I.J.Modern Education and Computer Science, **2013, 1, 14-20**
Published Online January 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2013.01.02

Improve Abstract Reasoning in Computer Introductory Courses

Aharon Yadin
Department of Information Systems, The Max Stern Yezreel Valley College (YVC), Israel
Email: aharony@yvc.ac.il

Abstract—Due to the elevated programming courses' failing rate in our department (45%) an action research was initiated. As part of this action research, that was performed during four semesters several course structures and learning tactics were examined. The evaluation methodology was simple and based only on the percentage of failing students. The success achieved was attributed to two main factors (1) using a visualization environment (Micro-world) for the whole duration of the course, which helped in understanding the more complex and abstract issues, and (2) using individual assignments that enforced better learning habits and development of individual algorithmic thinking. The paper describes the various attempts, as well as the final structure, that reduced the failing students by over 77%.

Index Terms—Algorithmic Thinking, Individual Assignments, Introductory Programming Courses

I. Introduction

This paper describes an action research for defining the right structure of an introductory programming course that will increase algorithmic thinking and by that decrease the high course drop-out rate. The research was performed in a small regional college due to an elevated percentage of failures..

Following the fast progress of technology, in recent years, and its wide integration in many human activities, education has been changed as well. From mainly a teaching discipline in the past, it transformed into an integrated learning environment that uses various technological tools and solution for enhanced understanding. As a result, education has shifted from just content delivery to a continuous process in which the students acquire facts and theories, through their own experience and build the conceptual models representing their understanding [1]. Conceptual models, sometimes referred to as mental models are considered the necessary building blocks for problem-solving skills. These skills which are a significant part of introductory courses' outcome are also required for succeeding in the modern society. The change, from teaching in which the instructor assumes responsibility for content delivery, to learning where the responsibility is transferred to the student is not new. This paradigm shift started over a decade ago and was addressed by many scholars [2], [3], [4], to name a few and was influenced by the understanding that effective learning occurs when the students construct their own knowledge. Following this understanding, at present, successful learning is viewed as a student-centered process in which students are exposed to various events, explore and enhance their experience and knowledge. This new perceived knowledge that is based on the students' own experience combined with already existing knowledge, constructs new layers of understanding that modify, renew and enhance the existing learners' conceptual models [5], [6]. With the massive technological integration in many aspects of our lives, the traditional learning environment has changed as well. Currently, technology is not confined any more only to the classroom. The wealth of available applications and the wide spread of computers made it possible to extend the learning process and provide it on demand, anytime and anywhere. The continuous process of learning that is based on adapting and enhancing one's own conceptual models occurs in a variety of learning locations and by using technology it can be even in virtual environments representing the real world.

In the second section I will discuss some of the existing learning theories in order to assess the students' learning processes. In the third section, the introductory programming courses will be described including methods for enhancing understandability. Section four describes the study and the results obtained and the last section provides the discussion and implications of the current study.

II. Learning theories

Over the years, many researchers were involved in understanding and evaluating learning and as a result many theories were developed. However, the learning theory that is widely used is the constructivism theory, which is based on Piaget's theory of children's development. According to Piaget, information and data are perceived through the various senses and maintained using "mental structures" that represent knowledge. Based on this theory, every living creature constantly compares its existing mental structures with the new received information in order to assess its validity. If the new received information makes sense, it will be integrated into the existing mental structure (or accommodated in Piaget's terms). This process of accommodation reaffirms and renews the mental

