

**MCK03100** ®

**Planning an Experimental Methodology for Measuring Cognitive Performance:  
Involving spatial relations and logical reasoning**

Dr Elspeth McKay

RMIT Business School of Business Information Technology, Australia

## **Abstract**

*This paper describes a comprehensive experiment which spanned four years, with a total of 280 volunteer students participating. To complete the work a novel, interdisciplinary, conceptual framework was implemented comprising: instructional science (for the concept learning models), cognitive psychology (for defining the cognitive style construct), and educational research (to provide the measurement tools for the instructional outcomes). In order to do this, the methodology contained the following elements: a procedure to determine participant's cognitive style, testing instrumentation, which included: a pre-test to evaluate prior domain knowledge, and a post-test to measure improved performance; a cognitive performance measurement tool, capable of accurately positioning participant performance relative to each other, and a test analysis tool to articulate the significance of the findings. The overarching goal of the research was to evaluate the contextual components of instructional strategies, for the acquisition of complex programming concepts. Therefore, a clear definition of the parameters of each research variable was required to determine whether the instructional conditions had an effect on the cognitive performance outcomes. The key research parameters used included: **the method** (learning content, instructional conditions, presentation mode, learner characteristics), **the measurable learning outcomes** (the interaction of instructional method and cognitive style construct). Overall, there were two exploratory studies and a final experiment conducted. Each of the experiments consisted of four stages: cognitive style screening test, pretest to determine prior domain knowledge, instruction period, post-test. The Cognitive Style Construct provided the 2-ways of describing an individual's approach to organizing and representing information (Riding and Cheema, 1991): Wholist-Analytic (the individual's mode of processing information, Verbal-Imagery (the representation of information during thinking). This cognitive style construct proved to be a useful cognitive modelling tool. Richard Riding's Cognitive Styles Analysis (a computerized program), was used as the screening test.*

**Key words:** cognitive style construct, human-computer interaction, knowledge performance bands, test instrument specification matrix, measurable instructional outcomes, meta-knowledge processing model

## **Background**

There are both positive and negative aspects of undertaking a postgraduate research project. Amongst the positive attractions are the personal benefits derived from selecting the paradigms to investigate, topics that ignite imagination, and conducting a methodology that will deliver desirable results. However, mapping a smooth progression through all the decision points, especially in the planning phase of an educational research project can become tricky. There are many methodological decisions to be made when writing project proposals that deter a novice researcher from preparing an innovative research methodology. Examining texts on how to design a robust educational research study, a novice is confronted with a plethora of strategies to choose from. Unfortunately, many of the characteristics of good research involve a certain amount of skill and knowledge that develops over time. Through the dissemination of early career researchers' work, those wishing to tackle a postgraduate proposal can take

heart from their lessons learned. This paper describes one such study that involved an investigation of the interactive effect of cognitive style and instructional format, on a defined set of cognitive performance outcomes. The cognitive style was defined as the verbal/imagery (how we represent information during thinking) and wholist/analytic (the mode of processing information) (Riding and Cheema 1991). The instructional format involved undergraduate students working through self-paced instructional booklets that contained either textual or graphical metaphors to instruct on gaining introductory programming skills (writing simple procedural algorithms) (McKay 2000)(a). Advantages of using the Rasch measurement model, is to utilize the statistical characteristics of the test items to determine the measurement scale, which, in turn, is defined by the items used to develop it (Griffin and Nix,1991):96.

This paper provides an overview of the initial work that led to the final experiment. It describes the environmental context of an experimental study to explain the importance of correctly targeted instructional material. The research question is then expressed producing a sequence of hypotheses that direct the following investigation; and provides the framework for expressing the research findings. The instructional content is then identified as introductory programming knowledge acquisition that involves spatial relations and logical reasoning. The measurable instructional outcomes are identified as programming knowledge performance bands (pkpb). There is an explanation of the experimental instrumentation that was devised to measure the cognitive performance outcomes. It will be shown that there were a number of innate environmental factors impacting on the final results. Finally, the experimental design is discussed in the light of the final experimental procedure, which enabled the measurement of the interactive effects of cognitive style and instructional format on specific cognitive performance outcomes.

