The ITB Journal

Volume 8 | Issue 1 | Article 2

2007

Using Extended Analogy to Teach Fundamental Computing Concepts

Damian Gordon

Recommended Citation

Gordon, Damian (2007) "Using Extended Analogy to Teach Fundamental Computing Concepts," *The ITB Journal*: Vol. 8: Iss. 1, Article 2.
doi:10.21427/D73T72
Available at: https://arrow.tudublin.ie/itbj/vol8/iss1/2

Using Extended Analogy To Teach Fundamental Computing Concepts

Damian Gordon
eLearning Research Group, School of Computing,
Dublin Institute of Technology, Kevin Street, Dublin 8, Ireland.
Damian.Gordon@comp.dit.ie

Abstract

Using analogies has long been used to help explain complex ideas in teaching. One of the most important ideas that a computing student must understand is the basic architecture of a digital computer. Since the late 1960s the standard teaching approach used to explain computer architecture is the Little Man Computer (LMC) Paradigm. This research seeks to expand upon this standard explanation through the use of a relatively recently developed technique called Extended Analogy. Extended analogy seeks to combine regular analogy with exaggeration or dramatic emphasis. In this case the LMC paradigm is extended by use of Samuel Beckett's play "Krapp's Last Tape". The students were presented with a range of exercises based on the play, including creative tasks such as writing a poem, writing a dramatic scene, and drawing pictures. They were designed to take cognisance of learning styles and, in particular, a new model of learning styles developed by the author. The complete process was assessed qualitatively and received a mostly positive reaction.

Keywords

Innovation, extended analogy, the Little Man Computer (LMC) Paradigm, learning styles.

1 INTRODUCTION

Learning to program can initially be very difficult for students. Two barriers to learning this skill are, firstly, the abstract nature of programming, concepts such as variables, data types, arrays, etc. have no real world counterparts, and secondly, programming requires an exactness of specification that contrasts strongly with the flexible nature of the English language. One teaching approach that has existed for many hundreds of years is teaching by analogy. This approach allows students to develop new ideas and new understandings based on existing and familiar understandings. This means that complex technical material can be taught through the use of analogies which can be more interesting ands captivating. This approach is widespread in teaching programming, examples of it can be found in [1] and [2]. A recent approach to this same problem is teaching by Extended Analogy [3]. Extended analogy combines analogy with high drama to make the lesson more memorable. This approach can help the student to remember the concept being taught using exaggeration or dramatisation [4,5].

In this research the concept being taught is the basic architecture of a computer, essentially how a computer computes. The architecture is explained using a Samuel Beckett play "Krapp's Last Tape" to illustrate the key operations of the system. The students are given a number of exercises to reinforce the ideas being taught and are given a questionnaire to provide them with the opportunity for feedback.

2 THE LITTLE MAN COMPUTER PARADIGM

Most computer architectures conform to the so-called von Neumann Architecture. This means that they execute programs by accessing both instructions and data on the same storage device, which makes the computer a very flexible device. By treating the instructions in the same way as data, a computer can easily change the program, and can do so under program control.

The computer performs the following sequence of steps:

1. Fetch the next instruction from memory at the address in the program counter
2. Decode the instruction using the control unit
3. Increment the program counter
4. The control unit commands the rest of the computer to execute the instruction
5. Go to step 1

This is the Fetch-Decode-Execute (FDE) cycle. These computers are known as Stored-Program Computers since the separation of storage from the processing unit is implicit in this model.

A conceptual device or thought experiment to teach this architecture was developed by Stuart Madnick of MIT in the 1960s and is called the Little Man Computer (LMC) Paradigm [6,7]. The LMC Paradigm consists of a room with a 'Little Man' who simulates the operations of a computer. The room has an array of locations that store information (and instructions), an input and output tray, and a calculator. The analogy between the LMC and real computers is not perfect, but this approach is a simple and powerful conceptual model which allows students easy entry level to the basics of computer architecture.

3 KRAPP'S LAST TAPE: THE PLAY

"Krapp's Last Tape" by Samuel Beckett has been called both "very straight-forward and accessible" [8] and "Beckett's most perfect piece of writing for the theatre"[9]. It was written in 1958 for actor Patrick Magee whose sonorous voice had captured Beckett's imagination after hearing him read from "Molloy" and "From an Abandoned Work" on BBC radio. The play was written in English and was provisionally entitled "Magee's Monologue". It is considered by many one of the most important plays written in the twentieth century.

The play concerns Krapp, a 69-year-old man who is seen throughout the play residing in a single room. He records his memories regularly, and although all the recordings are made by the same person, the effect of listening back on older tapes gives the illusion of distinct identities. Each recording is logged by Krapp in a ledger for easy later retrieval. His initial interactions with his recordings are quite jovial; both Krapp and his past voice laugh together. As the play continues things become darker with Krapp expressing distaste for his younger self.