structure and sometimes it modifies and enhances it which represents learning. If, on the other hand, the new information is very different, or contradicts the mental structure, it will be discarded or changed so that it will fit the structure. If students are forced to "understand" the new information, for example as it happens by delivery of content, but if it does not fit their mental structure, they will memorize it without the proper understanding. This type of "learning" implies that it is not conceptualized and will not contribute to future problem-solving capabilities. According to the constructivism theory, learning is defined as integration of new experiences with the past mental structures. As such, learning means changing these previous models with relevant new information [7]. For the past 4 decades, cognitive researchers [8], [9], [10], [11] have distinguished between two types of knowledge: declarative and procedural. Declarative knowledge (also referred to as propositional knowledge) is defined as factual information ("knowing that"), while procedural knowledge ("knowing how") is about how to perform a specific task, or the skills required to operate in the environment. Before choosing the proper teaching mechanisms, the instructor has to define the required learning outcomes and select the proper activities that will help acquiring the two types of knowledge. The shift in the instructors' role from teaching to facilitating learning is based on the understanding that teaching is not just transmission of information to the students (declarative knowledge), but rather, it should be used to create various relevant activities that will stimulate students and help them construct their own mental models representing meaning. By using this learning theory, instead of delivering content, the instructor has to define the learning environment, including activities, methods and assignments, so that it will enable the students to acquire the required declarative as well as procedural knowledge.

The constructivist model is a learner-centred process in which the learning responsibility relies on the students. In achieving the defined learning goals, students may be involved in both group and individual learning activities. Many researchers however have reported that group learning is more successful and helps the students build their understanding faster and more efficiently [12], [13], [14]. Group learning has had many different names: peer learning, collaborative learning, team studying, collective learning, study or work group, etc. However, according to [15], regardless of the name, all of these learning methods can be categorized by three general types: (1) informal learning groups – which are formed ad hoc. This is a one-time learning session for addressing a specific issue; (2) formal learning groups – which are formed for a specific task, with a longer duration (for example a project). Such formal learning groups usually require several meetings; and (3) study teams – which are formal learning groups, working together for an even longer duration (whole semester, or the whole academic year). In many cases, study teams form a social group in which the relationships among the team extend the study sessions. However, although collaborative learning is more efficient and the study group and its social interaction form a supportive learning environment, the learning (or accommodations in the mental structures) and attaining knowledge remains an individual process. For that reason, some researchers suggest structuring courses not only on collaborative study, but on cooperative study as well. In such teams there is a greater emphasis on individual responsibility and accountability [16]. This and the introduction of technology supported collaborative learning systems that provide virtual and remote collaboration, imply that students have to be more autonomous in their learning attitude [17].

III. INTRODUCTORY PROGRAMMING COURSES

Undergraduate Introductory Computer Science (CS1) courses which represent the students' first encounter with the professional computing world are often perceived by the students as problematic based on the relatively high drop-out rates. Furthermore, the skills, both programming and problem solving, acquired by the students after successfully completing these courses are often not sufficient [18]. These students' difficulties are not a new issue and were addressed by many debates among researchers, scholars and educators. One of the explanations that was suggested for these difficulties is the high degree of abstraction and complexity required when dealing with the programming paradigm concepts [19], [20]. Other researchers suggest that the introductory courses have to only briefly address the programming concepts, and to concentrate on algorithmic thinking. This means spending more time training students on ways to find solution to problems [21], [22], instead of concentrating on the programming language itself. As such, this approach uses a higher level of abstraction, almost ignoring the specific programming language and focuses mainly on building and enhancing the capabilities required for algorithm constructing [23]. By using the constructivist theory definitions, this approach is about modifying or enhancing the mental models. This debate on the issue of defining the most successful ways to tackle the CS1 courses is fueled by the low students' enrollment which unfortunately, was not affected by the fact the market recovered from the problems caused by the burst of the dot.com bubble. The decreased interest in the CS (Computer Science) discipline [18], [24] combined with the very high (sometimes up to 50%) drop-out rates [25], [20], [26], [27] increased the urgency for various additional attempts to solve the problem.

In dealing with the students' difficulties, several researchers claim that some of the modern programming languages used for CS1 courses require the understanding and mastering of advanced concepts at an early stage of the learning process. This means that the factual information required by the CS1 courses interferes with the procedural knowledge. Students who cannot cope with this early understanding are failing the