## **Overview**

There can be no doubt that measuring human-computer interaction (HCI) is complex (Preece 1994). Difficulties arise for researchers when trying to locate interference from the complex nature of the learning tasks (Yost, Varecka and Marefat 1997). To tackle this problem the author devised a tri-level knowledge framework to deal with these complications. The first level dealt with research process or the fusion of strategic knowledge required for conducting the experiments (Lukose 1992). The second dealt with the mechanisms designed to elicit the knowledge acquisition strategies utilized by the participants to achieve the instructional outcomes. While the third level, would describe the final experiment and the inherent environmental elements, which impact on the results.

Therefore, to correctly identify the strategic knowledge of actual research process, identifying the research variables began with the exploratory studies. The Verbal-Imagery dimension was used as an independent variable, to split the sample into verbal and visual treatment groups. A pretest was used to indicate participants' prior domain knowledge to enable measurement of improved cognitive performance on a post-test. Overall, four programming instructional booklets were created:

- the first booklet enhanced existing text-based instructional material (Bagley 1990) with a mnemonic strategy (pictures/graphics) which was used for the initial exploratory study (McKay 1999)(a)
- the next three booklets, represent the instructional programme designed to elicit specific cognitive performance benchmarks for programming knowledge, that were used for the second exploratory study (McKay 1999)(b), and the final experiment (McKay 2000)(a). The first contained pre-instructional materials, while the remaining two-booklets were used as the experiments' treatment.

In the first exploratory study, the post-test score was the dependent variable with post instruction performance measured by subtracting the pretest scores from the post-test. While in the second and the final experiment, it was cognitive performance on instructional outcomes that was measured. Following is a short description of the first two studies.

### ***First exploratory study (Pilot-1)***

The aim of the first experiment was an attempt to:

**investigate the interaction of cognitive style (Verbal-Imagery) and instructional material (enhanced with graphics) with the acquisition of abstract computer programming concepts.**

It was hypothesized:

**that *Verbalizers* learn abstract concepts differently from *Imagers*, when they have received *text-only* instructional material. Likewise, *Imagers* learn how to apply abstract concepts in varied context differently from the *Verbalizers*, because they are able to transfer *images* of abstract programming concepts into new computing contexts.**

Therefore, it was assumed that Verbalizers would learn new programming concepts better with a text-only instructional format than with the text-plus-graphics. Imagers however, would learn the same set of concepts more effectively with the text-plus-graphics format.

However, Pilot-1 suggested that the Verbalizers appeared to learn the targeted programming concepts better with the text-plus-graphics format (McKay 1999)(a).

Conversely, Imagers were unable to translate the graphical representations into new contexts, and therefore performed better with textual description of the programming concepts.

### ***Second exploratory study (Pilot-2)***

The second experiment shifted the focus from the broad programming learning outcomes, with instructional material enhanced with graphical representations, to take a more selective approach, and explore the relationship of instructional format and

cognitive style (McKay 1999)(b). The aim of this experiment was to explore the following question:

**to what extent does matching instructional strategy with cognitive style enhance performance in a structured programming task?**

Based on the findings of Pilot-1, which were contrary to popular belief, that Verbalizers perform best when receiving textual information (Riding and Caine 1993) and that individuals who prefer to think using pictures rather than words benefit more from mental imagery, than people who use words (O'Halloran and Gauvin 1994), a number of hypotheses were proposed.

Based on the results of Pilot-1, it was hypothesized in the second experiment:

**that the *Verbalizers* learn abstract concept differently from *Imagers*; they will benefit from the *text-plus-graphical metaphor* instructional strategy. Furthermore, *Verbalizers* learn how to apply abstract programming concepts in varied context differently from the *Imagers*. Likewise, *Imagers* will be unable to utilize the *graphical metaphors* to improve their acquisition of abstract programming concepts.**

Verbalizers appear to be better at transferring their conceptualisation of the graphical representations into new computing contexts.

Therefore, it was proposed that Verbalizers would learn new programming concepts better with a text-plus-graphical metaphor format rather than the text-plus-textual metaphor material.

Conversely, it was proposed that Imagers would learn the same set of concepts more effectively with the text-plus-textual metaphor format.

