Available online at www.sciencedirect.com

ScienceDirect

Procedia - Social and Behavioral Sciences 00 (2012) 000–000

Procedia Social and Behavioral Sciences

www.elsevier.com/locate/procedia

WCES 2012

Learning style, metaphor and pair programming: Do they influence performance?

Irfan Naufal Umar [a] *, Tie Hui Hui [b]

[a]*Universiti Sains Malaysia, 11800 Penang, Malaysia*
[b]*SEGi College Penang, 10200 Penang, Malaysia*

Abstract

The purpose of this study is to investigate the influence of learning style preference on learning C++ language, especially in terms of the quality of codes written and the number of errors made in programming. A total of 120 computing students were involved in this seven-week study. They were randomly assigned to either a group that received the metaphor with pair programming (MPP), to another group that received pair programming (PP) only, or to a third group that received only the metaphor instruction (MI). Participants in both the MPP and PP groups worked in pairs based on the visual-verbal learning style dimension, and those in the MI group worked individually when completing programming tasks. Two computer programming coursework were used to measure the students' coding performance. The results indicated that visual students in the MPP group performed significantly better in the quality of codes and made fewer mistakes during coding than those in the PP and MI groups. Also, the MPP method did help the verbal students to excel in C++ coding as compared to the PP and MI groups. The direct correlation was observed between their quality of codes and the number of errors. Metaphors helped the visual students in developing deeper conceptual understanding by linking the known to the newly acquired concepts. In pair programming approach, the verbal students developed better understanding through peer discussions. The understanding of individual learning style is needed in order to enhance coding performance as robustness of a programme is correlated with the accuracy of statements in the programming.

Keywords: learning style; computer programming; metaphor; pair programming

1. Introduction

Software is a set of instruction that makes the computer system operates in specific ways. These instructions are mostly written in certain defined programming languages, i.e., Java, ISP.net and C#. Programming is an essential component of computing curriculum for students who undertake computing courses. In many colleges, C++ language is usually introduced to the first year computing students. They are frequently associating the language as a difficult module to learn. As programming demands complex cognitive skills, the mental model of these novices is not as comprehensive as compared to the experience programmers. As they usually find it difficult to understand the abstract concepts, they tend to make basic syntax mistakes and become frustrated trying to make the codes executable. This indicates that the mental model which includes problem solving, creativity, analytical, intuition and

* Corresponding Author : Irfan Umar. Tel.: +604-6535230
E-mail address: irfan@usm.my

insight (Garrison & Archer, 2000) are needed in learning C++. Thus, it is common to see students with better developed mental model outperform their peers in the course and others who performed poorly or failed in programming courses will eventually drop out from the computing programme. These who continue will often opt for career path that does not require programming.

Likewise, educators involved in delivering the abstract concepts to the first year computing students are also having difficulties in helping the students to understand the basic syntaxes. In line with Miliszewska and Tan (2007), the complex cognitive skills such as planning, reasoning, problem solving and analytical thinking are the prerequisites for learning programming. Furthermore, problem solving skills which include reasoning and analytical thinking are required to analyze a given problem scenario. In this aspect, educators are persistently encouraged to innovate and seek for alternative instructional strategies that could assist the students to better understand the C++ concepts.

Metaphor is a figure of speech with an implicit association, in which the meaning for one thing is usually applied and related to another (Cazeaux, 2007). It is a high level abstract concept used to assist the students to apply as a reference for linking their existing knowledge to the newly introduced C++ syntax. By connecting the known knowledge to the abstract concepts, it helps them to develop deeper mental images to reason about the novelty.

Pair programming (PP) is an approach where two persons work together in solving a given scenario. It has been applied in software development industry to increase productivities on programmers (Beck, 2005). Educators believe that it has educational benefits in improving students programming performance when incorporating it effectively into the teaching environment. The use of cooperative learning through PP which consists of students with different learning style may influence the students' proficiency in learning the language that somehow will be reflected in their coding performance.

Similar to any types of learning, the programming learning process is associated with the use of memory. This memory is categorized into the short term memory or the sensory memory, working memory and long term memory. Meanwhile, the software development process is drawn on the system development methodology. As such, developing an executable programme is perceived as a learning process. Based on the given problem scenario, the problem solving process takes place at the sensory memory before progressing to the working memory. Figure 1 shows the flow of learning to programme that links with the use of memory.