4 KRAPP'S LAST TAPE: AN EXTENDED ANALOGY

"Krapp's Last Tape" centres on the idea of a single individual in a single room, recording and accessing information (on his tapes) by indirect means through his ledger. This clearly maps onto the LMC paradigm. The Little Man becomes Krapp, the LMC room becomes Krapp's Den, and the stored information becomes Krapp's Tapes. The analogy of the LMC paradigm becomes the extended analogy of "Krapp's Last Tape" which will hopefully be more dramatic and memorable for the students. Therefore, in an effort to assist the students understanding of the basic architecture of a digital computer, the students were required to read the play and undertake exercises based on it. Another benefit to this approach is that it exposes the students to the extremely important concept of indirection. Because Krapp does not directly search through his boxes of tapes, but rather locates the address of the appropriate tapes using his ledger ("…box three, spool five…"). This same indirect approach to accessing information can be seen throughout computer systems, for example, in pointers, which store the address of a particular variable (but not its value). Other examples of indirection can be seen in hashing, look-up tables, and overlay networks.

Additionally the students were required to read a text which was difficult to understand at first but persistence helps make it much clearer. This is an important skill to develop for an I.T. professional, since they are often required to develop software for an organisation whose system may be expressed in its own business-specific terminology. This can seem like a daunting task at first, but a little practice can be of great benefit. Thus, these exercises were an ideal opportunity to begin developing the necessary patience and thoroughness required.

5 FACTORS CONSIDERED IN DEVELOPING EXERCISES

The exercises were designed to take into consideration the individual strengths and weaknesses of the students as learners. Learning depends on a large number of factors, many of which are specific to the individual person. The individual learning style represents the particular set of strengths and preferences that an individual or group of people have in how they take in and process information [10]. The idea of a learning style is certainly a controversial one, and one which there is little agreement over. Everything from the exact definition of learning styles to the very existence (and stability) of learning styles has been hotly debated. The most commonly accepted definition is by Keefe [11] who defines it as "the composite of characteristic cognitive, affective, and physiological factors that serve as relatively stable indicators of how a learner perceives, interacts with, and responds to the learning environment."

5.1 The Nexus Model of Learning Styles

The exercises were designed to encourage students to employ a range of skills and to help scaffold and encourage a creative approach to computing. The design of the exercises was based on a four-quadrant learning styles model developed by the author [12] called the Gordon-Bull learning styles model. The individual learning style represents the particular set of strengths and preferences that an individual or group of people have in how they take in and process information. The learning styles model in this case divides students into four categories:

Alpha (α) Style - these are the practical students, they like to understand how topics being taught relate to the real world. They also like topics which are clearly structured.

Beta (β) style - these are the discussion-oriented students, they like to work in groups, and derive most benefit from intrapersonal learning.

Gamma (γ) style - these are the holistic students, they prefer an overview of topic before delving into specific detail. They are also highly imaginative individuals and bring this resource to the learning process.

Delta (δ) style - these are the analytical students, they are dispassionate students who like to focus on concepts, theories and logic.

5.2 Learning Styles Exercises

Based on these categories, a series of exercises were devised to compliment each student's style on one exercise and to challenge them on three others. The objective of which was to help create a more rounded learner in terms of being able to approach a problem from a range of different perspectives, and also to equip the students with the skills they will need to develop to be an industrially-oriented practitioner. The ultimate goal of this approach is to help the students develop as reflective practitioners [13]. That is, they must learn to employ metacognition in decision-making and problem solving, which means that when working through a problem they draw on previous experiences and test various possible solutions until they resolve the issue. The exercises were as follows:

- ***Exercise 1: Chronology of Krapp***

  After asking the students to read the play, the first exercise that they were required to undertake was an analysis of the text to create an internal chronology of the play. This exercise was designed to foster logical thinking and an analytical approach to problem-solving. This exercise will appeal to the delta-style students (the practical learners) who will have a natural ability with it but all students will need some skill in this way of thinking.