## **1. Building the Context**

An important element of courseware design is the attention given to the anticipated characteristics of the target learners (Dick and Carey 1990). Therefore, to establish details relating to particular characteristics of the tertiary population (from which the sample would be drawn), a questionnaire was given out to first year students (Mager 1988), enrolled in a Bachelor of Business at an Australian university. This profile facilitated the design of instructional materials for the internal/external conditions-of-the-learner. The results of this questionnaire revealed details on: their physical characteristics, formal training, anticipated attitudes, and their sources of reinforcement

### **1.1 Physical characteristics**

It was determined they would range in age from 18 to 48 years. The sample would be half female and male; some would be living at home with parents; most travel independently (by car); and there would be no apparent physical limitations (assistance to be provided if additional support is needed).

## **1.2 Formal training**

There would be varied levels of computer literacy. Most will be undergraduate students with some post-graduate mature age participants; all will have experienced training type workshops/tutorials; some participants would be articulating from Technical and Further Education (TAFE); most will have completed VCE; English reading and writing skills are assumed to be well developed; the recent school leavers will have well developed computer literacy skills; mature age participants may require an additional introduction to DOS; some may possess limited/weak computer literacy skills; participants with applied science backgrounds may possess mixed levels of computer literacy. Within this group, there may be a higher tendency for haptic tendencies (Howell 1972) with lower verbal processing skills (tutors' tutorial observations). There may be several of them exhibiting the dependent cognitive style, with fewer independents ((Moore and Bedient 1986). While those students possessing a high computer literacy level may have lower than average reading/writing skills. They may not be able to concentrate for long-periods and may not realise that they know how to write an algorithm. Finally, some participants may be familiar with a 4GL (known as procedural languages, which automate the programming procedures, rather than have the programmer direct every part of the process).

## **1.3 Anticipated attitudes**

A free training workshop for further business computing studies will be appealing. They should be eager to participate in an accelerated learning programme to achieve course level knowledge without the stress involved with a formal exam process. There will be a willingness to transfer learning to related learning domains. They may exhibit apprehensive tendencies towards their ability to perform logical processes. In relation to their interests, they will be highly varied. Some may have self taught computer literacy skills; few will be married; some will have family commitments; and some younger participants may socialise freely on campus during other studies.

## **1.4 Sources of reinforcement**

It was determined that the participants would perceive that skills gained during the workshop could be transferred to work in other computing packages. For instance, to object oriented programming knowledge, and database programming environments.

# **2. Research Objectives**

The overarching question raised by this research project was:

**Does the interaction of instructional format and the cognitive style construct effect the acquisition of abstract programming concepts?**

At the time, there was no experimental evidence to validate these issues in the context of computer (knowledge)-mediated learning. Therefore, the final experiment was an attempt to provide relevant evidence to test the following hypotheses:

The first hypothesis tests whether instructional format affects the acquisition of abstract programming concepts in terms of a cognitive style construct (Riding and Rayner 1998). For instance, do the Analytic-Verbalizers perform best when receiving text-based instructional material? Conversely, do the Wholist-Imagers perform best when receiving pictorial-based instructional material?

Therefore the first hypothesis formulated a general statement covering the overall purpose of the study:

**H<sub>1</sub>**  
*Instructional treatment will have an effect on the cognitive performance of one cognitive style group compared to another.*

In order to analyse the effects of instructional treatment and the cognitive style construct, it was necessary to conduct separate comparisons of each independent variable. For instance, could it be shown that the effects of instructional format will be different for a particular group of individuals? In other words, could it be shown that the individuals who possess the same cognitive style characteristics (Wholist-Verbalizer, Analytic-Verbalizer, Wholist-Imager, and Analytic-Imager) perform any differently from their counterparts receiving the contrary instructional format (text-plus-textual metaphors/text-plus-graphical metaphors)?

The second hypothesis was:

**H<sub>2</sub>**  
*Cognitive performance will be affected by the type of instructional treatment they receive.*

Therefore, analysing the separation of the independent variables, the cognitive performance within each instructional treatment was necessary. For instance, could it be shown that different cognitive style groups (Wholist-Verbalizer, Analytic-Verbalizer, Wholist-Imager, or Analytic-Imager) perform differently when given the same instructional format?

