Volume 1 No.6, December 2011 ISSN 2224-3577
International Journal of Science and Technology

http://www.ejournalofsciences.org

Teaching Computer Programming in the 21st Century

[1]Mutua Stephen, [2]Wabwoba Franklin, [3]Abenga Elizabeth, [4]Kilwake Juma, [5]Ogao Patrick

[1, 2, 4] Department of Computer Science
[3]Department of Curriculum & Instructional Technology
[1,2,3,4] Masinde Muliro University of Science & Technology
[5]Kenya Polytechnic University College

ABSTRACT

Over the years, research has shown that programming has proved to be a challenging task to many. Due to this, several program visualization tools have been developed to aid in teaching programming. This study aimed at assessing the impact of using programming visualization tools in the teaching and learning of computer Programming. An overview of the tools that were used during the study is given followed by review of literature on the benefits of PV tools in teaching Programming. The study is based on Edga Dale's (1954) Cone of Experience, which forms the foundation of resource based learning theories. Literature reveals that the use of program visualization tools in teaching and learning Programming have posted positive results in various institutions. This is followed by a report of a study conducted using experimental research design approach. The same class was taught two programming introductory courses using BlueJ and Jeliot3 tools; and the performance of the students in the two courses was compared. In addition, during the classes, the covert-direct observation method was used to observe student interactions' and behaviors as they programmed and solved problems during the lessons. Results revealed that these tools if effectively used can improve on the alertness of students, interest in the subject and ultimately positive results.

Keywords: *Program Visualization (PV), Programming, Algorithm Visualization (AV)*

1. INTRODUCTION

Programming is a course in Computer Science and related fields, and some Engineering disciplines which plays a core role that is practically applicable in both academic and professional projects. However, learning to program is a challenging and complex process that requires support of proper educational tools[2]. Various research findings have showed that there is a universal problem in teaching and learning programming [15] [17] [10]. This is clearly evident on novice students [4] who are learning the basics of programming, as well as in advanced programming concepts which most students tend to avoid taking as they are offered as electives.

This trend may be attributed to the subject's abstractness [9] and that the students lack concrete model in their everyday life to handle the concepts at hand [15]. Nevertheless, the pedagogical approaches used in teaching the programming courses may be a contributing factor towards the poor performance and understanding of this crucial subject. To improve on the classroom experience, several program visualization (PV) tools have been developed over the last decade. Their trials in assorted Universities and other institutions have posted positive results as they have shown significant improvement on performance of various purported 'weak' students [8] [20]. Perhaps it is because students can understand the overall program as a whole as well as its finer details [14]. Further, the students can later revisit these tools and play animated codes execution thus enforcing understanding.

PV tools are flexible and didactic instruments that can greatly enrich the experience of both the learner and the teacher. This paper explores the use of PV tools as an advanced and modern technological approach towards attaining the pedagogical objectives effectively and efficiently. The work presents and compares the results of teaching two consecutive programming courses using the traditional approach versus the PV tools' approach. The results of the work show a significant improvement of students taught using this approach and thus enforce the findings of other researchers.

The cone of experience and resource based learning

The cone illustrated by figure 1 is a graphical representation of a theory proposed by [3] about the effects of the use of various instructional materials in education. Dale theorized that learners retain more information by what they 'do' as opposed to what is 'heard', 'read', or 'observed'. The theory states that the amount of learning experienced depends on the sense(s) involved in the learning process. According to this theory, a hierarchy exists between the concrete and the abstract levels of thinking, which has an analogy in the understanding, and use of instructional materials. The 'learning by doing' propagated by Dale has become known as 'experiential learning' or 'action learning' that forms the basis of realism theories in education, which underline the need for concrete examples as a basis for the acquisition of concepts.

The cone of experience is important to programming because it can be used as a basis for the selection and use of instructional materials and techniques for teaching. This study took the stand that instructional materials and tools such as PV tools, are vital in the teaching of programming, and that the amount of learning

experienced depends on the type of senses involved in the learning process and how actively the students participate in the learning process. This study therefore sought to find out the difference in performance of students taught programming using PV tools.

In support of the realism theories, Sampath, *et al* (1990:36) explain that human beings derive all experiences from three main sources, namely direct sensory contact which involves doing; pictures and other forms of representation of objects which involve observing; and oral or printed words which involve symbolizing. Direct sensory contact is the most effective hence teachers of young learners must learn to provide experiences that give such contact. PV tools provide such contact.

