Students' Perception on RAPTOR Application Implementation in Problem Solving and Program Design

Mageswary a/p Muniandi[1], Nor Anisah binti Mohd Saad[2,] Jacey a/p Mariadass @ Manickam[3]

[a]magiswarry37@gmail.com, [b] anisaad@puo.edu.my, [c] jacey@puo.edu.my
[a] Jabatan Teknologi Maklumat & Komunikasi, Politeknik Ungku Omar, Ipoh, Perak, Malaysia.
[b] Jabatan Teknologi Maklumat & Komunikasi, Politeknik Ungku Omar, Ipoh, Perak, Malaysia.
[b] Jabatan Teknologi Maklumat & Komunikasi, Politeknik Ungku Omar, Ipoh, Perak, Malaysia.

Abstract

RAPTOR is an abbreviation for Rapid Algorithmic Prototyping Tool for Ordered Reasoning. It is a free graphical authoring tools designed specially to help students visualize their algorithms and avoid logic errors. RAPTOR programs are created visually and executed visually by tracing the execution through the flowchart. A flowchart is a collection of connected graphic symbols, where each symbol represents a specific type of instruction to be executed. Student's prefer using flowcharts to express their algorithms to required syntax in program design. This application provides graphical symbols that can change the way flowchart is taught in the classroom. This application was used by students to design and practice flowchart in Problem Solving and Program Design course. The purpose of this paper is to identify students' perception on RAPTOR application implementation in Problem Solving and Program Design. The study was carried out by distributing a survey in the Google-Form questionnaires to 210 students from Diploma Information Technology (Digital Technology) (DDT) in the Department of Information Technology & Communication, Polytechnic Ungku Omar. The finding shows that majority of respondents gave positive feedback to the using of RAPTOR in term of reaction of teaching and learning and impact to the skills. By using RAPTOR students were able to design a flowchart effortlessly and effectively. Based on survey outcome, conclude that RAPTOR could be used as a useful learning tool in a classroom.

Keywords: Flowchart Symbols, Programming Development Process, Flowcharting Techniques, Problem Solving in Programming, Algorithm;

1. Introduction

Problem solving and program design is a compulsory course taken by students majoring in Diploma Technology (Digital Technology), Polytechnics Malaysia. This course presents the methods in problem solving and program design. The idea learned in this course can be applied to several of the real-life problems which can be resolved by creating computer programs. It is help to define the stepwise specification of the algorithm, pseudocode and flowchart.

Since the introduction of computers in the 1940s flowchart have been a part of computer programming. In 1947 Goldstein and von Neumann [1] presented a system of describing processes using operation, assertion, and alternative boxes. They notice that "coding begins with the drawing of flow diagram." Preliminary to coding, the algorithm had been recognized and identified. The implementation of flowchart on a machine has represents a high-level description result. Although, they proposed a programming approach with numerical algorithms finally it has become a standard practice in the field of computer programming. Many of books

and reference are completely dedicated to education the flowcharting method. Farina, in his book Flow charting [2], conveys her point of view about flowcharting that it is an art which is requiring a lot of practices. Flowchart should be design before start a program coding. Program development in many professional and educational institutions was practiced this opinion. In Flowcharting Techniques [3], Bohl holds that flowcharting helps "distinguish between the procedure a computer program is written to express and the syntactical details of the language in which the program is written." She agrees the flowchart is "an important tool in problem solving" and states, "The individual who incapable to design a flowchart could be unable to get ahead a problem, problem analyse, solution decision, or problem solving."

There is a considerable evidence shows that the students in initial programming courses facing a trouble in applying idea of writing code learning in the concepts inherent in the field and computer science [4]. Flowchart can be very supportive for visual leaners for both comprehending algorithms and writing [5]. The outcomes of several years of implementing flowcharts in learning algorithms and programming is shows that they were frequently rejected by the learners. The main reason for this was the fact that designing and specifically modifying flowcharts using pencil and paper is an impractical, tiresome process and time-consuming for beginners [5]. Furthermore, the paper format flowchart is static and cannot provide any assist for understanding the dynamic nature of program implementation and the control structures [6]. However, all obstacles in creating flowchart can be solved by the using of RAPTOR tools.