The subsequent hypothesis was:

**H<sub>3</sub>**  
*Cognitive performance will be affected by the individual's cognitive style.*

To complete the analysis of the interactive effect of instructional treatment and cognitive style on the acquisition of abstract programming concepts, an investigation of the **interaction** of cognitive style and instructional format is necessary. For instance, could it be shown that Verbalizers receiving the text-plus-textual metaphors perform differently from Imagers receiving the text-plus-graphical metaphors?

Therefore the final hypothesis this dissertation investigated was:

**H<sub>4</sub>**

*Cognitive performance will be affected by the interaction of cognitive style with the type of instructional treatment.*

The first layer of the tri-level knowledge framework devised to deal with the research process and fusion of strategic knowledge for conducting such a complicated research project was almost complete. Clarification of the instructional content (discussed in brief next) completes the definition of the research process.

### **3. Strategic Knowledge of Programming Concepts**

There were three instructional modules identified in the learning hierarchy for writing new computer programs. The anticipated instructional outcomes required to demonstrate strategic programming knowledge, were defined as separate instructional modules to depict: the entry level skills (*basic computer literacy*), problem solving (*redefining the problem to prepare an algorithm*), and knowledge of the programming language rules.

Each module identified a sequence of tasks necessary to achieve the instructional outcome. The process of acquiring the procedural knowledge required for writing computer programs, involves an internal/external exchange process (McKay 2002). It was necessary to determine levels of previous familiarity with similar cognitive processes. Therefore, when the participants were tested, they were encouraged to the answer in their common idiom. This was a deliberate attempt to identify levels of prior knowledge applicable to programming.

Consequently, the lessons learned from the Pilot studies were important:

#### **3.1 Defining prior domain knowledge level (Pilot-1)**

Prior domain knowledge was defined as the learner having previously developed prototypes of abstract concepts, specifically within a computer-programming context. The participants were classified as possessing novice or experienced programming knowledge, according to their pretest score. In the first exploratory study, novice-programming knowledge was selected as a pretest score of <28%. It was deduced from these figures, that these participants had no prior knowledge of programming. Experienced programming knowledge was reflected by a pretest score of =>28%. None of these experienced programming participants had professional experience in computer programming. Like the Bagley (1990) study, their experience with programming, was limited to previous courses at secondary college, and/or exposure to other programming languages.

#### **3.2 Benefit of hindsight (Pilot-2)**

In Pilot-2, the instructional content became more focussed by narrowing the instructional context. Therefore the definition of the independent variable (instructional strategy) was more complex (additional programming metaphors). Given that these metaphors were to act as active cognitive processing agents, the importance of locating the participants' prior domain knowledge levels became more evident.

Although the participants were novice-programmers, the high performance levels on the pretest for the first exploratory study (Pilot-1) could be explained by the association learners can make between new information and prior knowledge, when using an elaborative interrogation strategy (Woloshyn, Wood and Willoughby 1994). They suggested that learners were able to make inferences and elaborations about new materials by answering why type questions. This means, that the simple or basic nature of the original Bagley (1990) pretest may have provided the equivalent of an elaborative interrogation strategy. This being the case, an alternative assessment methodology, to accurately measure the changes in cognitive skills performance, was chosen for the final experiment. The effects of prior domain knowledge (investigated by Bagley,1990), were further minimised by using the same structured format in both instructional booklets for both treatments.

## **4. Acquisition of Programming Knowledge**

The second level of the tri-level research knowledge framework was to locate the variables, likely to instigate the learning process. Because of the need to devise a reliable way to measure the interaction of delivery strategies and cognitive style, the instructional treatment was an important independent variable, as it directly relates to the internal/external exchange process by linking the major components of instruction; method of delivery, instructional conditions, and instructional outcomes (Reigeluth 1983). As such, these variables can be taken up as distinctly different research treatments (Fig:1). Consequently, it was important to employ a learning domain, which could be broken into discrete modules for the purpose of measuring strength of cognitive performance.

**The main instructional outcome was to devise an instructional strategy for acquiring the cognitive skills to develop programming algorithms, which use the DOWHILE and REPEAT UNTIL control structures.**

The following discussion explains how the experimental design evolved; the research parameters were identified and described in detail (McKay 2000)(a); they will be outlined in brief for the purposes of this paper. Next there will be a short section describing the mechanisms used to elicit the strategic knowledge (acquisition of introductory programming skills). Defining the learning content, and motivational tools will follow to provide the reader with more information on the research design methodology. There will be a description on the strategies adopted to validate the learning content, measurable instructional outcomes, and the assessment instrumentation.

