learning	86
2.5.2	Categories of Self-Regulated Learning	88
2.6	Summary	89

CHAPTER 3: RESEARCH METHODOLOGY

3.0	Introduction	90
3.1	Research Population and Sample	90
3.2	The Research Variables	91
3.3	Research Design	93
3.4	Data Collection Procedures	95

3.5	The Instruments	97
3.5.1	The Felder and Soloman’s Index of Learning Styles Questionnaire (ILSQ)	97
3.5.2	Computer Programming Performance Test (CPPT)	99
3.5.3	The Motivated Strategies for Learning Questionnaire (MSLQ)	101
3.6	Instructional Materials	105
3.6.1	Topics of Sequence and Selection Control Constructs	105
3.6.2	The Instructor’s Lesson Plan	107
3.6.3	The Instructor’s Protocol	107
3.6.4	Metaphors Training Notes	108
3.6.5	Metaphors Exercise Notes	108
3.6.6	Metaphors Assessment Rubrics	109
3.6.7	Pair Programming Learning Activity	110
3.6.8	Pair Programming Checklist (Instructor’s Guide)	111
3.6.9	Direct Instruction Method Checklist (Instructor’s Guide)	112
3.6.10	Programming Assignments	112
3.6.11	Assessment Score Sheet	113
3.6.12	Group Material Listing	114
3.6.13	Rules for Pair Programming	114
3.7	Validity of the Instruments	115
3.8	Research Procedures and Implementation	116
3.8.1	The Pilot Study	118
3.8.2	Training of Lecturers	120
3.8.3	The Metaphors with Pair Programming (MPP)	122
3.8.4	The Pair Programming (PP)	124
3.8.5	The Direct Instruction Method (DI)	126
3.8.6	Monitoring the Study Implementation	127
3.9	Data Analysis	129
3.10	Summary	130

CHAPTER 4: RESEARCH FINDINGS

4.0	Introduction	132
4.1	Assumption of MANCOVA	132
4.2	The Pre-test	138
4.3	Distribution of Groups	140
4.4	The Results of the Experimental Study	143
4.4.1	Descriptive Statistics	144
4.4.2	Test of Hypothesis 1	146
4.4.2 (a)	Recall Performance	147
4.4.2 (b)	Retention Performance	148
4.4.3	Test of Hypothesis 2	149
4.4.3 (a)	Recall Performance	149
4.4.3 (b)	Retention Performance	150
4.4.4	Test of Hypothesis 3	150
4.4.4 (a)	Recall Performance	151
4.4.4 (b)	Retention Performance	151
4.4.5	Test of Hypothesis 4 to Hypothesis 6	151
4.4.5.1	Test of Hypothesis 4	153
4.4.5.2	Test of Hypothesis 5	153
4.4.5.3	Test of Hypothesis 6	154
4.4.6	Test of Hypothesis 7 to Hypothesis 8	154
4.4.6.1	Test of Hypothesis 7	156
4.4.6.2	Test of Hypothesis 8	157
4.4.7	Test of Hypothesis 9 to Hypothesis 10	158
4.4.7.1	Test of Hypothesis 9	159
4.4.7.2	Test of Hypothesis 10	160
4.5	Summary of Findings	161

CHAPTER 5: DISCUSSIONS, CONCLUSION AND RECOMMENDATIONS

5.0	Introduction	167
5.1	Effects of the Instructional Methods on Recall and Retention	170
5.1.1	Recall Performance	171
5.1.2	Retention Performance	178
5.2	Effects of the Learning Styles on Recall and Retention	180
5.2.1	Overall Effects of Learning styles	181
5.2.2	Effects of Learning Styles on Recall and Retention in Each Treatment Group	183
5.2.2 (a)	Recall Performance Between Visual and Verbal Learners	184
5.2.2 (b)	Retention Performance Between Visual and Verbal Learners	185
5.2.3	Effects of Learning Styles on Recall and Retention in the Three Treatment Groups	190
5.2.3 (a)	Recall Performance for Visual Learners	191
5.2.3 (b)	Retention Performance for Visual Learners	191
5.2.3 (c)	Recall Performance for Verbal Learners	193
5.2.3 (d)	Retention Performance for Verbal Learners	194
5.3	Effects of the Self-Regulated Learning on Recall and Retention	196
5.3.1	Overall Effects of Self-Regulated Learning	196
5.3.2	Effects of Self-Regulated Learning on Recall and Retention in the Three Treatment Groups	197
5.3.2 (a)	Recall Performance for High SRL Learners	198
5.3.2 (b)	Retention Performance for High SRL Learners	200
5.3.2 (c)	Recall Performance for Low SRL Learners	203
5.3.2 (d)	Retention Performance for Low SRL Learners	207
5.4	Recommendation for Future Studies	210
5.5	Conclusion	213
	REFERENCES	216

APPENDICES

A	The Felder and Soloman's Index of Learning Styles Questionnaire (ILSQ)	240
B	McGill and Volet (1997) Conceptual Framework on the Description of Various Components of Programming Knowledge	244
C	Computer Programming Performance Test (CPPT)	246
D	The Motivation Strategies for Learning Questionnaire (MSLQ)	278
E	The Instructor's Lesson Plan	282
F	The Instructor's Protocol	285
G	Metaphors Training Notes	296
H	Metaphors Exercise Notes	310
I	Metaphors Assessment Rubrics	330
J	Pair Programming Learning Activity	333
K	Rules of Pair Programming	335
L	Pair Programming Checklist (Instructor's Guide)	336
M	Direct Instruction Method Checklist (Instructor's Guide)	341
N	Programming Assignments	343
O	Assessment Score Sheet	346
P	Group Material Listing	348

LIST OF PUBLICATIONS	350
----------------------	-----

LIST OF TABLES

		Page
Table 1.1	The maximum and minimum scores obtained by students attempting programming questions in IPD	9
Table 1.2	The maximum and minimum scores obtained by students attempting programming questions in CPM	9
Table 1.3	Percentage of students who attempted programming questions in Introduction to Programming and Database (IPD)	10
Table 1.4	Percentage of students who attempted programming questions in Computer Programming Methodology (CPM)	10
Table 1.5	Grade range percentage in IPD	11
Table 1.6	Grade range percentage in CPM	11
Table 1.7	SPM mathematics scores as entrance qualification in IPD examination terms	12
Table 1.8	SPM mathematics scores as entrance qualification in CPM examination terms	12
Table 1.9	Magnesen's (1983) percentage material retained based on modality of interaction	16
Table 2.1	A conceptual framework of the various components of programming knowledge (McGill & Volet, 1997)	47
Table 2.2	Four categories of programming knowledge (Mayer, 1997)	50
Table 2.3	Instructional strategies in three learning phases (Winnipeg, 1996)	53
Table 2.4	The metaphoric grounds (Veale & Keane, 1994, p.2)	58
Table 2.5	The interpretation of metaphor – “Bank Managers are Vampires” (Veale & Keane, 1994, p.4)	61
Table 2.6	Summary of studies on metaphors and pair programming	72
Table 2.7	A theoretical framework of self-regulated learning (Pintrich, 2000, p.454)	84
Table 2.8	A cyclical model of self-regulated learning phases	87
Table 3.1	The 3 x 2 factorial design involving the visual-verbal learning styles dimension	93