RAPTOR is an iconic programming environment, designed specifically to help students visualize classes and methods and limit syntactic complexity. RAPTOR application was created a visually flowchart using a grouping of symbols.

Based on writers' experience, even though instructor's effort to attract leaners' attention on the extra basic method of algorithms lesson, they still need to spend a lot of class time on syntactic troubles that students face. Besides, Felder [7] identify that the most of students are visual learners and so that the instructors should tend to present information verbally. Between 75% and 83% of students are visual learners [8,9].

Majority of students had to learn a non-intuitive context in learning about object-orientation and algorithmic thinking in traditional programming languages which was applied to word-based nature. Based on Scanlan [10] findings the students had understood algorithms presented as flowcharts better than those presented in pseudocode. Several studies [11,12,13] had proved that students performed better in courses when trained with iconic programming languages. Since, there was a
huge figure of evidence supporting the idea that students understand programming concepts better when given in a visual representation. RAPTOR lets students to create an algorithm by joining a basic graphical symbol. Students create their class hierarchy in a symbols design and then represent method bodies as flowcharts. The resulting programs can then be run in the environment, either step-by step or in continuous play mode.

RAPTOR provides a simple graphics library, refer on AdaGraph [14]. The platform is visually displaying the position of the presently executing symbol to the content of all variables. Students are able to create an algorithm visually and they also able to view the solution of the problem visually.

Students are applying RAPTOR in Problem solving and program design course. The course is mainly trained in C++, and RAPTOR is executed to visualize how data flow is works. Students can design their flowchart in RAPTOR and then visualize the output.

1.1 Raptor Program Structure

RAPTOR is a graphical programming development environment to create a flowchart and visualize the flow of data in program design. A flowchart is a collection of connected graphic symbols, where each symbol represents a specific type of instruction to be executed. The connections between symbols determine the order in which instructions are executed. These ideas will become clearer as you use RAPTOR to solve problems. The RAPTOR Application window is shown in Figure.

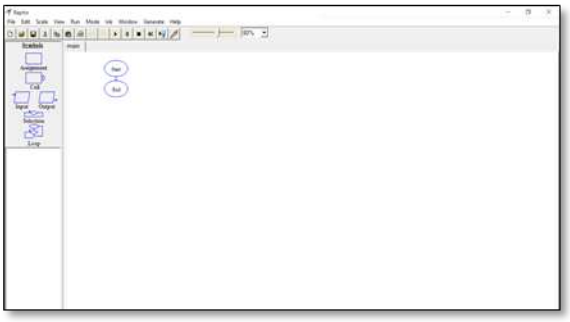


Figure 1: RAPTOR Application Window

The RAPTOR application is a set of linked symbols that represents an action to be performed. The arrows that link the symbols is to verify the flow of data is acts in the correct order. The RAPTOR application will begin at the "Start" symbol and follow the arrow to execute the application. This application will stops executing after reached at the "End" symbol. Figure 2 shown the connection symbol "Start" and "End" in RAPTOR Application.

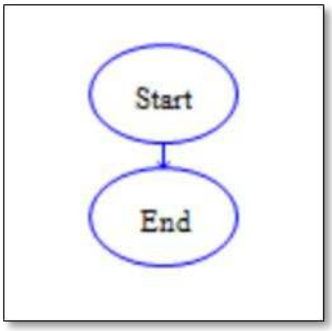
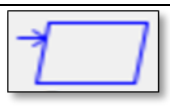
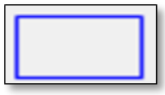
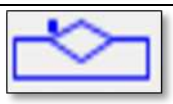
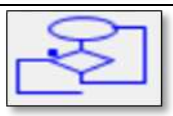


Figure 2: Start / End Connected symbol

RAPTOR has 6 unique symbols and each of them represents a specific type of instruction. They are named as Input, Output, Assignment, Call, Selection and Loop. This specified symbols in RAPTOR as shown in the Table 1.