course because they do not understand the more abstract programming concepts [23]. For addressing these difficulties, some researchers and educators started using visual environments in order to improve understanding some of the abstract concepts related to programming and problem solving. For example, visual environments are used to illustrate an abstract concept while changing it into a more concrete object. The visualization approach for enhancing students' understanding is not new and it has been used to teach children in the late 70's, for example by using LOGO [28], [29], [30], [31]. LOGO is a simple and basic programming language developed for learning by example or "discovery learning". This learning and exploration environment was designed to stimulate cognitive development and creativity. Visualization environments, tools and methodologies were later addressed as learning by example or Micro-worlds [32], [33], [34], [35]. These multimedia based Micro-worlds are small, interactive and dynamic visual learning environments which represent a conceptual model of some part of the real world. For better handling abstract issues, the model usually simplifies the real world and makes it more understandable by using a concrete visual representation and by providing various tools to explore or manipulate it [36]. The reason for implementing such mechanisms in which first children and later students could develop algorithms without the usage, or knowledge of formal programming language was explained by Eric Roberts: "In real-world programming languages like C, there are so many details that learning about them tends to dominate the first few weeks of a programming course. All too often, they become the focus of the course, and the much more critical issues of problem solving get lost in the shuffle" [37]. The learning by example puts a greater emphasis on the learning based on one's own experiences, which leads to developing the right problem solving and algorithmic thinking skills, instead of mastering the specifics of a particular programming language.

The fast technological advancements affected the visual environments as well and brought a wealth of additional new tools that were addressing the students' difficulties and were aiming to solve the problem. The new environments defined and designed a friendlier and gentler approach for teaching programming. One such environment is "Karel the Robot" [38] that was originally introduced for teaching Pascal. This is a non-threatening, visual environment with a robot living in a two dimensional world (Micro-world). The robot performs tasks that emphasize programming logic. The student instructs the robot to successfully perform some pre-defined tasks while avoiding the various obstacles presented in the world. By defining and controlling the robot activities, the student is gradually exposed to the principles of a programming language. Furthermore, the environment provides a solid foundation for developing problem solving methodologies such as logical deduction and reasoning. The Karel environment was later migrated to support additional programming language, especially Java [38], [40], [41] and Python.

IV. THE STUDY

The current action research was performed during four semesters as part of the CS1 course. The course is delivered during the first semester of the first year and represents the primary encounter students have with programming, logic and problems solving. CS1 is intended to set the foundations for the later more complex courses, however for students with no prior programming knowledge it is difficult and represents a significant challenge. Our CS1 course is concentrating on procedural programming, while the next programming courses concentrate on the Object Oriented paradigm. The first programming language, used in this is course is Python, while next programming courses use JAVA. During 2009 the course was taught on both semesters and on 2010 and 2011 only during the first semester. The total number of students enrolled is relatively small and in addition there was a large fluctuation in this number, as demonstrated by figure 1.

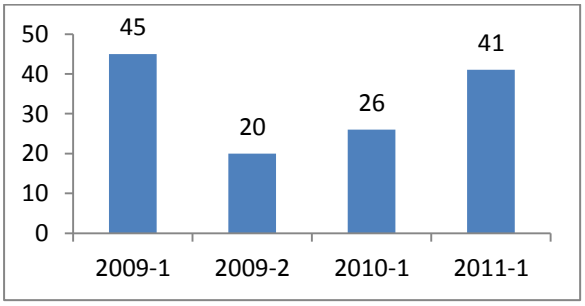

Figure 1. Number of students per semester

The problems associated with the our course were similar to problems reported by other academic institutes, i.e. a relatively high drop-out rate and the students who successfully completed the course possessed lower than expected programming and problem solving skills. Originally, the course structure was simple and consisted of three hour lecture using Python, a two hours lab exercise and an additional support course. Python is an easy-to-use interpreted language, yet powerful, portable, object-oriented and open source. It can be used for writing stand-alone programs, quick scripts, and prototypes for large applications. In using Python, the aim was to concentrate more of developing algorithms and improving problem solving skills (procedural knowledge) and less of the language syntax and constructs (declarative knowledge). The support course was included mainly for lowering the understanding barriers and helping students construct their mental models that represent knowledge. The support course was a two hour lecture and lab, using "Karel the Robot" Micro-world. The intension was to strengthen the algorithmic thinking capabilities and provide a visual environment and an easier way of understanding. This visual environment was intended mainly for the more abstract issues such as nested loops, nested conditions and recursion.