Table 3.2	The 3 x 2 factorial design involving the Self-Regulated Learning dimension	94
Table 3.3	The ILSQ dimensions and items	98
Table 3.4	The MSLQ dimensions and items (Chen, 2002)	101
Table 3.5	The summarised procedure of MPP	124
Table 3.6	The summarised procedure of PP	126
Table 3.7	The summarised procedure of DI	127
Table 4.1	Levene's test of equality of error variances across the three groups	134
Table 4.2	Box's M test of equality of covariance	135
Table 4.3	Wilks' Lambda significance level across the three groups	135
Table 4.4	Bartlett's test of sphericity within MANCOVA	136
Table 4.5	Correlations between dependent variables and the covariate	137
Table 4.6	Homogeneity of the covariance matrix test	138
Table 4.7	Descriptive statistics of pre-test	139
Table 4.8	ANOVA analysis of the pre-test results for the three treatment groups	139
Table 4.9	Summary of post hoc test for the CPPT pre-test in three groups	140
Table 4.10	Distribution of groups based on instructional methods	140
Table 4.11	Distribution of groups based on learning styles and SRL	141
Table 4.12	Distribution of groups based on learning styles	142
Table 4.13	Distribution of groups based on SRL	142
Table 4.14	The descriptive statistics of the overall factorial	142
Table 4.15	Descriptive statistics for the recall performance of the three treatment groups with different learning styles	145
Table 4.16	Descriptive statistics for the retention performance of the three treatment groups with different learning styles	145

Table 4.17	Descriptive statistics for the recall performance of the three treatment groups with different SRL levels	145
Table 4.18	Descriptive statistics for the retention performance of the three treatment groups with different SRL levels	146
Table 4.19	MANCOVA analysis for the (a) recall and (b) retention for the three treatment groups	146
Table 4.20	Summary of post hoc test for recall performance amongst the students in the three treatment groups	147
Table 4.21	Summary of post hoc test for retention performance amongst the students in the three treatment groups	148
Table 4.22	MANCOVA analysis for the (a) recall and (b) retention between the visual and verbal learning style groups	149
Table 4.23	MANCOVA analysis for the (a) recall and (b) retention between the high and low SRL students	150
Table 4.24	MANCOVA analysis for the (a) recall and (b) retention between the visual and verbal students in the three treatment groups	152
Table 4.25	Post hoc result and descriptive statistics for recall between visual and verbal students within the three treatment groups	152
Table 4.26	Post hoc result and descriptive statistics for retention between visual and verbal students within the three treatment groups	153
Table 4.27	MANCOVA analysis for the (a) recall and (b) retention between the visual and verbal students in the three treatment groups	155
Table 4.28	Summary of post hoc test for recall performance between the visual and verbal students in the three treatment groups	155
Table 4.29	Summary of post hoc test for retention performance between the visual and verbal students in the three treatment groups	155
Table 4.30	MANCOVA analysis for the (a) recall and (b) retention between the high and low SRL students in the three treatment groups	158
Table 4.31	Summary of post hoc test for recall performance between the high and low SRL students in the three treatment groups	158
Table 4.32	Summary of post hoc test for retention performance between the high and low SRL students in the three treatment groups	159

Table 4.33	The overall summary of findings (Hypothesis 1 to 10) using MANCOVA	166
Table F.1	Tenors and Vehicles used in the scenario	287
Table G.1	The order of operators with precedence	305

LIST OF FIGURES

		Page
Figure 1.1	Knowledge of programming language (Koffman, 1986)	3
Figure 1.2	Number of students attempted programming questions in IPD in their first semester	8
Figure 1.3	Number of students attempted programming questions in CPM in second semester	8
Figure 1.4	Theoretical framework of this study	22
Figure 1.5	Research framework incorporating learning styles, instructional strategy, self-regulated learning and programming performance	23
Figure 2.1	Programming process	38
Figure 2.2	Steps in the conversion of human thought to programming (Couros, 2004)	42
Figure 2.3	Model of memory involved in learning programming (Muhammed Yousoff, Mohd Sapiyan & Khaja Kamaluddin, 2006)	44
Figure 2.4	Mental schema developed in Long Term Memory (LTM)	44
Figure 2.5	Four modes of instructional interaction (Merrill, 1998, 2002)	52
Figure 2.6	A memory variable sign for a memory location	54
Figure 2.7	The Sapper description of the metaphor – “Bank Managers are Vampires” (Veale & Keane, 1994, p.3)	60
Figure 2.8	Fundamental dimensions of cognitive style (Banner & Rayner, 2000)	75
Figure 3.1	The overall research design	93
Figure 3.2	The formula of measuring learning styles	98
Figure 3.3	The dimensions of MSLQ	103
Figure 4.1	Normal Q-Q plot of recall performance (immediate post-test)	133
Figure 4.2	Normal Q-Q plot of retention performance (delayed post-test)	134
Figure F.1	A triadic model of metaphor	288
Figure F.2	The structural model of metaphors with entailment	288

Figure F.3	A structural model of metaphors introduction	289
Figure G.1	Data type tree structure	296
Figure G.2	The expression of string of characters	296
Figure G.3	The expression of single character type	297
Figure G.4	The ladder metaphor	297
Figure G.5	The expression of integer numbers	298
Figure G.6	The expression of real numbers	298
Figure G.7	The expression of boolean data type	299
Figure G.8	Variables as named memory locations	300
Figure G.9	Memory variables as physical boxes	300
Figure G.10	Cash register machine as meaningful variables	301
Figure G.11	Meaningful name of memory locations/variables	302
Figure G.12	The children's shape toy (Dunican, 2002)	302
Figure G.13	Shapes representing data types	303
Figure G.14	General visual representation	303
Figure G.15	The ticket windows as input-output execution	304
Figure G.16	Calculations show order of operators with lines indicating precedence	306
Figure G.17	The scoreboard-calculator as arithmetic expression	306
Figure G.18	Arithmetic mill and store metaphor	306
Figure G.19	Assignment and arithmetic expression	307
Figure G.20	IF statement T-junction options	307
Figure G.21	T-junction road routes metaphorical concept in Pseudocode	308
Figure G.22	The roundabout metaphor	308
Figure G.23	The roundabout metaphor with five conditions	309
Figure G.24	The roundabout metaphorical concept in Pseudocode	309