Table 1: Symbols in RAPTOR

PURPOSE	SYMBOL	NAME	DESCRIPTION
Input data		Input box	Allow the user to insert data
Output display		Output box	Display output
Calculation / Process data		Assignment box	Perform calculation using an appropriate mathematical operation.
Processing		Procedure call	A group of instruction will execute to identified the named procedure.
Selection		Selection Control Structure	Executed statement which is match to the defined condition.
Repetition		Loop Control Structure	Controls the execution flow until match to the defined condition.

2. Objective

The main objective is to identify the student's perception on RAPTOR implementation in the problem solving and program design. We propose the graphical programming development environment to create a flowchart and visualize the flow of data in the program design.

3. Problem Statement

Majority of students face many challenges to learn designing a flowchart with traditional method or manual. They unable to enhance their skill with the paper format flowchart which is static and cannot provide any assist for the understanding. Furthermore, students face difficulty in visualizing the data flow in program design and it does caused students lost interest in learning design a flowchart.

4. Material and Method

This section presents the research goal and research questions, the tools implementation in Problem Solving and Program Design support, the data sources, the participants, the research procedure, and the method for data analysis.

4.1 Research Goal and Research Questions

The survey objective of this study was defined as the following which was applied of method Goal Question Metric (GQM) approach [15] where we first define a research goal (conceptual level), then define a set of research questions (operational level) and finally describe a set of metrics answer the defined research questions (quantitative level).

4.2 The research question (Rs) as below is

specified based on the research Objective:

R1: RAPTOR kept my concentration during develop the flowchart.
R2: RAPTOR is easy to learn for create a better program design.
R3: RAPTOR application is very effective way to design a flowchart.
R4: RAPTOR had helped me to improve my problem-solving skills.
R5: RAPTOR had enhanced my efficiency in design flowchart.
R6: I enjoyed design flowchart in RAPTOR.
R7: RAPTOR made me motivated to create a better program design.
R8: I want to continue using RAPTOR in the future.
R9: Overall, I'm pleased with the RAPTOR tool for program design.

4.3 Data sources

This section presents the outcomes from the survey to finds the students perception on RAPTOR implementation in problem solving and program design based on their learning experience. The survey consists of 9 statements reflecting the research questions R1 – R9.
The survey had used a five-point Likert scale from Strongly Disagree (1), Disagree (2), Neutral (3), Agree (4) and Strongly Agree (5).

4.4 Participants

This survey had involved 210 samples of students from Program DDT in the Department of Information Technology and Communication, Polytechnic Ungku Omar.

4.5 Data Analysis

Survey was conducted thru online using Google-Form. Data are automatically generating in the Google-Form

5. Result

This segment presents the outcomes from the survey to finds the students' perception on RAPTOR application implementation in problem solving and program design.

R1: RAPTOR kept my concentration during develop the flowchart.

Diagram 1 represents the descriptive statistics and the outcomes for the survey item related to concentration. The statistics proves 89.5% students agree that RAPTOR application kept on their concentration during develop the flowchart.