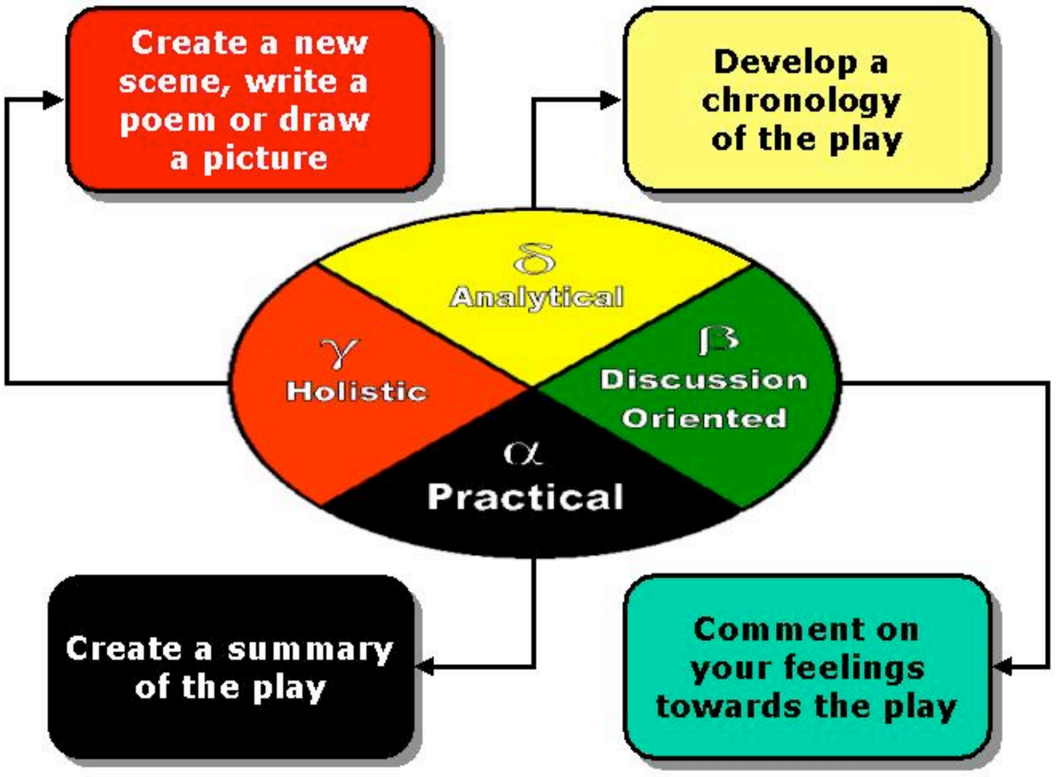
Create a new scene, write a poem or draw a picture
Develop a chronology of the play
δ Analytical
γ Holistic
β Discussion Oriented
α Practical
Create a summary of the play
Comment on your feelings towards the play

Figure 3. Learning Styles and Exercises

- ***Exercise 2: Krapp in Brief***

  The second exercise that the students undertook was to create a summary of the play. This required that they go back and re-read the play slowly, and distil the salient points of the play. This is a very significant skill for an I.T. professional to have, the ability to recognise the most important points in a specification. This exercise appeals to the alpha-style students (the practical learners) but again all students need some skill in this way of thinking.

- ***Exercise 3: Krapp's Lost Tapes***

  The next exercise was a creative one; to either write an additional scene for the play, write a poem about the play, or draw a picture of Krapp. This exercise required that the students access their creative abilities. This creativity has to be tapped into every time a computer program is written, it is a fundamental skill of any programmer. The gamma-style students (the creative learners) will enjoy at these kinds of exercises but all students need some skill in this way of thinking.

- ***Exercise 4: Krapp in Action***

  The final exercise the students undertook was simply to reflect on the work that was done and to identify what effect, if any, it had on them. The students were given a questionnaire to help in this process, to help them become reflective practitioners. Examples of questions included:

  - *Had you read any Beckett works before you started the course?*
  - *Will you read any more?*
  - *What did you think of Krapp's Last Tape?*
  - *Did you find the exercises interesting related to Krapp's Last Tape?*

The beta-style students (the intrapersonal learners) will excel at these kinds of exercises but all students must develop skills at this.

These exercises also map onto various stages of a software development lifecycle. The first exercise (the analytical one) is like the analysis of requirements of a system. The second exercise (the practical one) is like the design phase of a system. The third exercise (the creative one) is like the development phase of a system. And the final exercise (the reflective one) is like the implementation and maintenance phases of a system. As state above, it is the ultimate goal of this approach is to help the students develop as *reflective practitioners*.

6 RESULTS AND CONCLUSIONS

As expected the outcome varied greatly from student to student. Those who excelled at one exercise often did less well in other exercises. So, for example, the first exercise which required the creation of a chronology resulted in a detailed analysis by some and a very superficial analysis by others. Equally with the practical and creative exercises different students did better at each one.

The final questionnaire provided the students with the opportunity to comment on how they felt the entire process went for them, and reactions were somewhat mixed. Students with prior experience of programming (and, therefore, a range of preconceived notions) were less willing to participate in the process, whereas those with no prior experience were more willing to immerse themselves in the process and thus achieved more significant gains. The final outcomes for the 2004 - 2005 K268/1 Software Development class are available at **http://www.comp.dit.ie/dgordon/KLT**.