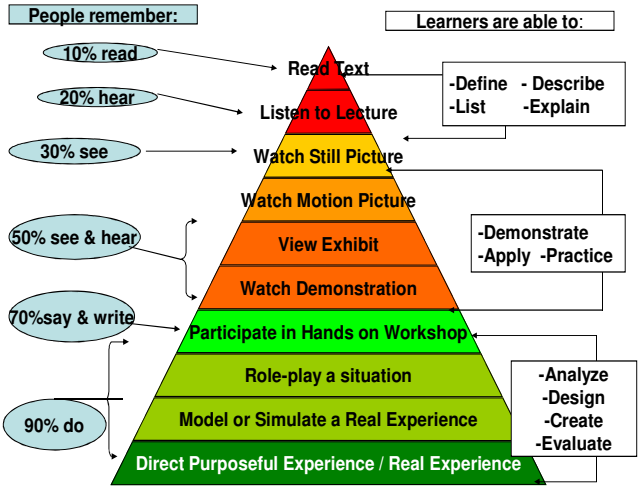

Fig. 1 Adapted from [3] and Modified by [1]

Dale's cone of experience is a development based on earlier theorists. For example, Rosseau, the originator of the child-centered approach presented the idea that a learner's capability and interest should be considered in learning, as opposed to the vast amount of information others want him/her to acquire. Froebel (1887), Pestalozzi (1901) and others supported the learner-centered approach leading to child-centered education that brought forth learner centered methods and materials in teaching. The child-centered education theory changed the methods and materials used in education to introduce learning by experience, discovery methods, freedom in learning, and the Dalton plan. All these have their place in the cone of experience. Behaviorist theories also influenced and formed the basis of Dale's cone of experience. These theories hold the view that environment influences behavior, leading to techniques such as programmed learning with books or machines, with branching technique involving the stimulation of interest and reinforcement. Behaviorist approach to education is in line with the 1974 UNESCO recommendation on teaching for world citizenship which can be achieved through an education system that offers its learners freedom and autonomy in learning through state of the art educational technology and the application of the cone of experience.

The cone of experience classifies instructional materials according to their effectiveness in communicating ideas. Thus, verbal symbols are placed at the top of the cone because they are believed to be least effective in teaching. Visual symbols are also abstract representations, but they bring in the new dimension of vision. PV tools combine visual symbol that are enhanced in colour and animation as well as audio, and are highly interactive, providing the student substantial control of the learning process.

The rest of this paper is organized as follows: Section 2 is a preview of related work while the methodology employed in the study is described in Section 3. Section 4 is an overview of PV as an emerging teaching technique and a description of the two tools used during the study whereas Section 5 presents the findings and discussion. The conclusion is detailed in Section 6 and finally, section 7 is a summary of the proposed future work in the field of PV.

2. RELATED LITERATURE

Visualization tools have been developed to augment the learning process. These may be in form of teaching aids, toys, models and/or software systems all aimed at enhancing the learning process [21]. Proper and consistent use of PV tools has proved to be very instrumental in honing the skills of programming. This is because students can use and reuse the tool(s) outside the classroom severally in order to master the principles behind the execution of some code section. The animations provided by the tools can be replayed until the concepts that were unclear in their minds are well formed and understood. This assertion has been proved by various researches that have shown that the use of visualization enhances learning.

[22] found that students who actively used the Jeliot program animation system improved their learning results compared to a control group that did not use the PV tool. [19] showed that program visualization increased the attention of students to the material being taught. In their study, [15] confirm that PV;

"… enhances students' learning regardless of previous programming experience. Moreover, it seems that the tool benefits novice learners more than learners with some previous experience."

In their research, [23] reported that in a programming course they taught 55% of the students did not pass the exam. After the first revision and move to Python programming language which they taught using a PV tool they had developed to enhance motivation, only 38% did not pass the exam. This significant improvement was attributed to the use of Turtlet a PV tool they had developed. [24] developed Alice, a PV tool whose focus

was introduction to object oriented programming concepts with an objective of attracting more girls to computer courses. The usage of the tool in classroom saw an improvement on the number of girls enrolling for the courses.