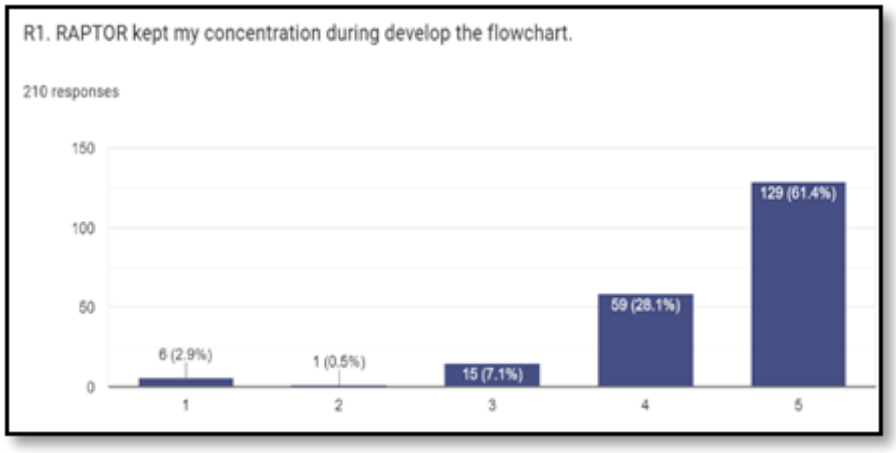


Diagram 1: Result on concentration

R2: RAPTOR is easy to learn for create a better program design.

Diagram 2 represents the descriptive statistics and the outcomes for the survey item related on easy to learn. The statistics proves 90% students agree that RAPTOR Application easy to learn for create a better program design.

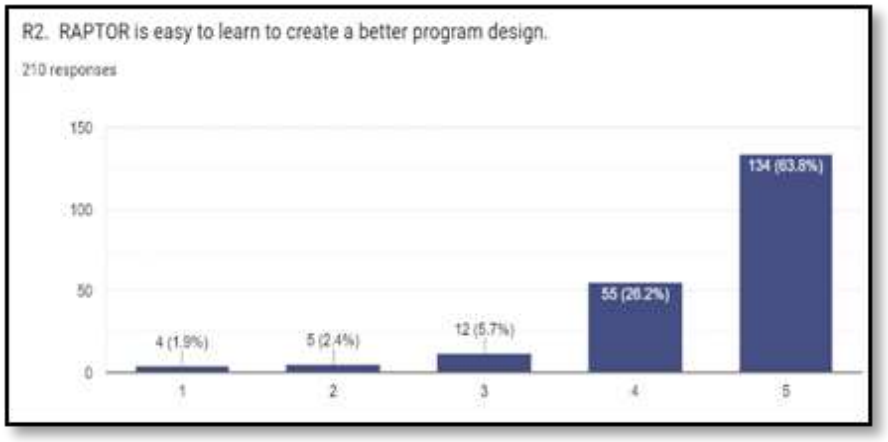


Diagram 2: Result on easy to learn

R3: RAPTOR application is very effective way to design a flowchart.

Diagram 3 represents the descriptive statistics and the outcomes for the survey item related on effectiveness. The statistics proves 89.5% students agree that RAPTOR application is very effective way to design a flowchart.

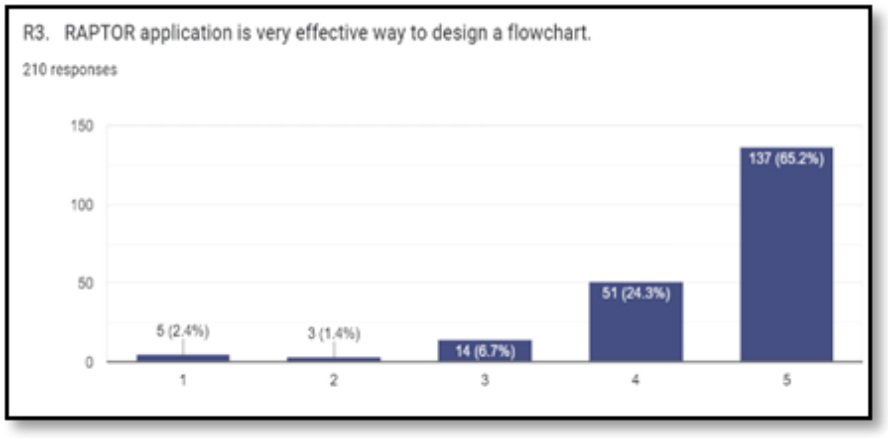


Diagram 3: Result on effectiveness

R4: RAPTOR had helped me to improve my problem-solving skills.