LIST OF ABBREVIATIONS

CPM	Computer Programming Methodology
CPPT	Computer Programming Performance Test
DC	Declarative-Conceptual knowledge
DI	Direct Instruction
DICS	Diploma in Computer Studies
DIT	Diploma in Information Technology
DS	Declarative-Syntactic knowledge
ILSQ	Felder and Soloman's Index of Learning Styles Questionnaire
IP	Introduction to Programming
IPD	Introduction to Programming and Database
IT	Information Technology
MPP	Metaphors with Pair Programming
MSLQ	Motivated Strategies for Learning Questionnaire
PC	Procedural-Conceptual knowledge
PDL	Program Design Language
PP	Pair Programming
PS	Procedural-Syntactic knowledge
SC	Strategic-Conditional knowledge
SPM	Sijil Pelajaran Malaysia
SRL	Self-Regulated Learning
VeLS	Verbal Learning Style
ViLS	Visual Learning Style
VPL	Visual Programming Language
XP	Extreme Programming

**KESAN METAFORA DAN PENGATURCARAAN BERPASANGAN
TERHADAP PRESTASI INGAT-KEMBALI DAN KETEKALAN
PELAJAR YANG BERBEZA GAYA PEMBELAJARAN DAN
TAHAP PENGAWALAN KENDIRI**

ABSTRAK

Kajian ini bertujuan meneliti kesan tiga kaedah pengajaran iaitu kombinasi Metafora dan Pengaturcaraan secara Berpasangan (MPP), Pengaturcaraan secara Berpasangan (PP) dan kaedah Pengajaran Langsung (PL) terhadap prestasi ingat-kembali dan ketekalan para pelajar sains komputer di institusi pengajian tinggi di Malaysia. Selain itu, kajian ini bertujuan mengkaji kesan kaedah-kaedah ini terhadap prestasi para pelajar yang berbeza gaya pembelajaran serta tahap pengawalan sendiri (TPK). Kajian kuasi-eksperimen berbentuk faktorial 3×2 ini melibatkan ujian-pra dan ujian-pasca. Faktor pertama melibatkan kaedah pengajaran sama ada MPP, PP atau PL. Faktor kedua melibatkan dua pembolehubah moderator iaitu gaya pembelajaran (visual atau verbal) yang diukur menggunakan Indeks Gaya Pembelajaran Felder dan Soloman (2002) serta TPK (tinggi atau rendah) yang diukur menggunakan Soal Selidik Strategi Motivasi Pembelajaran (Pintrich & DeGroot, 1990). Prestasi ingat-kembali dan ketekalan pelajar dijadikan pembolehubah bersandar yang diukur menggunakan Ujian Prestasi Pemrograman Komputer (UPPK) serta-merta dan ujian UPPK tertunda. Sejumlah 123 pelajar dari sebuah kolej swasta di Pulau Pinang terlibat dalam kajian ini dan mereka dipilih secara rawak untuk menerima sama ada MPP, PP atau PL. Skor ujian-pra UPPK digunakan bagi menguji kesetaraan varians antara ketiga-tiga kumpulan kajian ini.

Dapatan kesan utama menunjukkan MPP dan PP memperolehi skor ingat-kembali dan ketekalan lebih tinggi secara signifikan berbanding PL. Juga, pelajar verbal memperolehi skor ketekalan lebih baik secara signifikan berbanding pelajar visual. Seterusnya, prestasi ingat-kembali dan ketekalan pelajar TPK tinggi adalah lebih baik secara signifikan berbanding pelajar TPK rendah. Dapatan ujian post-hoc pula menunjukkan: (i) pelajar visual dalam MPP dan PP mencapai prestasi ingat-kembali lebih baik secara signifikan berbanding rakan visual mereka dalam PL, namun bagi ketekalan, hanya MPP menunjukkan pencapaian lebih baik secara signifikan berbanding PL, (ii) bagi pelajar verbal, kumpulan MPP dan PP menunjukkan prestasi lebih baik dan signifikan bagi ingat-kembali berbanding PL, (iii) pelajar verbal mengatasi pelajar visual secara signifikan dalam ketekalan untuk kumpulan PP dan PL, (iv) pelajar TPK tinggi dalam MPP mencapai prestasi lebih baik secara signifikan berbanding rakan-rakan mereka di PP dan PL untuk ingat-kembali dan ketekalan, (v) wujud perbezaan signifikan dalam ingat-kembali antara para pelajar TPK rendah dalam kesemua kumpulan, dengan MPP menunjukkan pencapaian lebih baik berbanding PP dan PL, serta PP mengatasi pencapaian PL, namun bagi ketekalan, hanya pelajar TPK rendah dalam MPP menunjukkan prestasi lebih baik secara signifikan berbanding rakan mereka dalam PL.

Kajian ini mencadangkan kaedah pengaturcaraan berpasangan dalam meningkatkan prestasi pengaturcaraan pelajar. Juga, penggabungan kaedah ini dan metafora adalah penting bagi meningkatkan prestasi mereka. Di samping itu, gaya pembelajaran dan TPK pelajar perlu dikenalpasti memandangkan ia adalah peramal prestasi mereka dalam konteks pendidikan pengaturcaraan.

**THE EFFECTS OF METAPHORS AND PAIR PROGRAMMING ON
RECALL AND RETENTION AMONGST STUDENTS WITH DIFFERENT
LEARNING STYLES AND SELF-REGULATED LEARNING LEVELS**

ABSTRACT

The study aimed to investigate the effects of (i) Metaphors with Pair Programming (MPP), (ii) Pair Programming (PP) and (iii) Direct Instruction Method (DI) on the students' recall and retention of programming performance amongst computing students in an institution of higher learning in Malaysia. This study further examined the effects of these three methods on the performance of students with different learning style and self-regulated learning levels (SRL). A 3 x 2 factorial design quasi experimental study with pre-test and post-test control groups design was applied in this study. The first factor was the instructional method, namely the MPP, PP and DI. The second factor had two moderating variables, i.e., the students' learning style (visual or verbal) which was measured using the Felder and Soloman's Index of Learning Styles Questionnaire (2002), and their SRL level (high or low) measured using the Motivated Strategies of Learning Questionnaire (Pintrich & DeGroot, 1990). Meanwhile, the students' recall and retention scores as the dependent variables were obtained from the immediate Computer Programming Performance Test (CPPT) and delayed CPPT. A sample of 123 students from a private college in Penang was selected and randomly assigned to the treatment groups. The CPPT pre-test scores were obtained to analyse the homogeneity of variance amongst the three groups.

The findings of the main effects showed that the MPP and PP groups significantly outperformed the DI group in recall and retention. The verbal students performed significantly better than the visual students in retention. Moreover, the high SRL students significantly outperformed the low SRL students in both performance tests. The post-hoc test revealed that: (i) the visual students in both MPP and PP groups significantly outperformed their peers in the DI group in recall, but only those in the MPP group significantly outperformed those in the DI group in retention, (ii) for the verbal students, both the MPP and PP groups significantly outperformed their peers in the DI group in recall, (iii) the verbal students significantly outperformed the visual students in retention in both the PP and DI groups, (iv) high SRL students in the MPP group significantly outperformed their peers in both PP and DI for recall and retention, and (v) amongst the low SRL students, a significant difference in recall was observed in the three groups, with MPP significantly outperformed PP and DI, and PP significantly outperformed DI, however, for retention, only the MPP group significantly outperformed those in the DI group.