There are many academic institutes which use Micro-world environments as part of their introductory

programming courses. However, unlike other institutes that use the Karel environment mainly during the first one or two lessons and just for preliminary understanding of basic programming constructs, the structure we employed was based on a semester long usage. This way Karel was used not only for understanding the basic programming constructs, but also for visualizing some of the more complex concepts. Specially, we used the environment so that students will be able to design and check various algorithms for solving problems while evaluating and debating among themselves and in class each algorithm. Although students worked individually, the lab acted as a formal learning group during the whole semester, in which the students worked individually, but learned collectively.

Unfortunately, this course structure which was based on Python as the primary programming language supported by a semester long usage of a visualization tool, had no positive effect in our case and the percentage of the failing students was very high (43.1%). Due to these poor results an action research study was initiated. The main idea was to find the best way for teaching the course. The only dependent variable used to assess the success was the failing students' percentage. The action research study was based on 3 evolutionary course versions (Table 1) and was run during 4 semesters.

In order to affirm the results obtained using the first course structure (Python and Karel the Robot) the same structure was repeated during the second semester. Unfortunately, during the two semesters in 2009, in which this structure was employed, the failing percentages were similar and very high (43.1% and 45.8% see Figure 2). A thorough analysis which included discussions with students regarding their difficulties revealed that "Karel the Robot", which initially was considered a visualization tool for enhancing understanding, caused more confusion. The course lectures concentrated on teaching procedural programming, while Karel is using an object oriented approach. This difference not only did not provide the required assistance, but it even caused more misunderstanding. Furthermore, although the two courses (CS1 and the support course) were two parts of the same course, they were delivered by two instructors, which may have caused additional confusion. Another much more troubling issue was linked to the Karel environment that proved to be unstable. During normal work, the environment may suddenly abort, without saving the current project. In such cases, all the work performed was lost. Due to the course structure, in which the Karel environment was used throughout the whole semester, the stability issues became of a great importance, unfortunately with a negative impact. During the first half of the semester, while the examples and exercises were relatively simple, everything worked fine. However, during the second half, when the exercises became more complicated and the students had to define many new procedures the environment turned out to be unstable. This problematic behavior translated into many lost hours and turned into a frustrating issue. As a consequence some students preferred to stop using the environment, even at the expense of decreased understanding and a lower grade.

Based on Python's success in other institutes, the decision was made to continue using Python as the first programming language, and replace the supporting visualization environment. On the third semester, a second version of the course structure was employed. "Karel the Robot" was replaced by GvR (Guido van Robot) a Python based implementation of "Karel the Robot". This is an open source product that can be downloaded freely and installed on the students' computers, supporting a variety of platforms. As part of the preparations for the course a long and intense benchmark was carried and several problems that were discovered in the product were corrected. For enhancing understanding the two courses were delivered by the same instructor, which allowed for better integration between the two courses and relating smoothly from one course concepts to the other. This change was very successful and the number of failing students, in this version of the course, was reduced by 63.5%, from 45.8% of failing students to 16.7% (Figure 2).

Due to the author's experience with individual and unique assignments [42], [43] it was decided to implement this tactic as well. This was done mainly, in an effort to further reduce the failing students' percentage. The last version of the course was very similar, with only one change. The support course (the GvR Micro-world), which included several assignments and contributed 10% to the CS1 course grade was changed to use individual and unique assignments. This type of assignments is based on individual assignments, which means that the students cannot share or borrow solutions with/from their friends. Each exercise is unique, so students can only discuss among themselves the algorithms; since each student receives a different assignment one student solution is irrelevant to the other. This change was successful and reduced failing students' percentage by additional 41.6%, from 16.7% to 9.8% (Figure 2).

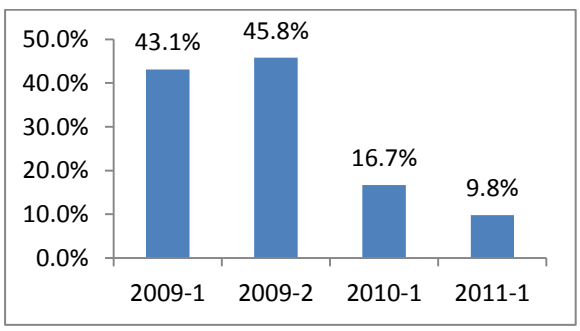

Figure 2. Failing students percentage

Table 1 summarizes the 3 versions of the course structure, including the main attributes of each version, the weaknesses and the results obtained by utilizing it.