Diagram 4 represents the descriptive statistics and the outcomes for the survey item related on improvement. The statistics proves 86.2% students agree that RAPTOR application had helped me to improve my problem-solving skills.

Diagram 4: Result on improvement

R5: RAPTOR had enhanced my efficiency in design flowchart.

Diagram 5 represents the descriptive statistics and the outcomes for the survey item related on enhancement. The statistics proves 90.5% students agree that RAPTOR application had enhanced their efficiency in design flowchart.

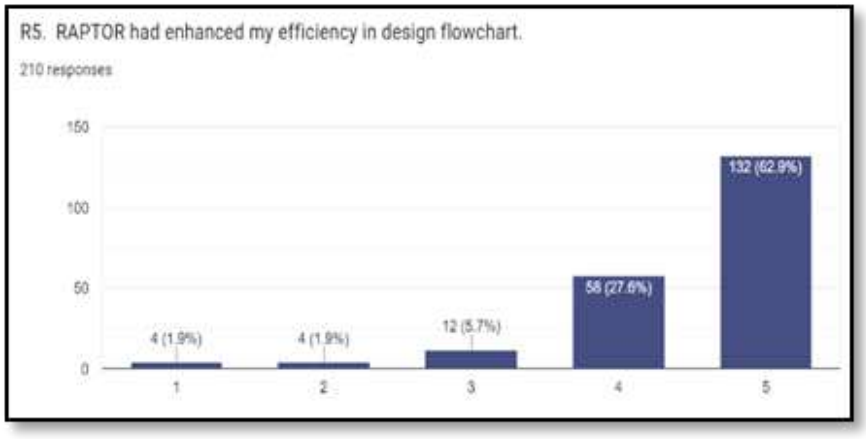


Diagram 5: Result on enhancement

R6: I enjoyed design flowchart in RAPTOR.

Diagram 6 represents the descriptive statistics and the outcomes for the survey item related on enjoyment. Based on the statistics proves 87.2% students agree that they enjoyed design flowchart in RAPTOR Application.

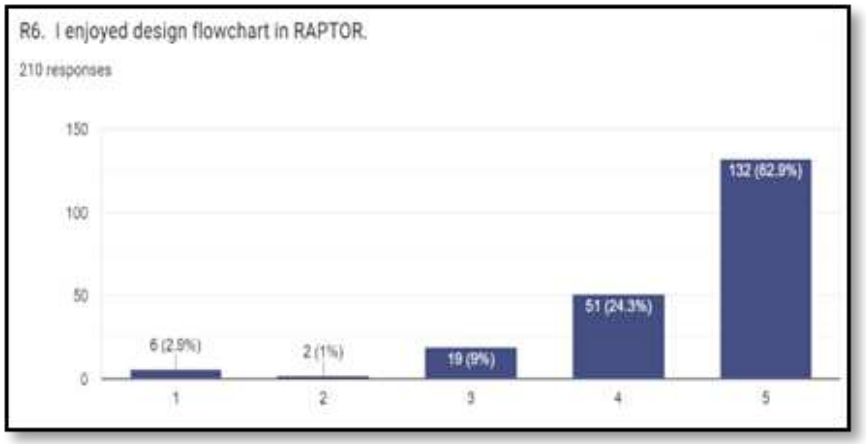


Diagram 6: Result on enjoyment

R7: RAPTOR made me motivated to create a better program design.

Diagram 7 represents the descriptive statistics and the outcomes for the survey item related on motivation. Based on the statistics proves 87.1% students agree that RAPTOR application kept on their concentration during the lecture.