### **4.1 Research Parameters**

Work proceeded on converting a study conducted by Dr Carole Bagley, in 1990. Her study involved an investigation of structured versus discovery instructional formats for improving the learning of programming concepts by novice and experienced adult learners. The strong instructional design framework of this research provided a fine example of how to clearly define the parameters of each research variable (Merrill 1994).

Consequently, the instructional treatment used in the first exploratory study consisted of the Bagley (1990) self-paced Structured Instructional Format Booklet. Fig:1 presents some of the key research parameters used in this design, including: the method (*learning content, instructional conditions, presentation mode, specific learner characteristics*), and the measurable learning outcomes (*instructional method, cognitive style construct*).

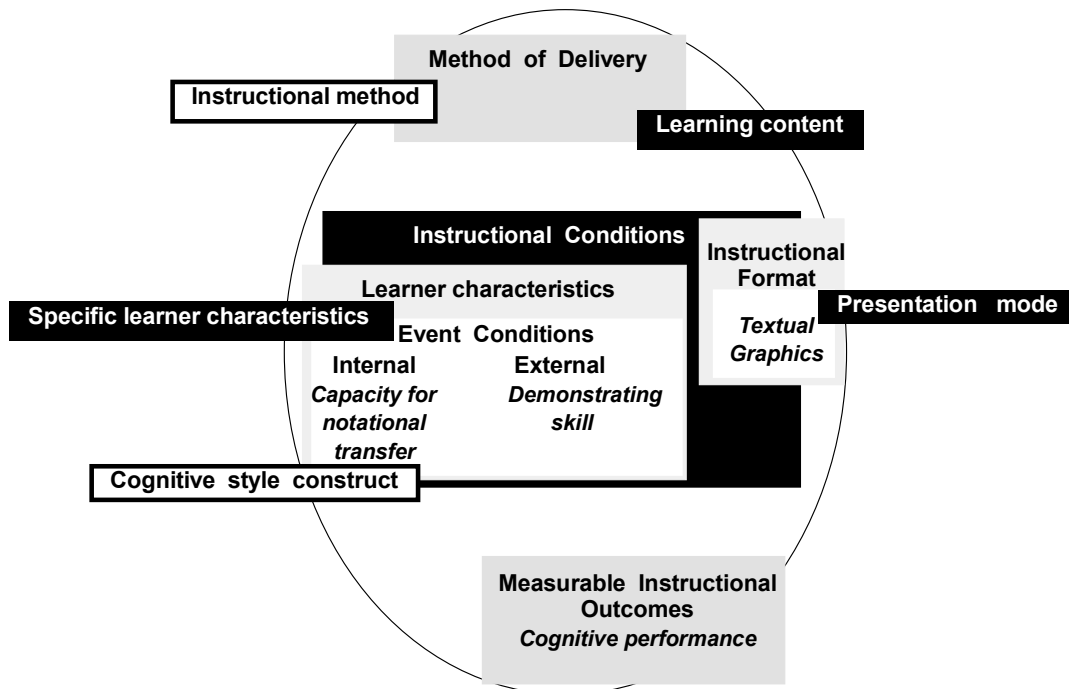


Fig:1 Research Parameters

## 4.2 Mechanisms to Elicit Strategic Programming Knowledge

After running the first exploratory study, it was felt that limiting the learning content to programming logic patterns would provide a more robust research design. Therefore, in order to conduct an empirical experiment, a new self-paced instructional booklet was designed to isolate the internal/external conditions-of-the-learner and to take advantage of the notational transfer that occurs when assimilating new information with previously learned cognitive strategies.

There are two important considerations about which you must be aware before designing a DOWHILE loop.

Firstly, the testing of the condition is at the beginning of the loop. This means that the programmer may need to perform some initial processing to adequately set up the condition before it can be tested.

Secondly, the only way to terminate the loop is to render the DOWHILE condition false. This means you must set up some process within the statement block which will eventually change the condition so that the condition becomes false. Failure to do this results in an endless loop.