This study recommends the use of pair programming method to improve the students' programming performance. Blending metaphors with pair programming is also essential in order to achieve positive programming performance. In addition, students' learning styles and their self-regulated learning levels need to be identified as they are the predictors for their performance especially in the context of programming education.

CHAPTER 1

INTRODUCTION

1.0 Introduction

Many business organisations have integrated information systems as a crucial element into their competitive strategy planning. Therefore, the needs for software developers and programmers are rising as information technology is continuously being incorporated into the architecture and strategy of organisations. Nevertheless, our country is still short of competent and skilled programmers. Therefore, it is becoming more important to provide students with appropriate programming skills for the job market.

Computer programming, being an essential part of computer studies curriculum is the main stumbling block for most students, especially in the first year of study. Covering basic concepts of programming is one of the three primary pedagogical goals of teaching programming, besides programming design skills and creative thinking. Mastering the programming concept is critical as it prepares learners for the next higher programming courses in their academic degree programmes.

1.1 Background of Study

The faculty of computing at a local private college offers a Diploma in Computer Studies (DICS) and a Diploma in Information Technology (DIT) to first year computing students. For the first semester, Introduction to Programming and

Database (IPD) course is the first course for students studying DICS; whereas the Introduction to Programming (IP) course is offered to those taking DIT. Both are theory papers, here, the students are introduced to the concepts of programming and language structures, such as variables, loops and conditional statements. The DICS students will proceed to the Computer Programming Methodology (CPM) course in their second semester. The students are to apply what they have learned in IPD based on the basic programming logic in this field. Pseudocode and program flowchart are the two basic tools applied in explaining and designing programmes. In learning programming, the students are required to understand the given novel scenarios. Once these problem requirements have been identified, the basic programming concepts are used for developing the programme solution.

McGill and Volet (1997) discovered that a common approach employed in teaching computer programming concepts to the students is to first teach the basic syntax of programming languages which emphasizes on the low level knowledge (declarative knowledge and procedural knowledge) that stresses on “know that” and “know how” in order to guide the students towards effective strategies for the whole programming processes.

The teaching and learning sequence starts from syntax, and proceeds to semantics and pragmatics of the language-like tools, as shown in Figure 1.1. When the students fail to comprehend the basic syntactical concepts in programming then they will not be able to describe semantics actions in programmes which is the second sequence of the learning stages. Over emphasizing on the low level knowledge will lead to the students’ misunderstanding of the basic programming

syntax and constructs. Thus, educators who have been involved in teaching programming concepts to first year computing students have tried in vain to cultivate learners' understanding in the fundamental area of semantics, which is program comprehension (Oliver & Malone, 1998; Haynes, 1998; DeCorte, Verschaffel & Schrooten, 1992; Linn & Dalbey, 1985; Linn, 1985).

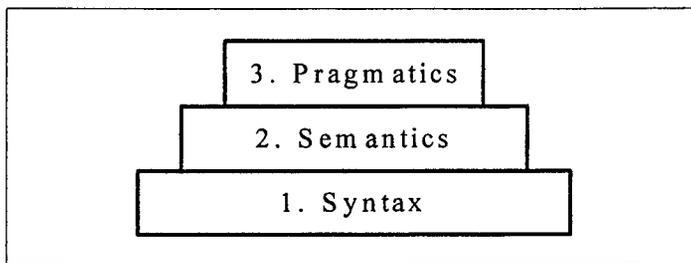


Figure 1.1. Knowledge of programming language (Koffman, 1986)

Perhaps other proxy instructional methods such as metaphors, cooperative learning, mind mapping, problem based learning and scaffolding could be considered in delivering computing courses. This study is to examine the effects of combining the metaphor and pair programming strategies in learning the basic programming concepts on the students' recall and retention performance amongst the students learning computing.

Learning style is the preferred mode in which the students respond to the learning context. It serves as a relatively stable indicator of how they perceive and process information in the learning environment (Felder & Brent, 2005). In most formal classroom education, course deliveries are in the forms of oral lectures and written text on the whiteboard and intermittently supplement with some diagrammatic illustrations that only benefit those students in the verbal-imager

continuum of Banner and Rayner's dimension of cognitive style (2000). Empirical studies revealed that learning style is blended with cognitive and physiological behavioural elements that will somehow influence the students' programming performance in terms of recall and retention (Pallapu, 2007; Oxford & Ehrman, 1988).

Meanwhile, Self-Regulated Learning (SRL) is a self-initiated action whereby the students set goals for their learning process which includes planning, monitoring, controlling and self reflection. This allows them to initiate, persist and disengage in acquiring new knowledge and skills according to individual learning pace that is oriented towards attaining own goals. Those with high SRL whom usually are independent learners, have the ability to regulate learning towards a desirable learning outcome and the skill to manage and organise their own learning needs, strategies and learning opportunities. Thus, SRL has positive effects on the students' learning abilities that subsequently enhance their programming knowledge and improve their programming recall and retention performance.

Thus, besides considering the alternative instructional methods to be used in course deliveries, the learning style preference (visual and verbal) and the different level of self-regulated learning (high or low) amongst students need to be considered in order to stimulate higher programming achievement in terms of recall and retention performance.

1.2 Problem Statement

Several researchers have investigated the problems faced by novices in learning computer programming concepts. Robins, Rountree and Rountree (2003) presented a fully inclusive review on the research pertaining to programming education. These papers provide common viewpoints on the characteristics and the misconceptions of novices which could be contemplated when designing approaches for programming education. An earlier research on this area has been carried out when Pseudocode and program flowchart are concentrated mainly on the concepts of basic programme constructs.

Computer programming demands complex cognitive skills. These sources conclude that planning, logical reasoning and problem solving play their role in the process of learning programming (Miliszewska & Tan, 2007; Dunican, 2002; Kurland, Pea, Clement & Mawby, 1986). Meanwhile, Soloway (2003), McGill and Volet (1997) noted that problem solving and critical thinking are both the ultimate skills sought after in following programming courses.

Learning programming is a complex task for novice students. It is not just about learning some programming language syntax. It also involves the students' ability to develop an algorithm that could solve a given problem scenario efficiently. As such, students find the application of basic programming concepts (sequence, selection and iteration) too difficult to comprehend. Previous research findings revealed some similarities in problem faced by students while learning these most fundamental concepts (Magoc, Freudenthal & Modave, 2010; Redondo, Bravo & Molina, 2004). This is usually the case as programming demands problem solving

skills that include logical reasoning and analytical thinking for analysing the novel scenarios before converting them into programming solution. For first year computing students, the lack of understanding in the basic programming concepts may be a major obstacle to learning higher programming courses. In all probability, students have seen and memorised these fundamental concepts in programming courses but have often not been comprehended deeply enough to be able to apply them to “real-life” problems in higher level programming courses. Therefore, programming tools such as program flowchart and Pseudocode used in solving novel problems are taught in the basic programming courses; and so will assist students to develop better understanding on programming concepts. In turn, it helps the students to overcome common challenges face while they are exposed to higher level of programming.