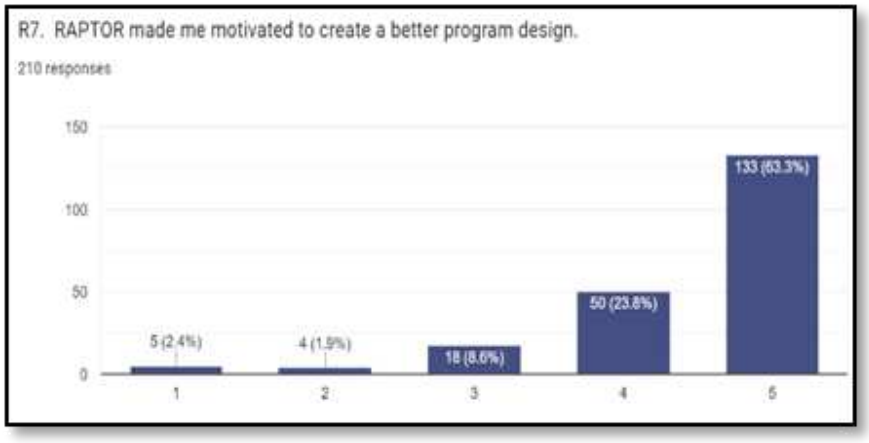


Diagram 7: Result on motivation

R8: I want to continue using RAPTOR in the future.

Diagram 8 represents the descriptive statistics and the outcomes for the survey item related on attainment. Based on the statistics proves 84.8% students agree that they wish to continue use RAPTOR in the future.

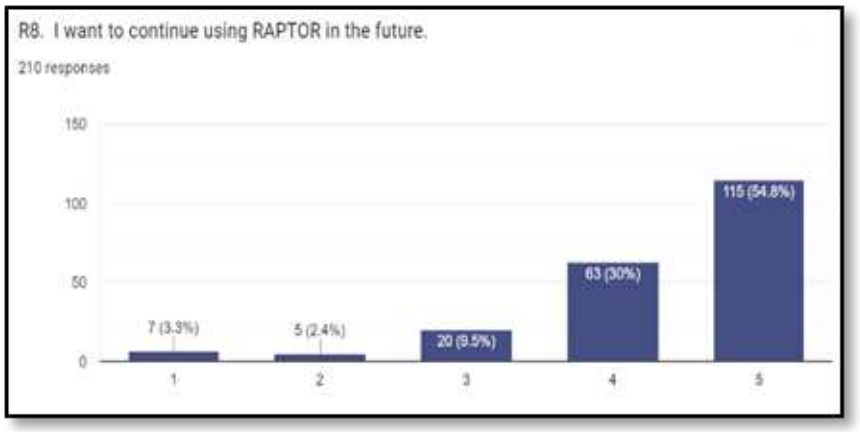


Diagram 8: Result on attainment

R9: Overall, I'm pleased with the RAPTOR tool for program design.

Diagram 9 represents the descriptive statistics and the outcomes for the survey item related to the satisfaction. Based on the statistics proves 85.8% students agree that they are satisfied with RAPTOR application as a tool for creating a flowchart in Problem Solving and Program Design course.

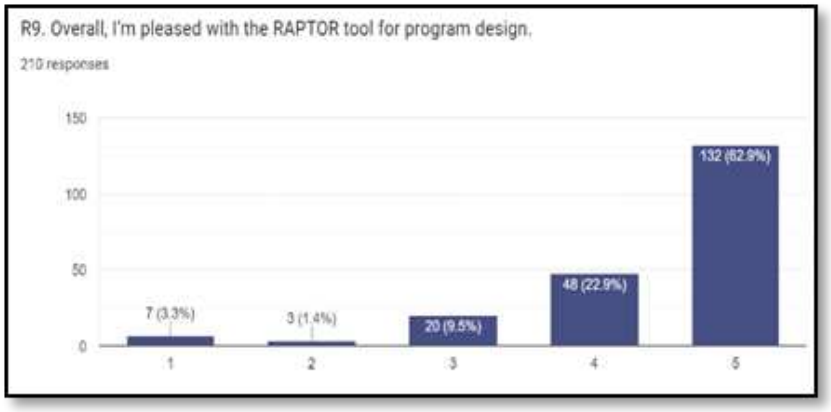


Diagram 9: Result on satisfaction

6. Discussion and Suggestions

The constraint of this study is not the experimental research. Therefore, future research would be to test two different students' groups which is using RAPTOR application and not using RAPTOR application on continues assessment to compare their result. Based on the surveys' findings we strongly suggest RAPTOR application use as a tool for creating a flowchart in Problem Solving and Program Design course.