Examples of positive student feedback included:

- *"Good practice for understanding specifications"*
- *"It was a bit of a laugh and brought the class together"*
- *"It made me think outside the box"*

Other students did not feel that they benefited from the process, so much so that they gave very negative feedback on the process, which will guide the next iteration of this research process. A breakdown of questions answered is presented below.

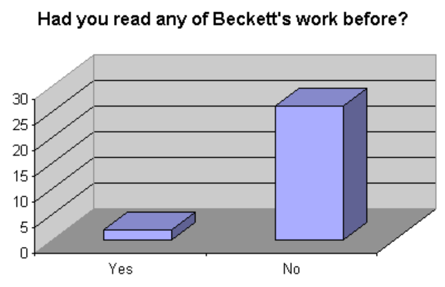

Figure 4. Had you read Beckett before?

Out of 28 responses:

- 26 No, 2 Yes

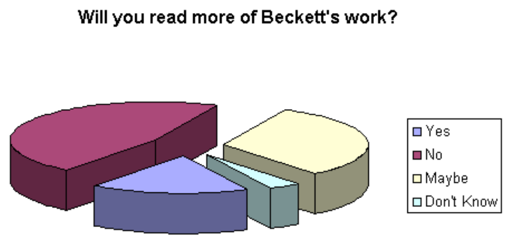

Figure 5. Will you read more Beckett?

Out of 28 responses:

- 5 Yes, 13 No, 9 Maybe, 1 Don't Know

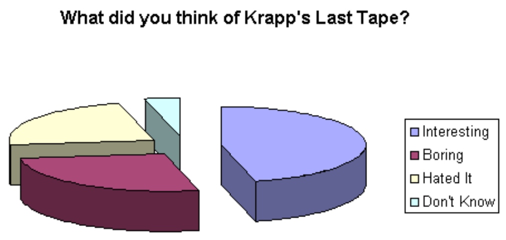

Figure 5. What did you think of the play?

Out of 26 responses:

- 12 Interesting, 7 Boring, 6 Hated it, 1 Don't Know

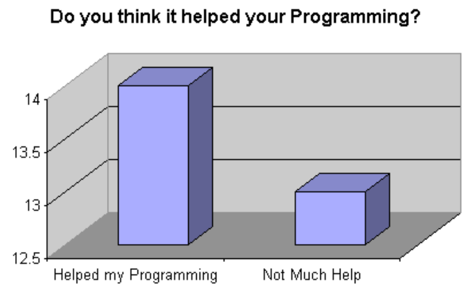

Figure 6. Has it helped your programming?

Out of 27 responses:

- 14 yes, 13 No

Teaching by analogy is an approach which has been used for many hundreds of years and has proven to be enormously beneficial (when properly implemented) whereas teaching by *extended* analogy is a relatively recent development and bears a great deal more investigation.

7 REFERENCES

[1] Kim, J., Kim, J. The Impacts of Examples on Analogical Reasoning: On Programming Education, Technical Report, 1998.
[2] Dunican, E. Making the Analogy: Alternative Delivery Techniques for First Year Programming Courses, PPIG 2002.
[3] Matocha, J., Camp, T., Hooper, R. Extended Analogy: An Alternative Lecture Method, SIGCSE 1998: 262-266.
[4] Goette, T., COBOL on Broadway: An Innovative Approach to Teaching Programming Concepts, Comm. of the International Information Management Association, 3, (2), 129--133, 2003.
[5] Ben-Ari, M. Recursion: from drama to program. Journal of Computer Science Education 11(3), 1997, 9-12.
[6] Yurcik, W., Vila J. Brumbaugh L., An Interactive Web-Based Simulation of a General Computer Architecture, Proceedings of IEEE International Conference on Engineering and Computer Education, (August 2000).
[7] Yurcik, W. Osborne, H. A Crowd of Little Man Computers: Visual Computer Simulator Teaching Tools, Proceedings of the 33nd conference on Winter simulation, December 09-12, 2001, Arlington, Virginia.
[8] Coots, S. (2001) Samuel Beckett: A Beginner's Guide, Hodder and Arnold.
[9] Fletcher, J. (2000) Faber Critical Guides Samuel Beckett Faber & Faber Ltd.
[10] Felder, R.M. (1996) “Matters of Style”, ASEE Prism, 6(4), pp. 18-23.
[11] Keefe, J.W. (1979) “Learning Style: An Overview" in NASSP's Student Learning Styles: Diagonosing and Prescribing Programs” (pp. 1-17), Reston, VA: National Association of Secondary Schools.
[12] Gordon, D. & Bull, G. The Nexus Explored: A Generalised Model of Learning Styles 15th International Conference of Society of Information Technology & Teacher Education 2004, Atlanta, Georgia, USA
[13] Schön, D., 1984, The Reflective Practitioner, Basic Books.