Universities in the West are raising concern regarding to the increase in high attrition and failure rates of first year programming modules (Tavares, Brzinski & Huet, 2001). Students who struggle with programming will eventually withdraw from computing courses; those who continue however will assiduously avoid future programming projects and ultimately choose a career path that does not involve programming. In accordance with Smith, Cypher and Tesler (2000), final year students often cannot programme at all.

From the researcher’s experience in teaching computer programming courses, students often approach programming “line by line” rather than using meaningful programme structures, for examples variables, sequence, selection and iteration. Inability to create meaningful comments and use meaningful variables in the

program flowchart and Pseudocode is another problem. Very often the students know the programming syntax and semantics of individual statement; however, they do not know how to combine them into valid workable programmes. This puts the students at a disadvantage as they do not know where to start. This starting problem is in alignment with that stated by Spohrer and Soloway (1986). Another difficulty faced by computing students is the need to understand that each instruction is executed in the state which has been created in the previous instructions. While this problem is crucial as they could identify the syntax and indentation errors in a given program flowchart or Pseudocode, unfortunately they do not notice the logical errors.

Generally, this traditional “chalk and talk” method of teaching approach is not effective in developing analytical and logical skills in learners. Furthermore, most lecturers conducting computing classes in the local private college have not undergone formal educational training. These lecturers are not graduate educators but are graduates of computer science or other disciplines with some qualification in computing and information technology. Thus, the adage “a lecturer teaches how he was taught and not otherwise”. As such, limitation in delivery is been shown.

A similar case has been reported in a local private college in Malaysia. In total, 179 students from the six examination semesters who sat for the Introduction to Programming and Database (IPD) and 146 students from second semester taking the Computer Programming Methodology (CPM) in the five examination terms have been studied. The majority of these candidates avoided answering programming type of questions in every examination as shown in Figure 1.2 and Figure 1.3. Meanwhile,

the data indicate that the passing rate for IPD is high (above 70%) whereas for the CPM course, the passing is only average (above 64%).

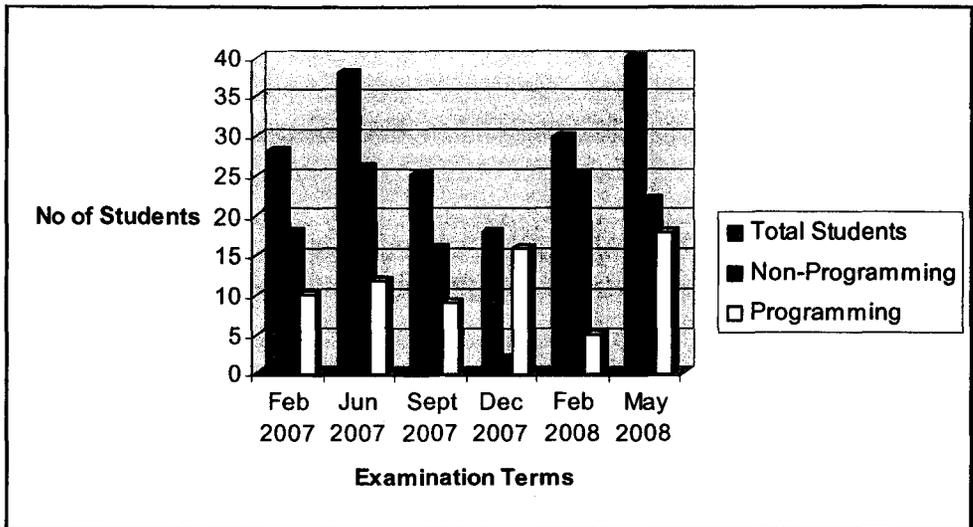


Figure 1.2. Number of students attempted programming questions in IPD in their first semester

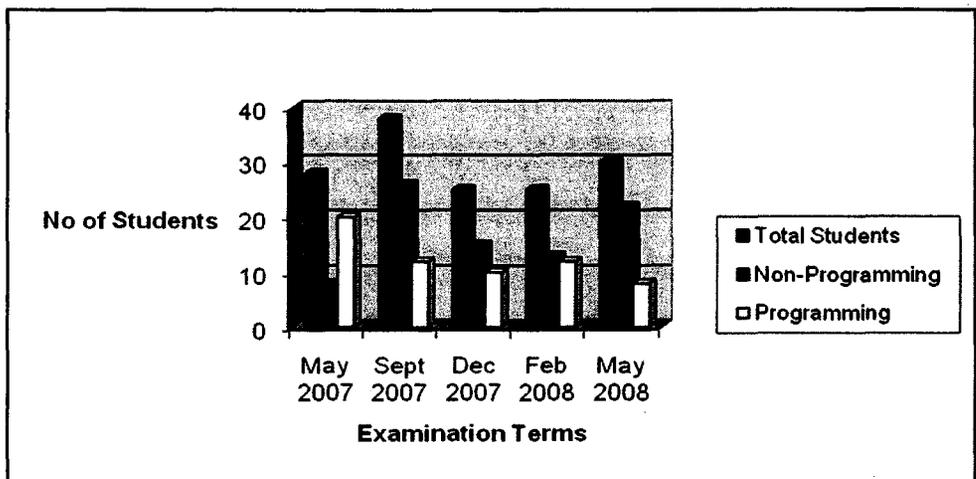


Figure 1.3. Number of students attempted programming questions in CPM in second semester

Table 1.1 and Table 1.2 show the minimum and maximum scores obtained by the students in the two examinations. Statistically, they indicate that most students answered poorly in Pseudocode and program flowchart questions and those who are strong in logical reasoning outperformed the others in these papers. All absentees have been removed from this analysis; this decision is based on the number of students who sat for the examination.

Table 1.1

The maximum and minimum scores obtained by students attempting programming questions in IPD

Exam Terms	Total Students	Marks (20)	
		Maximum	Minimum
Feb 2007	28	8	0
Jun 2007	38	20	6
Sept 2007	25	9	5
Dec 2007	18	12	0
Feb 2008	30	7	0
May 2008	40	19	12

Table 1.2

The maximum and minimum scores obtained by students attempting programming questions in CPM

Exam Terms	Total Students	Marks (20)	
		Maximum	Minimum
May 2007	28	17	0
Sept 2007	38	11	0
Dec 2007	25	13	3
Feb 2008	25	11	0
May 2008	30	12	5

In semester one and two, only 43 percent and 44 percent of the students respectively chose programming logic questions and other selected theories as shown in Table 1.3 and Table 1.4. In December 2007 (Table 1.3), 89 percent and May 2007 (Table 1.4), 71 percent of them chose programming questions. These two passing percentages were higher when compared to other examinations cohorts because the majority of these students who sat for the course were the repeating candidates. Statistically, it has shown that repeating candidates have a higher potential of attempting the programming questions.