However, in spite of these positive postings, the integration of any tool in teaching requires a careful and elaborate consideration to ensure that the aid tool is not a point of confusion. Proper tool selection, understanding and timely integration must be embraced for effective learning process. If so, the PV tools can provides visual cues, graphical techniques and at times audio means to facilitate student understanding and reasoning [25]. This perhaps confirms that a picture is worth a thousand words.

3. METHODOLOGY

This study was conducted in Masinde Muliro University of Science and Technology within a span of eight months. The experimental research design approach was used as is appropriate for fact finding [6]; where the same class was taught two programming introductory courses. The first course was a second semester first year (CS121- Procedural Programming) course which was taught without applying the use of any PV tool. The second course was a first semester second year course (CS210 – Object Oriented Programming) which was taught with integration of two PV tools. The performance of the two courses of the same class was compared. In addition, during the classes, the covert-direct observation method was used to observe student interactions' and behaviors [5] as they programmed and solved problems in classroom.

To ensure that the results gathered were reliable and valid, this research was conducted without the knowledge of the population. This implied that they behaved in their natural way. Further, the examination results were handled by the researchers hence no modification could arise since the target population did not have access to them.

4. PROGRAM VISUALIZATION

In Computer Science and related fields, various toys and software visualization tools have been developed [21] over years to aid in teaching. This has led to various research fields within Software Visualization to which Program Visualization falls. Closely related to PV is

Algorithm Visualization (AV) which focuses on teaching computer algorithms which is a field in Computer Science. Unlike PV which focuses on visualization of data and code for human understanding AV visualizes the higher level descriptions of the software [27]. However, the scope of this study was limited only to PV.

PV tools focus on explaining the execution of computer programs (a set of instructions) hence facilitating an access to dynamic and usually hidden processes during program run-time [26]. This gives a student the ability to see and perceive what happens on a statement that he/she writes in some programming language. Further, they awaken and strengthen students' interest in the course as well as aid instructors in achieving the set objectives. Some examples of these tools include; Jeliot [11], Jeroo [4], Jive [13], Alice [24], Ville [15], and BlueJ [7] among others. All these tools are freely available and can be downloaded on the various websites as shown.

During this study, two tools were used to aid in teaching the CS210 course. These were chosen after a careful consideration of the various tools and their ease of use. The tools are briefly described in the following subsections.

4.1 JELIOT3

JELIOT3 was as a result of improvement on various versions which were released before it. It was developed at the University of Joensuu [11] to aid learning and teaching procedural and object oriented (OO) programming [12]. It allows students to write down their own programs and execute them during which they can watch the step by step execution animations. The tool contains an easy to use interface and is designed to ensure consistency in all animations among other goals. It is also extensible as its source code is open thus allowing modifications. Fig. 2 is a sample interface of a program in execution.

The Section labeled A provides a space where one type the code. On clicking the play button below it, the code is executed and animations tracked from Section B. if the program has some output, it is displayed at the section labeled C.

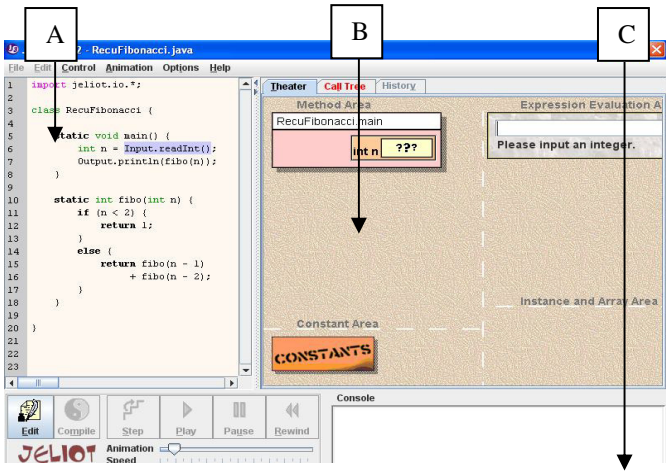

Fig. 2: JEliot User Interface

4.2 BLUEJ

BlueJ [7] has received a wide acceptance in teaching JAVA object oriented features to novice students and professionals. It is developed to gradually introduce object oriented concepts in which students can model real world scenarios using pictorials. As this happens, the tool

ISSN 2224-3577

generates an appropriate dummy code which aids them in grasping the language's syntax.