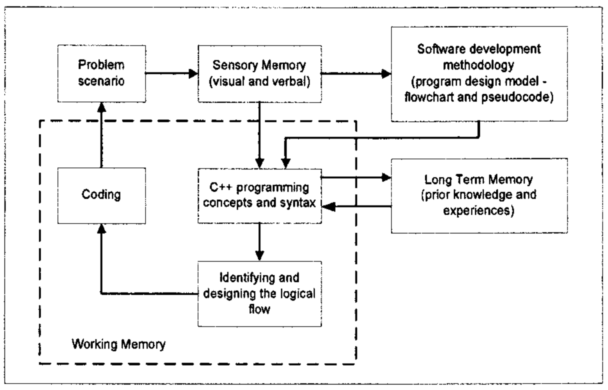
Problem scenario
Sensory Memory (visual and verbal)
Software development methodology (program design model - flowchart and pseudocode)
Coding
C++ programming concepts and syntax
Long Term Memory (prior knowledge and experiences)
Identifying and designing the logical flow
Working Memory

Figure 1: Programming learning model

In this study, metaphor as visualization representation is used to utilize the sensory memory; and pair programming as cooperative learning approach is used to provide verbal students with the opportunity to form deeper conceptual understanding of these novel concepts as they remember better through discussion with their peers. Thus, the understanding of learning style on individual learning process may enhance the students' coding performance. The visual-verbal style dimension is an important factor to be considered in classroom delivery as most of the learning materials are presented in text-based forms. Therefore, understanding learning style is important as it may influence how learning environment should be designed and how students learn best in programming.

Learning style has been defined as a preferred learning approach that individual responds to and uses stimuli in the context of learning. It is a group of attributes and behavior that establish the way of learning preference. For Kolb (1994), learning style is the way in which students perceive, process, store and recall attempts of learning. Meanwhile, Vermunt (1992) claimed that the combination factors such as cognitive, affective and psychological have contributed to individual differences in terms of their views and attitudes towards a situation. Likewise, the way they learn also varies with their learning preferences. These styles have been acknowledged as the prime construct which serve as relatively stable indicators of how students respond to the learning environment (Deborah, 2005). Nevertheless, with variation, these stable indicators somehow may change from one learning environment to another as the students adapt to the learning approach, progress and respond to the programming problems. An individual's learning style is strongly influenced by the way he or she approaches a learning task (Riding & Rayner, 1984). Empirical research reported that students consistently achieve better result in academic performance when novel concepts presented matched their preferred style (Pallapu, 2007). This study focuses on the visual-verbal learning dimension defined in the Felder and Silverman's learning style mode (Felder & Silverman, 1988) as most students undertaking science education are visual learners (Richardson, 1984), while most lectures are in verbal form – the information presented is predominantly auditory.

2. Research Question

In this study, three primary research questions were used to address and test the hypotheses:

1. Are there any significant differences in terms of (a) the quality of code written and (b) the number of errors made amongst the visual students who received (i) the combination of metaphor and pair programming (MPP), (ii) the pair programming (PP) and (iii) the metaphor instruction (MI) only?
2. Are there any significant differences in terms of (a) the quality of code written and (b) the number of errors made amongst the verbal students who received (i) the combination of metaphor and pair programming (MPP), (ii) the pair programming (PP) and (iii) the metaphor instruction (MI) only?
3. Is there any direct correlation between the quality of codes written and the number of errors made in the programming?

3. Research Methodology

The purpose of this study is to investigate the effects of learning style on the students' coding performance in terms of the quality of codes produced and the number of errors made in the programming assignments. It aims to examine whether the visual-verbal learning dimension could be the moderating factors when instructional strategy such as (i) metaphors as visualization techniques, and (ii) PP as cooperative learning, are used in learning C++ language. It also aims to investigate the correlation of quality of codes with the number of errors in the written programmes.

A 3 x 2 factorial design was used to examine the independent variable (instructional method – MPP, PP or MI) on the two dependent variables (quality of code written and the number of errors made) with learning style (visual and verbal) being the moderating variable. The content of the course identified for this research was the basic concepts of C++ language. A group of 120 first year computing students from a private institution in Malaysia participated in this seven-week experimental study. They were enrolled in a three-year Bachelor of Science in Computing (majoring in programming) program. These first year semester one students from three intact classes were randomly assigned to either in the MPP, PP or MI treatment group. Two experimental groups, the MPP and PP each consisted of 40 students and 38 students respectively; and the control group (n = 38) received the MI treatment. The students who received the MPP and PP treatments, worked in pairs based on the learning style preference, in that each pair consisted of one visual and one verbal learner. They were expected to work cooperatively in solving the given programming assignment questions. However, those receiving the MI treatment worked individually.