Table 1.3

Percentage of students who attempted programming questions in Introduction to Programming and Database (IPD)

Exam Terms	Total Students	Questions Attempted by Students			
		Non-Programming	%	Programming	%
Feb 2007	28	18	64	10	36
Jun 2007	38	26	68	12	32
Sept 2007	25	16	64	9	36
Dec 2007	18	2	11	16	89
Feb 2008	30	25	83	5	17
May 2008	40	22	55	18	45

Table 1.4

Percentage of students who attempted programming questions in Computer Programming Methodology (CPM)

Exam Terms	Total Students	Questions Attempted by Students			
		Non-Programming	%	Programming	%
May 2007	28	8	29	20	71
Sept 2007	38	26	68	12	32
Dec 2007	25	15	60	10	40
Feb 2008	25	13	52	12	48
May 2008	30	22	73	8	27

An analysis of the examination grade scores reveals that the students were not uniformly distributed across groups of grades. The percentage of students in grade D (26 – 39) and grade C (21 – 30) is high for IPD, as shown in Table 1.5.

Table 1.5
Grade range percentage in IPD

Exam Terms	No. of Students	F		D		C		B		A	
		0 - 39	%	40 - 49	%	50 - 59	%	60 - 69	%	70 - 100	%
Feb 2007	28	7	25	11	39	6	21	1	4	3	11
Jun 2007	38	9	24	10	26	9	24	6	16	4	11
Sept 2007	25	4	16	8	32	7	28	--	0	6	24
Dec 2007	18	3	17	6	33	4	22	3	17	2	11
Feb 2008	30	7	23	11	37	9	30	2	7	1	3
May 2008	40	5	13	15	38	10	25	6	15	4	10

Statistically, it shows that theory questions were their first option. With the lacking of problem solving skills, students who encountered difficulties in programming have progressed to semester two and yet they fail to master it, as shown in Table 1.6. The majority of these students have achieved grade D (29 – 40) and grade C (8 – 17) in CPM. This indicates that the students have failed to apply what they have learned in IPD (first semester).

Table 1.6
Grade range percentage in CPM

Exam Terms	No. of Students	F		D		C		B		A	
		0 - 39	%	40 - 49	%	50 - 59	%	60 - 69	%	70 - 100	%
May 2007	28	9	32	8	29	4	14	3	11	4	14
Sept 2007	38	16	42	15	39	3	8	2	5	2	5
Dec 2007	25	10	40	8	32	2	8	3	12	2	8
Feb 2008	25	7	28	10	40	4	16	4	16	--	0
May 2008	30	11	37	10	33	5	17	1	3	3	10

As shown in Table 1.7 and Table 1.8, the initial analysis of the SPM entrance qualification reveals that majority of the students have achieved grade B (4-3) and grade C (6-5); with an average percentage (11-32) of students obtaining grade A (2-1) and a small percentage in grade D (7) in mathematics. This indicates that these students who face difficulties in learning computing courses are not weak learners in general. In fact, it is the nature of the course that programming concepts demand complex cognitive tasks and logical problem solving skills which in turn tend to be difficult for the students to understand.

Table 1.7
SPM mathematics scores as entrance qualification in IPD examination terms

Exam Terms	No. of Students	D		C		B		A	
		7	%	6 - 5	%	4 - 3	%	2 - 1	%
Feb 2007	28	1	4	11	39	9	32	7	25
Jun 2007	38	2	5	15	40	13	34	8	21
Sept 2007	25	--	0	8	32	13	52	4	16
Dec 2007	18	2	11	8	45	6	33	2	11
Feb 2008	30	3	10	6	20	16	53	5	17
May 2008	40	3	8	8	20	18	45	11	27

Table 1.8
SPM mathematics scores as entrance qualification in CPM examination terms

Exam Terms	No. of Students	D		C		B		A	
		7	%	6 - 5	%	4 - 3	%	2 - 1	%
May 2007	28	--	0	10	36	11	39	7	25
Sept 2007	38	--	0	15	40	16	42	7	18
Dec 2007	25	1	4	7	28	9	36	8	32
Feb 2008	25	--	0	8	32	12	48	5	20
May 2008	30	4	13	6	20	15	50	5	17

The findings on the previous three terms examination papers (May 2008, February 2008 and December 2007) revealed that these past examination papers were assessing the students' understanding of basic programming concepts, were based on the McGill and Volet's (1997) programming conceptual framework. 60 percent of the assessment items were tested on the high level knowledge such as (i) procedure-syntactic, (ii) procedure-conceptual and (iii) strategic/conditional as compared to only 40 percent on the students' low level knowledge of the basic programming concepts which are the declarative-syntactic and declarative-conceptual knowledge. For the students to attempt the complex high level questions, the first year semester one examination papers should have more questions which test on low level knowledge so that they could build their confidence level when attempting programming problems as well as encouraging them to be involved with programming solutions in the future.

Eight senior lecturers of programming courses at the Faculty of Information Technology (IT) department in a local college were interviewed in order to explore the problems faced by them with regard to teaching programming modules and their views on the difficulties encountered by the first year students in programming. Quantitative interviews were chosen as the aim is to initially explore the area. The main criterion for selecting these lecturers is based on the number of years having taught computing courses.

Questions regarding the aim, size and duration of the course and the lecturers' experience were asked during the interview sessions. The purpose is to have a background understanding of the courses conducted. Five out of eight lecturers

interviewed are master degree holders and others have a basic degree qualification. The three lecturers are currently pursuing their master degree. Six lecturers have over 10 years of teaching experience. However, the remaining two have several years of industry experience prior to teaching. They are involved in software engineering and projects development.

In this study, the eight lecturers have identified five difficulties which are perceived to be important in learning programming. These five main difficulties are mainly (i) programming syntax and concepts are difficult to comprehend and master, (ii) lacking of problem solving and analytical thinking skills, (iii) ineffective use of programme structures for problem solving, (iv) difficulties in understanding that each instruction is executed in the state which has been created by the previous instructions, and (v) inadequate approach on the effectiveness of the languages (pedagogical or industry standard), paradigms (procedural, functional or object oriented) and teaching methodologies (“objects first” or “object later”).

The outcome of the interview survey was similar to that in the literature on computer programming. According to Mohd Nasir (2008), the difficulties in mastering programming syntax closely mirrored the problems identified by the senior lecturers. Another concern raised by these lecturers is the effectiveness of languages, paradigms and teaching methodologies used, which is very closely related to Mohd Nasir’s findings.

Research suggested that complex cognitive skills such as planning, reasoning, problem solving and analytical thinking play their role in learning to programme (Milliszewska & Tan, 2007; Dunican, 2002; Kurland, et al., 1986). Problem solving skills which include reasoning and analytical thinking are needed to analyse a given problem scenario. Students then need to understand the various problem presentation tools and programming languages before applying them effectively and correctly in the given programming scenario. Nevertheless, doing lots of exercises is the way of learning how to programme. However, this could not mean that novices could master programming in such a short duration as it takes about ten years for them to become expert programmers (Soloway & Spohrer, 1989).