Fig:2 Common Expository Instructional Format

The instructional strategy adopted expository learning (Fig:2) and interrogatory sections (Fig: 3).

Instructional format articulates the learning content. It comprises many components, including text (Wileman 1993), pictures/graphics, and symbols (Salomon 1979 Reprint 1985); (Perkins 1991); (Merrill, Tennyson and Posey 1992). The new instructional material was designed to draw on the learner's syntax and language skills (both verbal and visual) to promote understanding and to maintain the proper order of concepts. The words are cues in the individual's working memory (Gagne 1985). Therefore, picking the correct instructional format was critical in providing for the acquisition of new concepts not previously encountered.

There were three types of instructional formats used: generic text instructions, text-only programming metaphors, text-plus-graphical programming metaphors.

#### **4.2.1 Generic text instruction**

To ensure that all participants received enough instructional content on abstract programming concepts, it was necessary to devise the introductory material as generic (*text-only*) instructions shown in Fig:2 above.

Look at each of the following programs. Decide whether the output for each example is accurate. If it is, circle YES, if not, circle NO in the table following the examples. Check the characteristics of the assignment statement and use of variables and check those that have been met in the example and those that have not been met.

*Fig: 3 Interrogatory Strategy (Bagley,1990:41 Lesson-2)*

This was primarily prescriptive material, which included: how to proceed through the booklet, general descriptions of the concepts, the tutorial objectives, task descriptions, and the summariser (Reigeluth,1983).

Expository instructional sections that were common to both treatment booklets were represented without the addition of metaphors (Fig:2).

#### **4.2.2 Text-plus-textual metaphor**

Similes and metaphors need to be understandable, despite their novelty and non-literal nature. Good metaphors are like good detective stories, with the connotative richness of meaning accounting for human's ability to interpret metaphors without specific prior learning (Tversky 1977) (Figs: 4 and 5).

Metaphors, using everyday looping functions were considered ideal instructional mechanisms to represent closely similar instances of examples and non-examples (Merrill 1994) of programming concepts. Furthermore, they enable the learner to recognize the distinguishing features of concept instances to be executed throughout the subsequent instructional material (Gagne 1985).

```

City loop tram circuit algorithm
  DOWHILE >=8am and <=6pm
    Tram leaves Flinders street depot at 8am to pick up passengers
    Tram travels on city loop to pick up and/or set down passengers
    Add number of passengers picked up by City Loop Tram to daily
list
  ENDDO
Print total number of passengers using City Loop Tram Service daily
END

```

Fig: 4 Text-plus-textual Metaphor Format

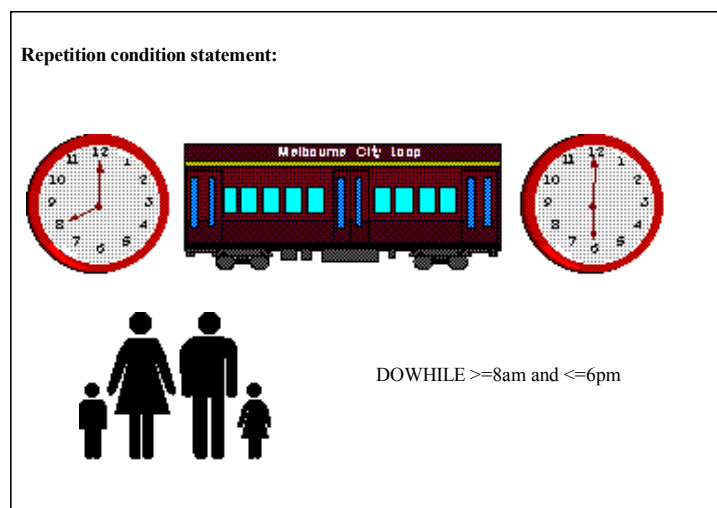


Fig:5 Text-plus-graphical Metaphor Format

Textual metaphors (Fig:4) were utilized in the second exploratory study to expand Bagley's expository instructional strategy (Fig:2). In so doing, they articulated the critical attributes of the concept-to-be-learned (Merrill 1994). In most cases, the textual metaphor was used after the generic textual description.