The Felder and Soloman (2002) Index of Learning Styles Questionnaire (ILSQ) was used to identify the preferred learning style of the participating students. In this study, only 11 items measuring the visual and verbal learning styles were used. The students are required to respond to the style that they prefer during learning. The students who responded mostly "a" on the learning styles reference in the ILSQ instrument were classified as visual learners and those who responded mostly "b" were classified as verbal learners. Meanwhile, two computer programming coursework (CPC) were used to measure the students' coding ability.

All treatment groups received program flowchart and Pseudocode in understanding the logical flow of C++ programming. During the practical sessions, the students were then instructed to convert these logical flows into workable C++ statements. In addition, those taught in the first treatment group (MPP) received metaphor and pair programming instructional strategy in enhancing their learning of the abstract concepts. The students in the second treatment group (PP) were exposed to pair programming instructional strategy in solving the given programming problems. However, the control group received the metaphor instruction only teaching method, in explaining the novel concepts of C++ programming.

The students in both the experimental groups were paired and each member of the pair was randomly assigned with a role, either as driver or navigator. Each pair consisted of one visual learner and one verbal learner. The explanation regarding the roles (driver or navigator) of each member in the pair was administered to both the MPP and PP groups. On every programming problem, they were persistently required to cooperate on the same design, algorithm, coding and testing. The role between the driver and the navigator was switched periodically.

During this seven-week study, the computer programming coursework (CPC) that consists of two programming assignments were used to examine the students' coding performance on both theory and practical knowledge of the basic C++ concepts. Their coding performance (i.e., quality of codes written and the number of errors made) was measured based on the average scores obtained from the two programming assignments. For each assignment, they were given a duration of three weeks to complete. The criteria to evaluate the quality of codes written were based on (i) the use of programming constructs in the statements, (ii) the number of coded lines and (iii) the logical flow of the programming codes. Meanwhile, the total number of errors made in program was based on the number of mistakes identified in programming assignments submitted.

4. Research findings

Throughout the seven weeks of learning C++ language, all participants submitted their two programming coursework (CPC). The students' coding performance was based on the quality of codes developed and the number of errors made in the two coursework or tasks. MANCOVA was conducted to examine the initial differences amongst the visual and verbal students taught in the three different instructional methods. In order to ensure that the participants were homogenous in their programming knowledge prior to the treatment, a pre-test score was used as the covariate. The descriptive statistics are indicated in Table 1, while Table 2 indicates the post-hoc test analysis. Meanwhile, Figure 2 and Figure 3 indicated that the better the quality of codes is, the fewer the mistakes encountered during programming execution as the software reliability rate is higher.

Table 1: Descriptive statistics for the scores between the visual and verbal students

Dependent Variable	Learning Style	Group	N	Mean	S.D.
Quality of codes	Visual	MPP	24	73.41	6.70
		PP	22	65.50	5.13
		MI	22	70.87	8.91
	Verbal	MPP	20	75.72	11.04
		PP	16	68.02	6.48
		MI	16	70.65	7.39
Number of errors made	Visual	MPP	24	6.59	2.47
		PP	22	9.31	3.00
		MI	22	9.31	3.68
	Verbal	MPP	20	5.98	3.68
		PP	16	6.77	3.74
		MI	16	10.27	3.28

Table 2: Summary of post-hoc test scores between the visual and verbal students

Dependent variable	Learning Style	Groups	Mean Diff.	p-value	Results
Quality of codes	Visual	MPP vs PP	7.90	0.00	Sig.
		MPP vs MI	2.54	0.28	Not Sig.
		PP vs MI	5.56	0.02	Sig.
	Verbal	MPP vs PP	7.70	0.00	Sig.
		MPP vs MI	5.08	0.06	Not Sig.
		PP vs MI	2.63	0.33	Not Sig.
Number of errors made	Visual	MPP vs PP	2.72	0.01	Sig.
		MPP vs MI	2.54	0.01	Sig.
		PP vs MI	0.18	0.86	Not Sig.
	Verbal	MPP vs PP	0.79	0.49	Not Sig.
		MPP vs MI	4.29	0.00	Sig.
		PP vs MI	3.50	0.00	Sig.