Lecturers are adopting numerous approaches to the teaching of programming. Through non-participant observation of lecturers and also from interviews with faculties, they conclude that teaching methods are very similar in the approach. The approaches are lectures, reading, and practical exercises of each lesson explaining basic concepts.

In line with McGill and Volet (1997), most of the introduction to programming courses focuses only on the low level knowledge (also known as the declarative knowledge and procedural knowledge). The declarative and procedural knowledge emphasize on “what” and “how”. As such, these knowledge are related to the “what” and “how” of the basic programming concepts.

Shih and Alessi (1994) believed that by over emphasis on the “how to” may not facilitate the transfer of what was learned to novel situation as it did not highlight the knowledge underlying such skills. However, over emphasis on the “why” could result in a mismatch between instruction and hands-on practice, although learners are provided with a wider knowledge base which can be applied in a variety of contexts. Students have been guided by formal instruction in the classroom in acquiring the “how to” skills of programming, but have not been assisted to understand the “why”.

Gage and Berliner (1992) outline the circumstances where direct instruction teaching is not appropriate: (i) objectives other than the acquisition of information is sought, (ii) long term retention is necessary, (iii) the material is complex, detailed or abstract, (iv) learner participation is essential to achievement of objectives, and (v) higher-level cognitive objectives (analysis, synthesis, evaluation) are the purpose of instruction. Given these research findings and the fact that they partially mirror the objectives of teaching programming, one can conclude that the current lecture mode of teaching is not suitable to teach programming. Table 1.9 shows the average percentages of material retained in long-term memory based on the modality of interaction as suggested by Magnesen (1983).

Table 1.9
Magnesen’s (1983) percentage material retained based on modality of interaction

Modality of Interaction	Percentage Retained Long Term
Reading	10%
Hearing	20%
Seeing	30%
Seeing and hearing	50%
Discussing	70%
Doing and discussing	90%
Teaching and tutoring	95%

The importance of “doing and discussing” and “teaching and tutoring” has forced a re-evaluation of how programming is taught with new and innovative approaches being developed which are a variation on the direct instruction approach. Educators involved in the teaching of programming need to reconsider their approach to teaching in light of current theories on cognition.

The learning methods and programming performance of individual learner are closely related to the different levels of Self-Regulated Learning (SRL) abilities (Kerka, 2005; VanDeWiel, Szegedi & Weggeman, 2004). The use of cooperative learning through pair programming groups which consist of learners with different learning style and different levels of self-regulated learning abilities theoretically influence their proficiency in learning (Song & Hill, 2007; Jossberger, Brand-Gruwel & Boshuizen, 2006; Felder & Brent, 2005; Felder & Spurlin, 2005; Reder & Strawn, 2001). However, the studies of SRL were mainly in learning situations involving adult learners, medical and linguistic learners. No study has been conducted to investigate the students’ programming performance (recall and retention) amongst computing students with high and low self-regulating abilities in relation to the use of different instructional strategies and learning style preference. Thus, this study would like to investigate whether metaphors with pair programming and learning style (visual and verbal) involving the students with different level of SRL abilities assist their learning of basic programming concepts.

There are empirical studies of visualisation approaches (Baldwin & Kuljis, 2001; Shu, 1992) and pair programming, formally known as Extreme Programming (XP) for programming education (Beck, 2000, 2005; McDowell, Brian & Linda,

2003; Williams & Upchurch, 2001). A lot of proposals for Visual Programming Languages (VPLs) are to be incorporated into teaching programming. Visual expressions such as diagrams, free-hand sketches, icons or graphical manipulators are used in visual programming. The empirical evidence indicated the benefits such as understanding and learning the abstract and concepts of the field resulting from diagram used in programming.

Visual programming techniques are often conceived with reference to metaphors (Baldwin & Kuljis, 2000). The explanation of variables declaration – variables represent memory location – is more easily learned when expressed in terms of metaphorical pigeon holes that represented memory locations. Furthermore, metaphors have substantial instructional benefits in relating the abstract nature of the programming task to the fundamental programming concepts.

Most of the approaches, however, concentrated on algorithm animation and no emphasis has been given to visualising the basic structure of programming concepts. These basic programming skills which are variables declarations and usage, conditionals, loops and basic data structures are essential for success with any programming paradigms. Literature review indicated that those novices whose first introduction to programming is in an object oriented environment may fail to acquire these basic programming skills. Duke, Salzman, Burmeister, Poon and Murray (2000, p.84) stated that “in later years, students who have not adequately mastered these basic programming skills (using while loops and Boolean expressions to capture a system’s internal logic) may be able to create higher-level designs, but struggle to convert those designs into actual code”.

Pair programming has been applied in software development industry to increase productivity on programmers and in education to increase learning. Educators believe that it has educational benefits in enhancing learners programming performance.

Alternative methods and techniques for teaching and learning need to be considered besides the direct instructional model, based on the transmission and reception of information and centered in the lecturer's illustration. Thus, this study plans to investigate the comprehensibility of program flowcharts and textual "Pseudocode" as representations for conditional logic in novices' knowledge and understanding regarding to basic programming concepts. Are these two approaches adequately helping students to view conditional logic in solving programming problems? Is there any other alternative approach besides these two notations? Could the concept of metaphors and pair programming aid learners in the learning of programming? How effective these methods will be in terms of in improving the student's performance based on recall and retention in learning the basic programming concepts? These are few of the questions need to be investigated and answered in this study.

1.3 Research Objective

In this study, the learners were grouped and taught in three different instructional methods. These methods were Metaphors with Pair Programming (MPP), Pair Programming (PP) and Direct Instruction (DI). The objectives of this study were to:-

1. identify whether metaphors could enhance the learning of the basic programming concepts.
2. investigate the effectiveness of pair programming in improving students' recall and retention.
3. investigate the effects of combining the metaphor and pair programming strategy on the students' recall and retention performance.
4. examine the effects of these three instructional methods on Visual Learning Style (ViLS) students and Verbal Learning Style (VeLS) students in their computer programming performance.
5. investigate if there are any significant differences in terms of students' performance between the ViLS and VeLS students in Metaphors with Pair Programming (MPP), Pair Programming (PP) and Direct Instruction (DI) groups.
6. investigate whether Self-Regulated Learning (SRL) will affect the students' recall and retention based on their individual classification (high SRL or low SRL).
7. investigate the effects of these instructional methods on high SRL students and low SRL students in their programming performance.

The emphasis of this research was on whether visualisation technique through the use of metaphors and cooperative learning through pair programming for use in classroom as effective alternative solutions in programming education as compared to the direct instructional approach.