TABLE I. COURSES VERSIONS SUMMARY

Ver.	Year	Tools	Instructors	Weakness	Failing %
1	2009	Python & "Karel the robot"	2 instructors; one for python and one for Karel	Karel stability issues that hampered usage and understanding	43.1% on 1st usage of this version, 45.8% on 2nd
2	2010	Python & GVR ordinary assignments	1 instructor for both Python and GVR		16.7%
3	2011	Python & GVR individual assignments	1 instructor for both Python and GVR		9.8%

V. RESULTS AND DISCUSSION

This paper describes an action research study that was performed in order to help students cope better with the difficulties related to introductory programming courses by improving their algorithmic thinking and problem solving skills. The original structure, which used Python and was based on a standard 3 hours lecture, followed by a 2 hours exercise and an additional 2 hours Micro-world lecture/lab was slightly modified. The last and more successful structure used same general components; however, the visual environment in the support course was replaced. In addition, the assignments as part of the support course emphasized individual learning in a cooperative environment, which added another level of success. During the four semesters of this action research, the students' failing percentage was dropped by 77.4% (from 43.1% to 9.8%).

The issues raised by this action research support similar findings presented in other papers that adding a visualization environment (Micro-world) improved the students' operational knowledge. In the first version of the course, the visualization environment was not successful; however it was related to stability issues with that environment, which lead to many students abandoning it. The net result was that the students enhanced their mental models by developing abstract knowledge related to programming concepts and algorithms for solving problems, instead of concentrating on syntax issues. This was evident, because the exam concentrates on algorithmic issues and not just syntax. Succeeding in the exam is possible only for students who understand the principles and are capable to solve problems. The use of the GvR Micro-world provided additional insight into the process. The importance of visual environments especially when dealing with abstract concepts is not new and was already addressed by many researchers (Papert, 1980; Dagdilelis and Satratzemi, 2001; Hoyles, Noss & Adamson,2002; Sarama & Clements, 2002 to name a few). However, this action research demonstrated the importance of these environments and a direct link between them and the actual CS1 course. The 77.4% improvement in the failing percentage may be attributed to the fact we used the Micro-world environment during the whole the semester, while, in many academic institutes, where Micro-worlds are integrated into the CS1 course they are being used only for the first one or two lectures.

The impact of using the Micro-world was intensified by the fact it created a semester long team based collaboration. Although each student had to work individually on his/her assignments, in the lab, there were sub-groups who worked and learned together, as was evident by the fact they used same seats throughout the whole semester. This supports similar findings by many researches that group learning helps students build their understanding more efficiently (Beckman, 1990; Cooper et al., 1990; Goodsell, et al., 1992). The lab exercises provided an additional way of collaboration, since it acted as a foundation for discussions regarding various solutions and the benefits and shortcomings of each one. The success attributed to using individual and unique assignments support similar findings and it contributed to further lowering the failing rate.

The reasons behind the fluctuations in the number of enrolled students are unknown. It may, however, be related to the high percentage of failing students. This is a relatively small regional college and the information, especially in the social networks era is spreading fast. There is some correlation between the failing percentage and the reduction in the enrolment. However, this issue will have to be monitored in the future, before it will become conclusive.