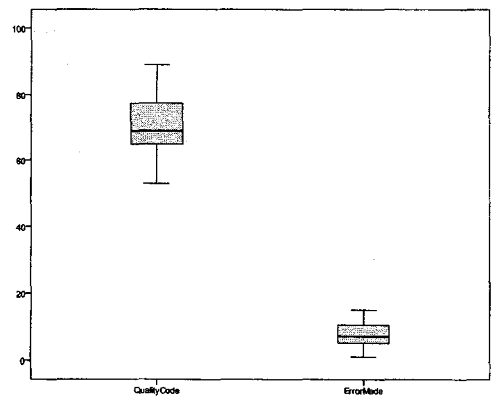

Figure 2: Correlation between the quality of codes and errors made in programming

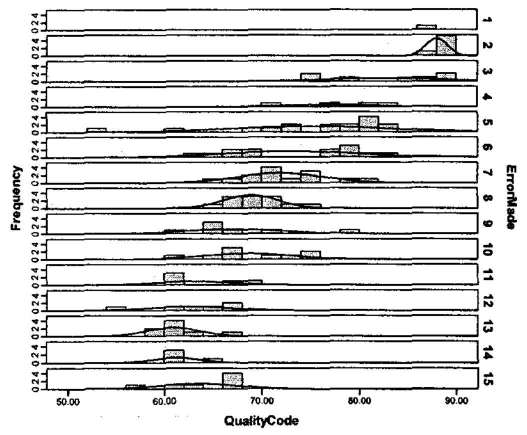

Figure 3: Quality of codes and errors made in programming by frequency

The MANCOVA result indicated a significant difference in the quality of codes written between the visual and verbal students taught in the three treatment groups ($F_{5,117} = 4.31$; p = 0.00). Similarly, there is a significant difference in the number of errors made between the visual and verbal students who received the different treatment methods ($F_{5,117} = 5.13$; p = 0.00). Therefore, these findings have rejected both the first and second hypothesis. The post-hoc test was conducted to further investigate the differences (Table 2) amongst the visual students who received the MPP, PP and MI methods as well as to those verbal students who were taught in the three different methods. The analysis also revealed a significant correlation between the quality of codes and the number of errors made in the CPC.

The post-hoc test result (Table 2) indicated a significant difference in the quality of code written between the visual students taught in the MPP group and those in the PP groups, with the former performed significantly better than the latter group ($\overline{X}_{\text{vis\_MPP}} = 73.41$; $\overline{X}_{\text{vis\_PP}} = 65.50$; Mean diff = 7.90; p = 0.00). A similar significant difference in the quality of code written was also observed between the visual students in the PP group and those in the MI group (Mean diff = 5.56; p = 0.02) with the MI group outperformed the PP group ($\overline{X}_{\text{vis\_MI}} = 70.87$; $\overline{X}_{\text{vis\_PP}} = 65.50$). However, there was no significant difference in the quality of codes written for the visual students in the MPP and those in the MI group.

In terms of the number of errors made in the CPC, the post hoc results revealed that the visual students in MPP group significantly made fewer mistakes in the coding than their peers in the PP group ($\overline{X}_{\text{vis\_MPP}} = 6.59$; $\overline{X}_{\text{vis\_PP}} = 9.31$; Mean diff = 2.72; p = 0.01). A similar significant difference was also observed between the visual students of MPP and those in the MI group (Mean diff = 2.54; p = 0.01), with those in the MPP produced less errors than their counterparts in the MI group ($\overline{X}_{\text{vis\_MPP}} = 6.59$; $\overline{X}_{\text{vis\_MI}} = 9.13$). However, no significant difference between them in the PP and MI groups was observed as these visual students performed equally good in constructing programme statements.

There were no significant differences in the quality of code written between the verbal students in MPP and those in the MI group as well as between those taught in the PP and MI groups. Only a significant difference was observed between the MPP and PP groups (Mean diff = 7.70; p = 0.00), with the verbal students in the MPP group performed significantly better than those of the PP group ($\overline{X}_{\text{ver\_MPP}} = 75.72$; $\overline{X}_{\text{ver\_PP}} = 68.02$). For the number of errors made in the programming coursework, the post-hoc test result in Table 2 also indicated a significant difference between the verbal students taught in the MPP and MI groups (Mean diff = 4.29; p = 0.00), with the MPP group making fewer mistakes in coding than the MI group ($\overline{X}_{\text{ver\_MPP}} = 5.98$; $\overline{X}_{\text{ver\_MI}} = 10.27$). Likewise, there was also a significant difference between the PP and MI groups (Mean diff = 3.50; p = 0.00), where the PP verbal students made significantly less errors in programming than their peers in the MI group ($\overline{X}_{\text{ver\_PP}} = 6.77$; $\overline{X}_{\text{ver\_MI}} = 10.27$).

