Teaching Computer Science: an NLP Perspective

Andrew Eales
Wellington Institute of Technology
Petone, Wellington, NZ
andrew.eales@weltec.ac.nz

Proceedings of the 16th Annual NACCQ, Palmerston North New Zealand July, 2003 (eds) Mann, S. and Williamson, A. www.naccq.ac.nz

ABSTRACT

Neuro-Linguistic Programming (NLP) offers a rich set of practical tools that can profoundly influence human performance and achievement. NLP has its origins in models developed to describe the intuitive techniques used by exceptionally gifted psychotherapists. It is an epistemology, describing how we know what we know, as well as a methodology that creates models describing the processes that humans use to competently perform specific tasks. NLP also outlines strategies that can be used to discover processes that achieve excellence in the human performance of a specific task. The techniques used by NLP are process-oriented rather than content-oriented, allowing them to be transferred to any discipline. This paper introduces NLP techniques and considers the application of these techniques to the teaching of computer programming. Observations derived from teaching programming courses are used to propose a model of the processes and strategies used by competent programmers. Novice programmers can be taught these strategies to create a psychological environment that facilitates the removal of preconceptions concerning the subject matter. Additionally, successful learning strategies and practical techniques encourage and accelerate the development of practical competency.

1. AN OVERVIEW OF NEURO-LINGUISTIC PROGRAMMING

Neuro-Linguistic Programming (NLP) is a branch of applied psychology that claims to provide techniques that teach the required skills to attain excellence in the performance of a specific task. Developed by Richard Bandler and John Grinder (Bandler and Grinder, 1979), NLP attempts to find ways of modelling human behaviour, particularly behavioural patterns that facilitate the accomplishment of excellence. NLP techniques are derived from models developed to describe the intuitive techniques used by exceptionally gifted psychotherapists. NLP techniques are process-oriented, focusing on the generation of specific outcomes, rather than attempting to specify behavioural content for a specific situation. As these techniques can be applied in the absence of task-specific details they can be applied to a diverse range of situations. Adler (Adler, 1994) and O'Connor (O'Connor, 2001) provide a thorough introduction to NLP concepts and techniques.

This paper introduces NLP techniques and considers the application of these techniques to the teaching of computer programming.

2. SENSORY MODALITIES

Human perception occurs within three primary channels or sensory modalities. These categories of sensory information are visual, kinesthetic (feelings and visceral sensation) and auditory. Secondary modalities

(olfactory and gustatory sensations) are ignored in this discussion, as sensations of smell and taste are unlikely to substantially assist programmers. Primary modalities can be divided into more specific modalities:

visual-analog: shape and colour

visual-digital: written symbols

auditory-tonal: pitch and tone-colour

auditory-digital: spoken language

kinesthetic-primary: sensation and physical feeling

kinesthetic-emotional: emotions.

Sub-modalities provide further refinements within the different sensory categories by considering single aspects of a single modality. Auditory submodalities include pitch, loudness, tone-colour and temporal patterns. All modalities can be either recalled from memory or mentally constructed as illustrated by the following examples:

"What is the syntax of an array declaration in Java?" - visual-digital, remembered.
"How is the following algebraic equation represented in Lisp?" - visual-digital, constructed.
"What is wrong with the fifth line of code?" - kinesthetic-primary evaluation by comparison to stored, visual imagery.
"What does Peter Jackson look like wearing a purple hat?" - visual-analog, constructed.
"What does moon dust feel like?" - kinaesthetic-primary, constructed

Indications of the internal representation system used by a people are gained by paying close attention to eye movements and language usage. Sentences such as "I do not see the relationship.", "This solution to the problem does not feel right.", and "I hear what you say." reflect visual, kinesthetic and auditory thought processes. Common eye movements that also indicate the use of a particular modality include movement to a top, left position (recollection of visual images) and movement to a top, right position (construction of visual images).

3. PERFORMANCE EXCELLENCE

The attainment of mastery in any task requires a progression through different levels of accomplishment. Novices typically start from a state of unconscious incompetence and then progress to states of conscious incompetence and conscious competence. Mastering a skill requires a progression to the highest-level of skill which is termed unconscious competence. Developing conscious competence and unconscious competence requires practice or task repetition. Programming instruction must attempt to continually exercise prior competencies while adding new material as required. For first-year students the amount of programming activity is more important than project size or level of difficulty.