REFERENCES

[1] A. Dillon, “Knowledge Acquisition and Conceptual Models: A Cognitive Analysis of the Interface” in Diaper, D. and R. Winder (eds.) People and Computers III, Cambridge, UK: Cambridge University, pp. 371-379, 1987

[2] R. Barr, and J. Tagg, “From Teaching to Learning-a New Paradigm for Undergraduate Education”, Change Magazine, Nov/Dec, pp. 13-25, 1995

[3] S. Bell, S. and A. Lane. "From Teaching to Learning: Technological potential and sustainable, supported open learning“, Systemic Practice and Action Research, 11(6), pp. 629-650, 1998

[4] R. DuFour, R. Eaker, and R. DuFour, On common ground: The power of professional learning communities, Bloomington, IN: Solution Tree, 2005

[5] Stoica, I, Moraru, S and Miron , C. Concept Maps, A Must for the Modern Teaching-Learning Process. Romanian Reports in Physics, 63(2), pp. 567–576, 2011

[6] Lavy, V. What Makes an Effective Teacher? Quasi-Experimental Evidence, The National Bureau of Economic Research Working Paper No. 16885, 2011

[7] E. Zhi-Feng, S. Liu, C. Chi-Huang, and Y. Shyan-Ming, "Web-Based Peer Review: The Learner as Both Adapter and Reviewer", IEEE Transactions on Education, 44(3), p. 246, 2001

[8] J. R. Anderson, Cognitive Psychology and its Implications, San Francisco: Freeman, 1980

[9] L. R. Squire, Memory and Brain. New York: Oxford University Press, 1987

[10] K. Johnson, Language Teaching and Skill Learning, Oxford, Basil Blackwell, 1995

[11] J. B. Biggs, Teaching for Quality Learning at University, Second edition, Buckingham, Open University Press, Open University Press/Society for Research into Higher Education, 2003

[12] M. Beckman, "Collaborative Learning: Preparation for the Workplace and Democracy", College Teaching, 38(4), 128-133, 1990

[13] J. Cooper, "Cooperative Learning and College Teaching: Tips from the Trenches", Teaching Professor, 4(5), pp.1-2, 1990

[14] A. Goodsell, M. Maher, V. Tinto, L. Smith, & J. MacGregor (eds.) Collaborative Learning: A Sourcebook for Higher Education. University Park: National Center on Postsecondary Teaching, Learning, and Assessment, Pennsylvania State University, 1992

[15] D. W. Johnson, R. T. Johnson, and K. A. Smith, Cooperative Learning: Increasing College Faculty Instructional Productivity. ASHE-FRIC Higher Education Report No.4, Washington, D.C.: School of Education and Human Development, George Washington University, 1991

[16] M. Prince, "Does Active Learning Work? A Review of the Research", Journal of Engineering Education, 93(3), pp. 223-231, 2004

[17] R. Webster, and F. Sudweeks, "Enabling Effective Collaborative Learning in Networked Virtual Environments", Current Developments in Technology-Assisted Education, (2) pp.1437-1441, 2006

[18] U. Nikula J. Sajaniemi, M. Tedre S. Wray, Python and Roles of Variables in Introductory Programming: Experiences from three Educational Institutions. Journal of Information Technology Education, 6, 199-214, 2007 Available at http://jite.org/documents/Vol6/JITEv6p199-214Nikula269.pdf Accessed August 2012

[19] A. Robins, J, Rountree, & N. Rountree, Learning and teaching programming: A review and discussion. Computer Science Education, 13(2), 137-172, 2003

[20] L. Rich, H. Perry,& M. Guzdial, A CS1 course designed to address interests of women, 35th SIGCSE Technical Symposium on Computer Science Education (pp. 190-194). Norfolk, Virginia, USA: ACM Press, 2004

[21] A. I. Forsyth, T.A. Keenan, E. I. Organick, and W. Stenberg, Computer science: A first course, John Wiley, 1975.

[22] G. Futschek, Algorithmic Thinking: The Key for Understanding Computer Science. In Lecture Notes in Computer Science 4226, Springer, pp. 159 – 168, 2006

[23] B. N. Miller, and D. L. Ranum, Teaching an introductory Computer Science Sequence with Python. Proceedings of the 38th Midwest Instructional and Computing Symposium, Eau Claire, Wisconsin, USA, 2005.