> "Advantages of the BlueJ environment include its graphical representation of the classes and objects within a project, and the simplicity with which students can interact with them through a sequence of pop-up menus." [8].

It uses the unified modeling language which is a standard and universally accepted representation hence forming a very good tool to introduce object oriented concepts like data abstraction and encapsulation, inheritance and polymorphism, message passing among others which are normally difficult for students to grasp. However, it does not contain the animations of the source code like the JELIOT3. Fig 2 shows a print screen of its user interface which is simple and straight forward.

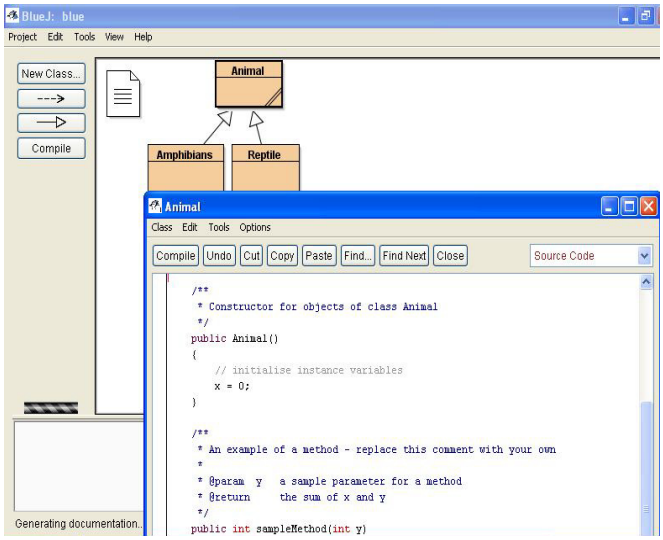

Fig. 3: BLUEJ Interface

5. FINDINGS AND DISCUSSION

Generally, from observations made during the study, several students had difficulties in grasping the key programming concepts. Most of them found it hard to develop simple computer programs that especially in using looping and decision-making structures. This made it impossible to fully cover the intended course work comprehensively due to the time spent in teaching some of the programming concepts. Though the examination was theoretical, majority of the students did not do well. The graph below shows the results that were posted for the first programming course taught. The grading system used was;

Table I: Grading System

Marks	Grade
70 – 100	A
60 – 69	B
50 – 59	C
40 – 49	D
Below 40	F

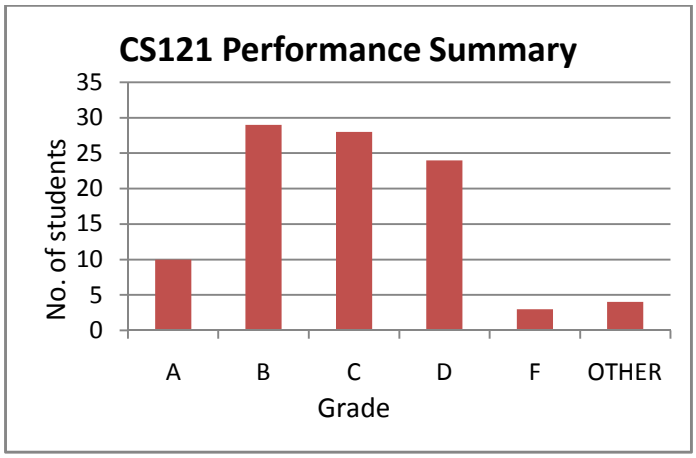

Fig. 4: CS121 Performance Summary

From this performance summary, only 10% of the candidates managed to get a score of 70% and 30% percent of the class scoring grade B. Cumulatively, close to 59% of the class scored 55 marks of the expected 100 and below. On the extreme side, 3% of the students failed to raise a score of forty (40). The grade labeled 'OTHER' implies that the students did not complete the course either because they dropped out or did not sit for the final examination.

The second course considered (CS210 – Object Oriented Programming) was introductory in that the students were being introduced to object oriented features which they had not tackled before. The course started with a theoretical introduction to object oriented concepts. The students were then introduced to BlueJ which they used to model real life problems. From observation, even before moving to the specific Java Integrated Development Environment (IDE), the students could write some programs on paper using the Java syntax. This made it easier to for the course lecturer to teach the practical programming in real language. The students looked enthusiastic and enjoyed programming.