Grinder and Bandler (1979) define a level of skill as the ability to make finer distinctions. Programming skill requires a clear understanding of the individual language constructs and their interaction. Matching appropriate operations and then modifying these to produce the desired outcome creates the solution to a specific problem. The ability to evaluate multiple solutions to a problem indicates a finer distinction than the ability to solve the problem using a single solution. Requesting students to provide or evaluate multiple solutions, or to compare inefficient solutions to an elegant solution can achieve similar results. Note that finer distinctions can occur within a hierarchy of different levels of abstraction that are termed chunks. Upward chunking moves from a more detailed view to a less detailed view, while downward chunking provides the opposite transformation. Programming and software development are similar in nature to creative endeavours such as architecture, writing and music composition, in that all of these activities require extremities of both upward and downward chunking. These extremes, as well as individual affinities towards a specific chunk size make these tasks inherently challenging.

4. LANGUAGE AND LIMITING BELIEFS

The use of language can elicit a specific internal (sub-conscious) representation to be made and encourage a specific ordering of modalities. The use of metaphor can also create individual internal representations. The creation of these representations is the important process, the representations themselves do not have to be consciously acknowledged or explained. Especially important for novice programmers is the understanding created by joining visual-digital (syntactical) representations and visual-analog (semantic) representations. These sub-modalities correspond to programming language syntax and a visual representation of programming language semantics. For example, a two-dimensional array given in visual-digital form: int Apartments [10][4]; has the visual-analog equivalent of a picture of a block of apartments, ten storeys high with four apartments on each storey. It is important that students be provided with the visual-analog representation. A verbal description of the representation forces students to visually construct a

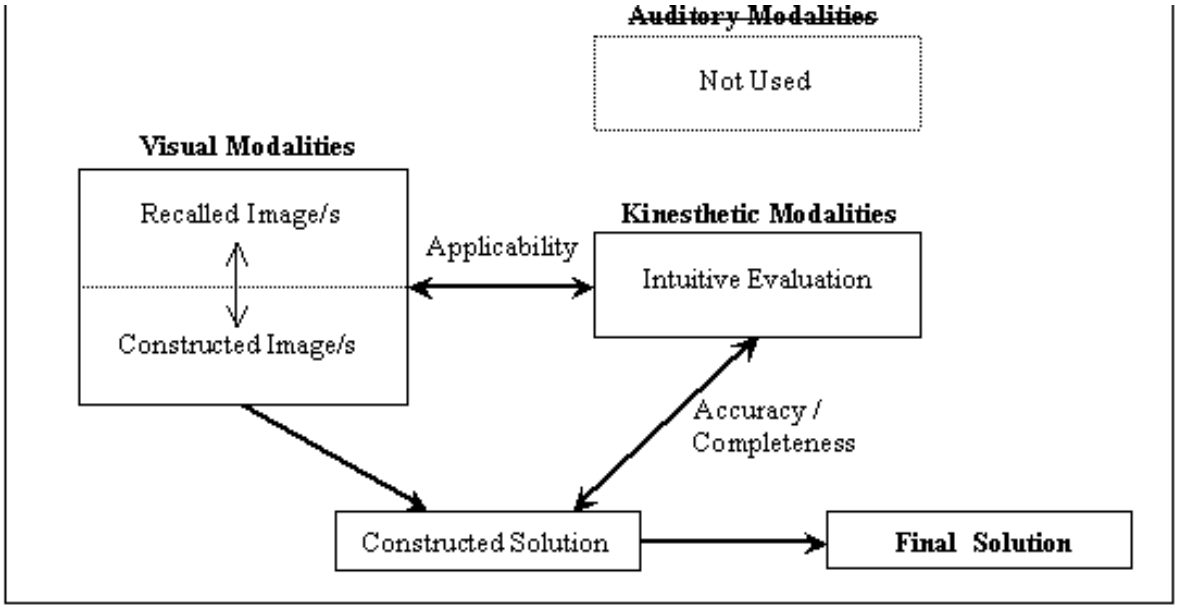

Figure 1. A model of programming excellence

representation, which may or may not be correct. Verbal descriptions are rarely internalised literally (Bandler and Grinder, 1979, p.124). People usually associate verbal descriptions (words) with internal visual or kinesthetic representations that provide semantic content to the words. Creating an accurate visual representation from a verbal description is a higher-level skill that assumes that the verbal description triggers the correct visual-analog representations.