The analysis also indicated a direct correlation between the quality of codes produced and the number of errors made in the coursework (Pearson correlation: -0.76, p-value: 0.00). This would mean that better quality of codes produced when lesser errors were found in the programme (Figure 2). Also, coding performance is related to the quality of program (has zero or lesser mistakes). Likewise, the graph in Figure 3 revealed that the frequency of errors made was also associated with the programme quality. Thus, hypothesis three was rejected.

5. Discussion

The research findings indicated that the differences on coding performance amongst the visual and verbal students between the three instructional methods were significant, both in the quality of codes written and the number of errors made in the C++ coursework. The analysis results also shown a significant correlation between the quality of codes produced and the number of errors found in the program statements.

Further analysis revealed that the visual students taught in the MPP method performed significantly better than those taught in both PP and MI for the coding performance in terms of quality of codes and errors made. Regardless of instructional methods, by adopting metaphor, the visual students taught in the MPP and MI groups produced equal quality of codes compared to their counterparts in the PP group. However, for the quality of codes, the finding showed that only the verbal students in the MPP group performed significantly better than those in the PP group. Also, the verbal students taught in the MPP and PP methods made fewer mistakes while coding as compared to those in the MI group. Based on the instructional methods used in delivering the C++ contents, the individual learning style preference has influenced the way the students learn. Therefore, lecturers should be mindful of the instructional strategies used in class lectures, and as well as to take note of the individual learning mode in order to have significant outcome on programming performance.

For visual students, the finding indicated that the use of the metaphor significantly influenced their coding performance in terms of quality of codes written and the number of errors made. Specifically, the visual students in the MPP group (receiving a combination of metaphor and pair programming) performed significantly better in coding performance than their peers in both the PP and MI groups. It shows that, metaphor aided the creation of memory images of the new concepts being introduced, and it further showed the connection between the existing knowledge and the newly introduced knowledge. This visualization technique helped the visual students in building clearer "mental model" of the novel concepts to reason about abstract situations as compared to those in the PP group (Cazeaux, 2007; Flanik, 2008). For the number of errors made in the programming coursework, there was a significant difference in errors made between visual students in the MPP and PP, as well as between those in the MPP and MI groups. Since the students in the MPP group had to work in pairs, they applied metaphors to finding solution through discussion. While working in pairs, the visual students used metaphors to connect the novel concepts with their prior knowledge that form new ideas for sharing. Discussion of opinions with their partners thus promote better problem solving skills, higher order thinking skills; and further improves their attitude towards learning C++ language (Mercado, Andrade & Reynoso, 2008; Meseka, Nafziger & Meseka, 2010).

For the verbal students, the finding revealed that the students taught in the MPP method significantly outperformed their peers who were taught in the PP group. However, it indicated that verbal students taught in both MPP and MI methods as well as those received the PP and MI methods produced equal quality of codes. Working in pairs to solve C++ tasks, the MPP method provides better programming conceptual understanding as the visual representation builds new concepts from the existing knowledge. By connecting concrete images with text information, metaphors develop clearer "mental schemes" of the novel concepts for problem solving as compared to these verbal students taught in the MI method. However, the insignificant difference in quality of codes written by the verbal students in MPP and those of MI only indicate that working in pairs did not significantly influence their quality of coding. In other words, among the verbal students, pair programming did not really assist their learning as metaphor has helped them to perform. In line with Pallapu (2007) and Fenrich (2006), metaphor is a high level abstract concept and verbal students learn better through discussion. Thus, the new concepts in graphical presentation gained during discussion in the auditory form would probably be incomplete and difficult for the verbal students to build complex mental schema. This is due to the fact that verbal students taught in the MPP method have to learn both abstract concepts, i.e., metaphor and C++ simultaneously. This is the possible reason why verbal students in both the MPP and MI and also those in both the PP and MI method performed equally good in creating quality of codes. The findings reveled that pair programming approach helped to reduce the number of errors made in the programme. In particular, the verbal students in the MPP and PP groups which highlight on social interaction have assisted them effectively by explaining the logical concepts used in solving the C++ tasks to their peers (Gillies, 2007). By giving constructive criticism to their pair members, they were able to crosscheck each other's codes. In turn, it minimizes the number of mistakes made in coding. Through peer interaction, the discussions aided the students to ask "why" and "how" on the programming statements that somehow reduce mistakes and increase quality in codes as well as increase self confident level in programming. Thus, the robustness of software is very much related to the quality of codes and the number of defects identified in it.