1.4 Theoretical Framework

The theoretical framework of this study (Figure 1.4) is based on Felder and Silverman's (1988) learning style model and Pintrich and DeGroot (1990) motivated strategies on self-regulated learning that leads to the development of the instructional strategy which is based on Veale and Keane's (1993) metaphor theory and pair programming. The assumption is that the learning style of students reflects on how each of them perceives (grasps) and processes (transforms) new information. To make learning computer programming more engaging, interactive and fun, the students are required to have high self-regulated learning skills. These learning processes are achievable through (i) knowing the learning style of individual student, as compared to the teaching style of instructors, (ii) the interaction between the students and lecturer throughout classroom lectures, or (iii) the use of instructional strategy to assist the students in learning computer programming.

In this study, the concept of metaphor is used as an instructional strategy in program flowchart and Pseudocode when explaining the basic programming concepts. In addition, pair programming as a cooperative learning technique is used as an instructional strategy to explain programming syntaxes. This study aims to enhance the students' programming performance (recall and retention) in learning the basic programming concepts.

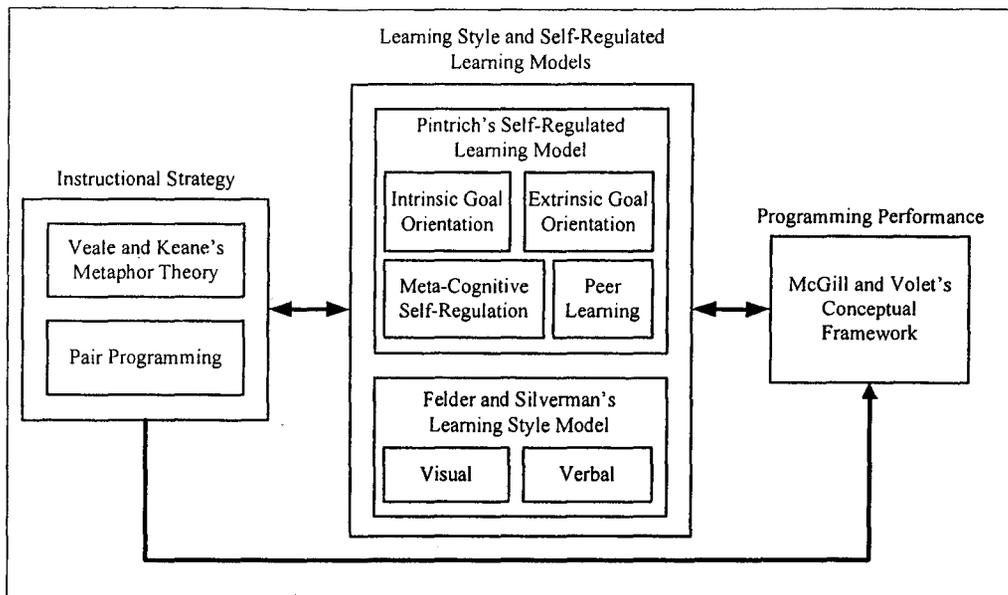


Figure 1.4. Theoretical framework of this study

Moreover, this theory is based on McGill and Volet's conceptual framework which integrates the three components of programming knowledge (syntactic, conceptual, and strategic) classified in the computing educational literature with the three types of cognitive knowledge (declarative, procedural, and conditional) proposed in the cognitive psychology literature. The conceptual framework established five categories of programming knowledge required for learning programming. These categories served to define several ways for the students to comprehend the basic programming concepts. The research framework based on the theoretical framework used for this study is presented in Figure 1.5.

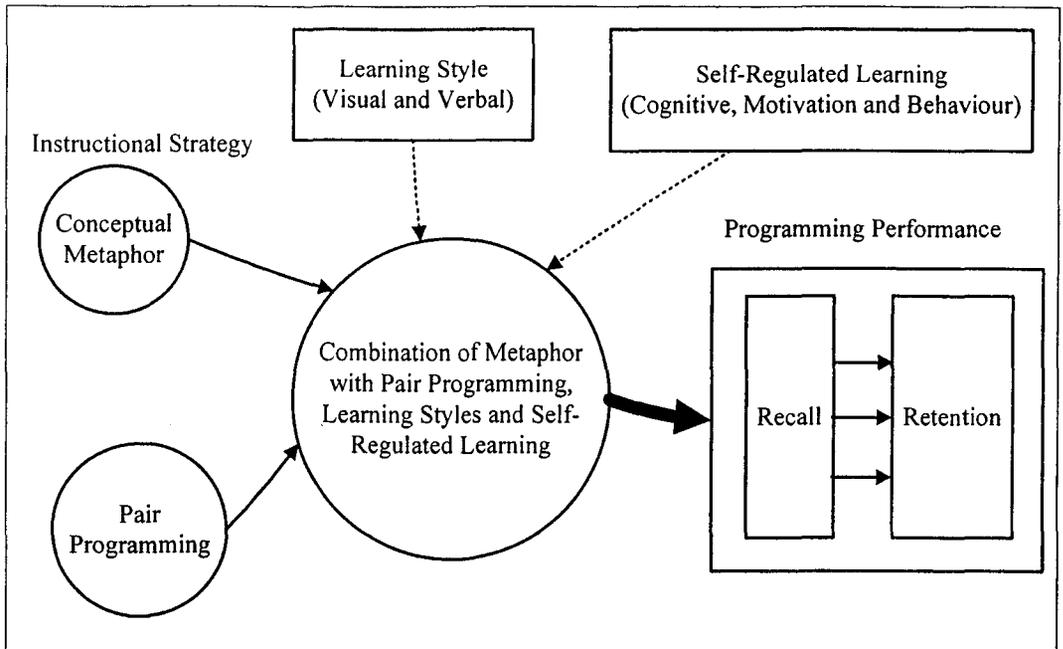


Figure 1.5. Research framework incorporating learning styles, instructional strategy, self-regulated learning and programming performance

1.5 Research Questions

Below are the three primary research questions.

1. Are there any significant differences in terms of (a) recall and (b) retention amongst learners taught in the MPP, PP and DI instructional methods?
2. Are there any significant differences in terms of (a) recall and (b) retention between the visual and verbal learners?
3. Are there any significant differences in (a) recall and (b) retention between the high SRL learners and low SRL learners?

The secondary research questions are listed as follows:-

4. Could the effects of instructional methods be moderated by the factors of learning styles?
 - (i) Are there any significant differences in terms of (a) recall and (b) retention between visual and verbal learners taught in the MPP instructional method?
 - (ii) Are there any significant differences in terms of (a) recall and (b) retention between visual and verbal learners taught in the PP instructional method?
 - (iii) Are there any significant differences in terms of (a) recall and (b) retention between visual and verbal learners taught in the DI instructional method?
5. Could the learning styles factor be affected by the instructional methods?
 - (i) Are there any significant differences in terms of (a) recall and (b) retention amongst the visual learners taught in the MPP, PP and DI instructional methods?
 - (ii) Are there any significant differences in terms of (a) recall and (b) retention amongst the verbal learners taught in the MPP, PP and DI instructional methods?