JEliot3 was included in the middle of the course to explain the critical concepts of control structures which involved looping and decision making. This proved to be an enjoyable task to most of the students who made a follow up to have the tool installed in their PCs. The animations made it easier to explain the concept of looping using a tool whose animation speed is controllable and can be paused and replayed posting consistent results. This was performed for three weeks and learning continued using the real Java environment.

Ultimately, the results of these two courses offered to the same group of students using different approaches were compared. The results were as follows;

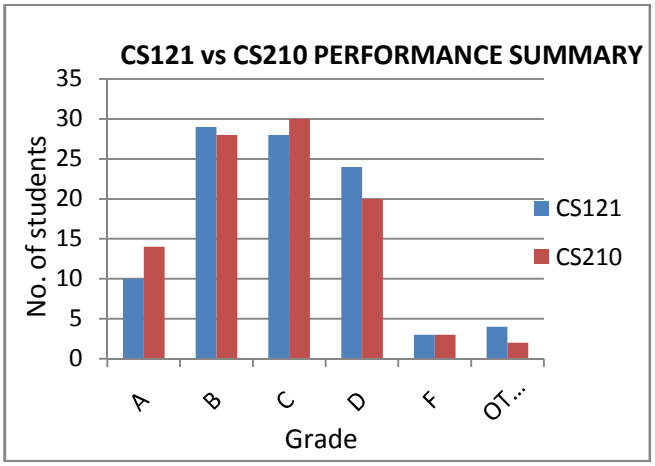

Fig. 5: CS121 vs CS210 Performance Summary

Comparing the results, the number of students scoring grade A increased to 13 compared to 10 students in prior course; while those that failed remained the same over the two successive years. Assuming the 50% score to be the pass mark for an average student, there was 6% class improvement on overall. This was a positive and significant improvement for the class signifying that a pool of the students grasped the concepts of this technical and challenging course. These results confirm that program visualization tools have proliferated educational benefits as also argued by [18].

6. CONCLUSION

From these results, we conclude that program visualization is an appropriate technique to teach programming in the current times. It is clear that if the tool(s) had been introduced earlier, perhaps the students would have had a much more positive attitude than they had. It is thus timely that all instructors find ways and means in which to make learning an interesting and enjoyable venture to many. There is need for several teachers to gradually begin to use new educational materials not only to improve the focus and effectiveness of instruction, but also to stimulate a strong desire from students of the course material so obtained which allows a more complete understanding of the topics addressed in the classroom. If well embraced, increased motivation of students will definitely be evident, and thus improved performance.

However, we also note that the introduction of new technologies in education requires a careful assessment of the impact it has on the teaching and learning itself: the need for easy accessibility and easy use of these tools must take priority over everything. The teachers need to choose well and understand a tool well since it can impact negatively on his/her performance and hence on the quality of his teaching if such tools fail in the classroom.

7. FUTURE WORK

The passion and strong interest in new advanced techniques of instruction remains a priority to all. More research is thus required to provide guidelines to both teachers and students on choosing the right tool. This is because of the proliferation of the program visualization tools which are freely available and all geared towards a common objective. There currently exists no comprehensive classification of these tools hence its time consuming to identify what tool to use. Further, there is need for research and development of program visualization tools in diverse programming languages which are currently on offer in the institutions of learning to address the shortage in some languages. With these done, teaching/learning computer programming will be an interesting experience for both learners and instructors.

REFERENCES

[1] Abenga, E., (2010), The Adoption and Use of New Educational Technologies in Training Teachers of English in Primary Teachers' Colleges in Kenya, *J-STEM Vol 3 Nos 1&2. Masinde Muliro University of Science and Technology*

[2] Bednarick, R., Moreno, A., Myller, N., & E, Sutinen (2005), Smart Program Visualization Technologies: Planning a Next Step, *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*

[3] Dale, E. (1954), Audio-visual methods in teaching, *revised edition. New York: A Holt-Dryden Book, Henry Holt and Company*

[4] Dorn, B. & Sanders, D. (2003), Jeroo: A Tool for Introducing Object-Oriented Programming, *ACM SIGCSE'03, February 19-23, 2003*

[5] ETA Ev. Briefs (2008), Data Collection methods for Program Evaluation: Observation, *Evaluation Briefs No. 16 Dec. 2008 pp.1-2*