Robert Dilts (1991) regards experience as consisting of a hierarchy of levels. Starting from the lowest level, these are Environment, Behaviour, Capability, Beliefs, Identity and Spirit. Higher levels of experience tend to influence lower levels. Thus the belief "Programming is hard" is likely to negatively influence capability, which occurs at a lower level. Such beliefs are termed limiting beliefs and can be negated by being challenged (Koornhof, 1996) with appropriate responses such as "Hard when compared to which other subjects?", or "Which specific type of programming is hard?", or "Are all programs hard to write?" Many programmers, including seasoned professionals listen to music while working. Music can therapeutically influence behaviour and provide a higher-quality work environment. Unfortunately, these positive influences can be negated by higher-level limiting beliefs.

5. MODELLING PROGRAMMING EXCELLENCE

Sensory modalities form different representational systems that allow us to experience the world in different ways. The selection of different modalities for different situations, as well as the patterns of interaction between different modalities can profoundly influence how we experience the world. The common practice of listening to music while working indicates that auditory modalities play no role in the internal process required during programming. Music may enhance the environment and facilitate behavioural patterns conducive to programming; it does not assist the process of programming. A proposed model of programming excellence derived from discussions with both students and experienced programmers is shown in figure one.

Solutions are created visually by applying recalled images that represent appropriate previous experiences related to the current programming task. The applicability of different recalled images is kinesthetically determined. These recalled images are then used to construct new images representing a possible solution or a promising partial solution. Feedback that evaluates the validity of the generated combinations and possible solutions is obtained kinesthetically. Teaching styles that mimic this process will be more successful than teaching styles that proceed through a different sequence of steps.

Additionally, language usage that violates the model will create confusion and uncertainty. As an example, compare the following two teaching scenarios, adding visual imagery as required to the proposed model:

1. "It is as clear as a bell that by combining this process with this string we obtain a solution. We can then see that the solution is correct by tracing through the process."

2. "We can see that this process is similar to the one studied yesterday. By modifying this diagram of the process we feel that the modified process may provide a solution. By combining the new process with our previous knowledge of strings, a solution appears possible. This solution feels logical and elegant."

The above model is a simplification that can be used in classroom situations. In reality, different people may have subtle differences in the ways that they access and organise internal information.

6. ANCHORING FAVOURABLE STATES

Internal states are essentially feelings and emotional states. When these states become conditioned responses to stimuli, the stimuli are called anchors. These anchors can be used to give access to emotional states. Anchors can be used to replace the feelings of frustration and bewilderment experienced by many students when they are forced to interact with the vagaries of programming languages, compilers and online help systems. A light-hearted moment that breaks a serious illustration or explanation will trigger relaxed internal states.

If this internal experience is accompanied by a change in the instructor's tonal inflection, the experience may be recalled at a later time by using the same tonal inflection. A discussion of anchoring, which can be achieved using a variety of other techniques is beyond the scope of this paper.

7. CONCLUSIONS

NLP provides a rich set of tools that can assist students to master practical tasks. Task repetition increases skills by encouraging both conscious and unconscious competency. The use of visual metaphors encourages internal representations that connect visual-analog and visual-digital modalities. Eye movements and language usage provide indications as which modalities are being used to retrieve or construct images or information. Use of appropriate language and visual imagery, which follows the proposed model of excellence, mimics the modal processes used by successful programmers. Listening to music should be encouraged as it encourages relaxed internal states while at the same time effectively nullifying auditory modalities that are of little value to the programmer. Limiting beliefs can be removed by challenging their validity, while favourable internal states can be recalled by anchoring. By following the processes presented by the proposed model, students can be taught in a manner that is consistent with the strategies used by experienced programmers. This paper provides a necessarily shallow overview of the techniques of Neuro-Linguistic Programming. It is hoped that computer programming instructors may find these ideas interesting enough to warrant further investigation.

REFERENCES

Addler, H (1994) NLP: The New Art and Science of Getting What You Want. Piatkus Publishers, London.

Bandler, R, Grinder, J. (1979) Frogs into Princes. Real People Press, Moab.

Dilts, R. B., Epstein, T. and Dilts R.W. (1991) Tools for Dreamers, 1st Ed. Meta Publications, Cupertine.

Koornhof, P. (1996) of Muses and Magic - Essays for Musicians.Performance Launching Network, Noordbrug.

O'Conner, J. (2001) The NLP WorkbookThorsons, London.