### 4.2.3 Text-plus-graphical metaphor

This instructional format involved the same lesson on programming logic patterns as the text-plus-textual metaphor format booklet. However, the textual metaphor was replaced with a graphical representation (Fig:5). Graphics picked to represent the textual metaphors were chosen by their recognisable and distinguishing (or salient) features.

They were introduced into the instructional strategy on the first page to visually orient the learner for the internal/external exchange processing.

Graphical metaphors replaced the textual metaphors, whenever possible. However, replacing the textual metaphor entirely, proved to be problematic for particular sections. For instance, replacing the sections on the comparison of the

DOWHILE, and REPEAT .. UNTIL constructs with graphical metaphors was difficult. This difficulty arose because the mix of required technical information and textual metaphor were too closely aligned to enable the text to be totally removed without degrading the example (Fig:6).

In this case, the graphical metaphors were placed beside the relevant textual metaphors. The graphical metaphors, however, were used to completely replace the textual metaphor for the task listings.

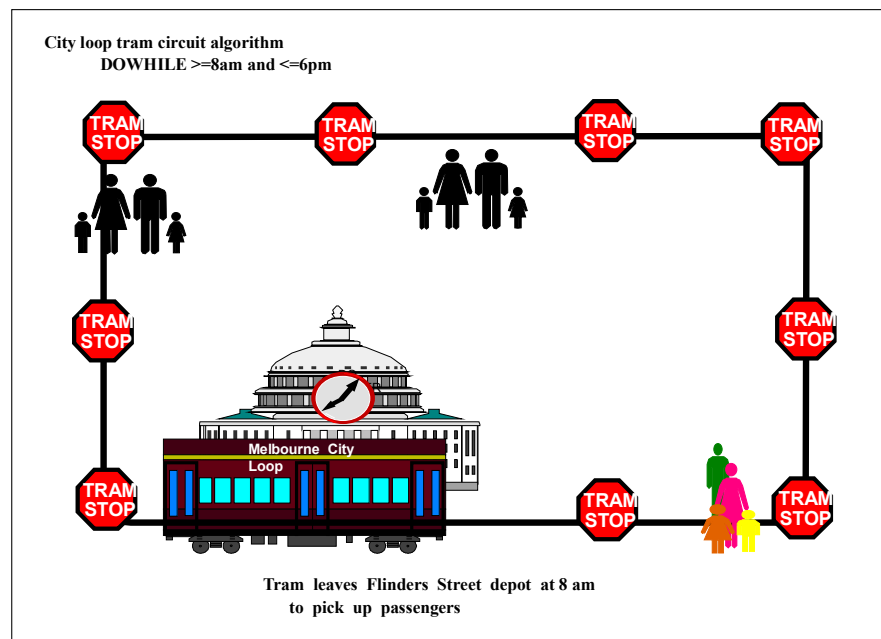


Fig: 6 Text-plus-graphical Metaphor and Vital Technical Information

### 4.3 Defining the Learning Content

The Bagley research design also enabled the isolation of the instructional strategy components, in terms of the Gagne capabilities (verbal information, intellectual skills, cognitive strategies, motor skills, and attitudes).

The learning content comprised the Bagley PASCAL programming lessons: data types, program structure, and writing new programs.

Dividing the learning content into these lessons was an example of clumping as described by Mager. This enables the instructional tasks to be identified as separate learning modules. A learning content analysis (Merrill 1994) revealed that the instruction targeted two-types of programming concepts to be acquired, at three-levels of programming knowledge: basic (*verbal informational*), intermediate (*cognitive strategies*), advanced (*procedural knowledge*).

The two types of programming concepts were: concrete concepts, and process concepts (*or defined / abstract*).

Behavioural analysis identified the mental operations the learner must be capable of, in order to apply the new programming concept at higher skill levels (Zemke and Kramlinger 1982); (Mager 1988). The instructional analysis (Merrill 1994) identified the instructional conditions that facilitate the learner gaining experience using the process concepts, including: the characteristics of the learners, the instructional media, and the levels of instructional guidance.

Definition of the learning content evolved during the running of the first exploratory study, when the author observed that the participants' concentration and motivation waned as the experiment proceeded, and that the original Bagley learning content covered too many different concepts (McKay 1999)(a).