[6] Kerlinger, F. N. (1973), Foundation of Behavioral Research, *New York: Holt, Renehart and Winston*

[7] Kolling, M., & Rosenberg, J. (1996), An Object-Oriented Program Development Environment for the First Programming Course, *SIGSE Bulletin 28 (1), 83-87, 1996*

[8] Kouznetsova S, (2007), Using Bluej and Blackjack to Teach Object-oriented Design Concepts In CS, *Department of Computing Science Sam Houston State University, Journal of*

Consortium for Computing Sciences in Colleges April 2007 (pp. 49 -55)

[9] Lahtinen, S. P, Sutinen, E. & J. Tarhio (1998), Automated Animation of Algorithms with Eliot, *Journal of Visaual Languages and Computing, 9(3): pp.337–349*

[10] Milne, I., Rowe, G. (2002), Difficulties in Learning and teaching Programming - Views of Students and Tutors, *Education and Information Technologies, 7(1), pp. 55-66*

[11] Moreno, G., A. (2005), The Design and Implementation of Intermediate Codes for Software Visualization, *Master's Thesis, University of Joensuu 2005*

[12] Moreno, A, Myller N, & Sutinen, E. (2004), Visualizing Programs with Jeliot, *ACM AVI Visualization, Master's Thesis, University of Joensuu 2005 '04, May 25-28, 2004*

[13] Paul, G. & Bharat J. (2005), Methodology and Architecture of JIVE*, ACM pp. 95 – 104, 2005*

[14] Pich, C., N, Lev & George, G. (2008), Visualization of Exception Handling Constructs to Support Program Understanding, *SOFTVIS 2008, Herrsching am Ammersee, Germany,September 16–17, 2008*

[15] Rajala, T., L. Mikko-Jussi, K. Erkki, Salakoski, P. (2007), VILLE – A Language-Independent Program Visualization Tool, *Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), Koli National Park, Finland, Vol. 88 November 15-18, 2007*

[16] Robins A, Rountree, J. & Rountree, N. (2003), Learning and teaching programming: A review and Discussion, *Computer Science Education;, Vol. 13-No. 2, pp. 137–172, 2003*

[17] Soloway, E. & Spohrer, J. (1989), Studying the Novice Programmer, *Lawrence Erlbaum Associates, Hillsdale, New Jersey. Pp. 497*

[18] Urquiza-Fuentes, J. and Angel Vel azquez-Iturbide, J., (2009), *Survey of Successful Evaluations of Program Visualization and Algorithm Animation, ACM Transactions on Computing Education, Vol. 9, No. 2, Article 9, June 2009* [19] Ebel, G. & Ben-Ari, M. (2006), Affective effects of program visualization*, Proceedings of the 2nd International Computing Education Research Workshop (ICER'06). ACM Press, 1-5*

[19] Kasurinen J., Mika, P. & Uolevi, N. (2008), A Study of Visualization in Introductory Programming, *PPIG, Lancaster 2008*

[20] Trees Fran (2008), Toys, Techniques and Tools for Teaching Computer Science, *CS & IT Drew University*

[21] Bassat L, R., Ben-Ari, M., & Uronen, P. A. (2003), The Jeliot 2000 program animation system, *Computing Ed. 40, 1, 15–21*

[22] Kasurinen J., Mika, P. & Uolevi, N. (2008), A Study of Visualization in Introductory Programming, *PPIG, Lancaster 2008*

[23] Kelleher C. & R. Pauschy (2006), Lessons Learned from Designing a Programming System to Support Middle School Girls Creating Animated Stories, *Proceedings of the Visual Languages and Human-Centric Computing IEEE Computer Society Washington, DC, USA 2006*

[24] Petre, M. (2010*),* Mental imagery and software visualization in high-performance software development teams*, Journal of Visual Languages and Computing 21 (2010) 171–183*

[25] Bednarick, R., Moreno, A., Myller, N., & E, Sutinen (2005), Smart Program Visualization Technologies: Planning a Next Step, *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies (ICALT'05)*

[26] Price, B.A., Baecker, R.M., & Small, I.S (1998), A Principled Taxonomy of Software Visualization, *Journal of Visual Languages and Computing* ***4****(3), pp. 211-266*