6. Conclusion

The study has emphasized the importance of understanding the individual learning style while mastering the C++ language. Meanwhile, the use of alternative instructional strategy such as the combination of metaphors with pair programming has shown improvement in coding performance amongst the visual and verbal students while learning the C++ language. Learning style is the preferred mode in which an individual student perceives and processes information. Metaphor as visualization technique allows visual students to build deeper mental model by connecting their existing knowledge with the newly introduced abstracts. On the other hand, pair programming encourages the verbal students to engage in peer discussions as they learn best through verbalizing their thoughts. With this, the better the quality of codes developed, the fewer the mistakes found in the programming codes. Thus, the students' coding performance is correlated with the accuracy of programme written in C++ language. In conclusion, knowing individual learning style preference helps lecturers to stimulate the liking for learning the language as they can comprehend the novel concepts without much frustration. It will be a fun learning journey when students are able to understand the concepts being introduced easily. These fundamental syntaxes are the stumbling block for those students who plan to make software engineering development as their career. So, it is suggested the lecturers should incorporate different presentation strategies in delivering the abstract concepts for promoting a fun and stress free learning environment.

References

Beck, K. (2005). *Extreme programming explained: Embrace change.*(2nd edition), Boston: Addison-Wesley.

Cazeaux, C. (2007). *Metaphor and continental philosophy: From Kant to Derrida.* New York: Routledge.

Deborah, G. (2005). Meeting differing learning styles of non-traditional students in the second language classroom. *Journal of College Teaching and Learning, 2*(8), pp. 1–7..

Felder, R. & Silverman, L. (1988). Learning styles and teaching styles in engineering education. *Engineering Education, 78*(7), pp. 674–681.

Felder, R.M. & Soloman, B.A. (2002). *Index of learning styles questionnaire.* Retrieved January 08, 2011, from http://engr.ncsu.edu/learningstyles/ilsweb.html.

Fenrich, P. (2006). Getting practical with learning styles in "live" and computer-based training setting. *Issues in Information Science and Information Technology, 3*, pp. 233–242.

Flanik, W.M. (2008). Conceptual metaphor and US Missile Defence: Preliminary theorizing and analysis. *The Annual Conference of the Canadian Political Science Association*, Vancouver, BC, pp. 1–20.

Garrison, D.R. & Archer, W. (2000). *A transactional perspective on teaching and learning: A framework for adult and higher education*, Oxford, UK: Pergamon.

Gillies, R. M. (2007). *Cooperative learning: Integrating theory and practice.* Los Angeles: Saga Publication.

Kolb, D.A. (1984). *Experiential learning.* Englewood Cliffs, NJ: Prentice Hall.

Mercado, C.A.A., Andrade, E.L.M. & Reynoso, J.M.G. (2008). The effects of learning objects on a C++ programming lesson. *Proceedings of the 19th Annual International Information Management Association*, San Diego, CA, October 13-15, 77–88.

Meseka, C.A., Nafziger, R. & Meseka, J.K. (2010). Student attitudes, satisfaction, and learning in a collaborative testing environment. *The Journal of Chiropractic Education, 24*(1), pp. 19–29.

Miliszewska, I. & Tan, G. (2007). Befriending computer programming: A proposed approach to teaching introductory programming. *Issues in Informing Science and Information Technology, 4*, pp. 277–289.

Pallapu, P. (2007). Effects of visual and verbal learning styles on learning, *Institute for Learning Style Journal, 1*, pp. 34–39..

Richardson, J. (1984). *Working with people.* San Francisco, CA: Associate Management Institute.

Riding, R. & Rayner, S. (1984). *Cognitive styles and learning strategies: Understanding style differences in learning and behaviour.* London: David Fulton Publishers.

Vermunt, J.D.H.M. (1992). *Learning styles and guidance of learning processes in higher education.* Amsterdam: Lisse Swets & Zeitlinger.