7. Conclusion

The RAPTOR application has influences students completely in every aspect that had been researched in this survey. Executing RAPTOR application is an effective method to design a flowchart and helped students to improve their problem-solving skills. Based on the findings discussed earlier, it has been shown that students respond with positive feedbacks towards using RAPTOR application in creating flowchart. Students were also inspired and enjoyed each of the flowchart design learning sessions with RAPTOR. Based on the survey, we conclude that students are understood the benefits of RAPTOR application as an important tool for creating a flowchart and program design effectively. This application has been shown as a helpful choice for students to enhance their problem-solving skills.

8. References

Goldstein, H.H., and von Neumann, J. Planning, and coding problems for an electronic computing instrument, part II, vol I. Rep. prepared for the U.S. Army Ordinance Dept., 1947. Reprinted in von Neumann, J. Collected Works.

Farina, F. Flowcharting. Prentice-Hall, Englewood Cliffs, N.J., 1970, iii.

Bohl, M. Flowcharting Techniques. Science Research Associates, Chicago, 1971, p. 53.

D. Harel, "Statecharts in the making: a personal account", Commun. ACM , vol. 52(3), pp. 67-75, March 2009.

M. S. Hall, "Raptor: nifty tools", J. Comput. Sci. Coll., vol. 23(1) pp. 110-111, Oct. 2007.

B. Shneiderman et al., "Experimental Investigations of The Utility of Detailed Flowcharts in Programming", Communications of the ACM, Vol. 20, No. 6, pp. 373-381, 1977.

Cardellini, L. An Interview with Richard M. Felder. Journal of Science Education 3(2), (2002), 62-65.

Fowler, L., Allen, M., Armarego, J., and Mackenzie, J. Learning styles and CASE tools in Software Engineering. In A. Herrmann and .M. Kulski (eds), Flexible Futures in Tertiary Teaching. Proceedings of the 9th Annual Teaching Learning Forum, February 2000. http://ccea.curtin.edu.au/tlf/tlf2000/fowler.html [4]

Thomas, L., Ratcliffe, M., Woodbury, J. and Jarman, E. Learning Styles and Performance in the Introductory Programming Sequence. Proceedings of the 33rd SIGCSE Symposium (March 2002), 33-42.

Scanlan, D. A. 1989. Structured Flowcharts Outperform Pseudocode: An Experimental Comparison. IEEE Software. 6, 5 (Sep. 1989), 28-36.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., and Hadfield, S. M. 2005. RAPTOR: a visual programming environment for teaching algorithmic problem solving. In Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education (St. Louis, Missouri, USA, February 23 - 27, 2005). SIGCSE '05. ACM, New York, NY, 176-180.

Cilliers, C., Calitz, A., and Greyling, J. 2005. The effect of integrating an Iconic programming notation into CS1. In Proceedings of the 10th Annual SIGCSE Conference on innovation and Technology in computer Science Education (Caparica, Portugal, June 27 - 29, 2005). ITiCSE '05. ACM, New York, NY, 108-112.

Calloni, B. A., Bagert, D. J., and Haiduk, H. P. 1997. Iconic programming proves effective for teaching the first-year programming sequence. In Proceedings of the Twenty-Eighth SIGCSE Technical Symposium on Computer Science Education (San Jose, California, United States, February 27 - March 01, 1997). J. E. Miller, Ed. SIGCSE '97. ACM, New York, NY, 262-266.

vanDijk, J. AdaGraph. Online [July 31, 2008]. Available at http://users.ncrvnet.nl/gmvdijk/adagraph.html.

Caldiera, V. R. B. G., & Rombach, H. D. (1994). Goal question metric paradigm. Encycclopedia of Software Engineering, 1, 528-532.