[24] A. Radenski, "Python first": A lab-based digital introduction to computer science, 11th annual SIGCSE conference on innovation and technology in computer science education (pp. 197-201). Bologna, Italy: ACM Press. 2006

[25] M. Guzdial, Media Computation Course for Non-Majors. ITiCSE Proceedings. P104-108 .ACM. NY. 2003

[26] N. Herrmann, J. L. Popyack, B. Char, P. Zoski, C. D. Cera, T. N. Lass, Redesigning introductory computer programming using multi-level online modules for a mixed audience, 34th SIGCSE technical symposium on computer science education (pp. 196-200). Reno, Nevada, USA: ACM Press. 2003

[27] N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, et al.. Improving the CS1 experience with pair programming, 34th SIGCSE technical symposium on computer science education (pp. 359-362). Reno, Nevada, USA: ACM Press. 2003

[28] W. Feurzeig, & G. Lukas, Logo: A programming language for teaching mathematics. Educational Technology, March, 1972.

[29] G. Fischer, Material and ideas to teach an introductory programming course using Logo. Irvine, Calif.: Department of Information and Computer Science, U. C. Irvine, 1973.

[30] R. Rubinstein,. Computers and a liberal education: Using Logo at the undergraduate level. Irvine, Calif.: Department of Information and Computer Science, U. C. Irvine, 1974.

[31] A. B. Cannara, "Experiments in Teaching Children Computer Programming." Technical Report No.

271. Institute for Mathematical Studies in the Social Sciences, Stanford University, 1976.

[32] S. Papert, Mindstorms: Children, Computers and Powerful Ideas, Nova York: Basic Books, 1980

[33] V. Dagdilelis, and M. Satratzemi,. Post's Machine: A Didactic Microworld as an Introduction to Formal Programming, Educational and Information Technologies, 6(2), 123-141, 2001

[34] C. Hoyles, R. Noss, & R. Adamson, Rethinking the Microworld idea. Journal of Educational Computing Research, 27(1&2), 29-53, 2002

[35] J. Sarama, & D. Clements, Design of Microworlds in mathematics and science education. Journal of Educational Computing Research, 27(1&2), 1-5, 2002

[36] J. Hogle, Computer Microworlds in education: Catching up with Danny Dunn. (ERIC Document Reproduction Service No. ED 425738), 1995 http://www.eric.ed.gov/ERICWebPortal/contentdelivery/servlet/ERICServlet?accno=ED425738 Accessed August 2012

[37] E. S. Roberts, The Art and Science of C: A Library-Based Approach, Reading, MA: Addison-Wesley, 1995

[38] R. E. Pattis, Karel the Robot: A Gentle Introduction to the Art of Programming. John Wiley & Sons Inc. New York, NY, USA. 1981

[39] B. Becker, Teaching CS1 with Karel the Robot. Proc. ACM SIGCSE 32nd Technical Symposium on Computer Science Education, Charlotte NC. 50-54 ACM Press. 2001

[40] D. Buck, & B. Stucki, JKarelRobot: A Case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum. Proc. ACM SIGCSE 32nd Technical Symposium on Computer Science Education, Charlotte NC. 16-20 ACM Press. 2001

[41] J. Bergin, M. Stehlik, J. Roberts, R. Pattis, Karel J. Robot: A Gentle Introduction to the Art of Object Oriented Programming. Dream Songs Press. 2005. http://csis.pace.edu/~bergin/KarelJava2ed/Karel++JavaEdition.html.

[42] A. Yadin, and R. Or-Bach, Fostering individual learning: when and how. ACM SIGCSE Bulletin 40(4): 83-86 2008

[43] A. Yadin, "Reducing Students' Dropout – The Case of Individual Assignments". eLSE 2011 - The 7th International Scientific Conference - eLearning and Software for Education – Bucharest, Romania, April 2011 (2), pp 297-302

Dr. Aharon Yadin is a Senior Lecturer at the Max Stern Yezreel Valley College (YVC), Management Information Systems Department. Aharon's primary teaching areas are related to computer architectures and programming and business/management information systems issues. Prior to entering the academic world, Aharon worked in the High Tech industry. He has over 30 years of IT experience including: management, system performance analysis and enhancement, computer center and IS management, wireless networks and communication technologies and document management.

Aharon has published over one hundred papers, assays and scientific and technological reports (many in Hebrew), he is the author of 12 Hebrew instructional books. In addition, he consults the European Commission on software related projects and technologies.