#### **4.3.1 Narrowed focus**

Due to the broad nature of the Bagley PASCAL lessons, a new learning content was designed for the second exploratory study (Pilot-2), and subsequently, for the final experiment. It was developed according to the concept design strategies of Reigeluth (1983); (Robertson 1994); and Merrill et al (1992). To ensure validity of the learning content, three experienced programmers conducted a protocol analysis of basic programming concepts: a computer science lecturer, a professional programmer, and the author.

They provided the programming logic patterns that were teachable in a one-hour practical tutorial session. Another task analysis of the learning content provided the sequence in which these abstract concepts were presented, including the organizational structure for the detailed presentation within each concept.

As the amount of instructional content was reduced for Pilot-2, it was easier to place an emphasis on progressing the instructional modules through the Concept Learning Development (CLD) levels (Klausmeier and Sipple 1980). For instance, carefully chosen metaphors were utilized as notational transfer agents. In many cases, they map directly to the critical attributes of the abstract programming concepts involved in the instructional content (McKay 2002)(a).

### **4.4 Motivational Tools**

Three self-paced instructional booklets were developed. The first was the Tutorial-1 booklet (9-pages), comprising the pre-instructional material for use by all the participants. The other two booklets were designed to be used in conjunction with the experiment. They are referred to as the Tutorial-2 booklets: Treatment-1 (13-pages) contained textual metaphors, and Treatment-2 (21-pages) contained graphical metaphors.

Approximately half of the sample was given Treatment-1, with the remainder receiving Treatment-2. These instructional treatments are discussed next.

#### **4.4.1 Self-paced pre-instructional booklet for Tutorial-1**

This booklet commenced with a brief review of the introductory PASCAL lecture material. The expository content for each learning goal included: definitions, best example, and a range of examples, non-examples, and worked examples of problems (Merrill et al. 1992).

The Tutorial-1 booklet was divided into three sections: the lecture review, the examples of simple PASCAL programs and practical problems with supplied solutions, and the hands-on computer activity to access PASCAL by typing in four-simple programs.

The instructional format and examples for the programming concepts in this booklet were developed drawing on Bagley (1990), (Hennefeld 1989), and (Savitch 1989).

#### **4.4.2 Self-paced pre-instructional booklet for exploratory study**

The learning content was developed according to the programming conventions used by Robertson (1994) for developing algorithms and pseudocode (or structured-English statements), using sequence, selection and repetition logic patterns (or programming control structures). The generic textual instruction was designed to guide learners through progressively more complex content (Bagley 1990). Following the expository learning content, a four page interrogatory section was provided to lead the learner through the instructional material and to complete the programming problems. Gaps were left intentionally in the practice examples to encourage the learner to write parts of the algorithms on their own (Fig:7). Gradually the learner creates more of the algorithms, with less information given. This fading technique was shown by Bagley to be more successful for novice programmers than for experienced programmers. The novice is encouraged to build schematic (Romiszowski 1981) (Hummel and Holyoak 1997) and taxonomic knowledge (Merrill et al.1992; Merrill,1994), that is easily transferred from both short and long-term memory (Winn 1982-a).

Instruction commenced with a description of an algorithm. An epitome statement (Reigeluth, 1983), provided the tutorial objectives and instructional goals for each sub-lesson. The learning content comprised four sub-lessons: logic patterns (*sequential, repetition, and conditional control structures*), repetition logic (*using the DOWHILE and the REPEAT .. UNTIL structures*), comparison of the two logic patterns, and example problems.

Each sub-lesson defined the content to be covered. After each expository learning section (Fig:2), a textual metaphor provided an everyday example of the concept being taught. A short review or summariser of the four sub-lessons was given, followed by an expanded epitome statement, which was devised to lead the learner to the next part of the lesson.

Two example problems that were explained in the instruction were expressed as textual metaphors (Fig:4). The examples and the following practice problems were presented in a common instructional format (Fig:2). This common format involved: an objective, the situation description, and the task description that involved a six point suggested solution.

The following task headings were designed to develop procedural knowledge: redefine the problem (*in terms of input/processing/output*), logic patterns (*control structures required to support the solution*), the repetition question (*most novice-programmers are unaware of the need for conceptualising this component*),

repetition starting point (*once again novice-programmers find this difficult*), repetition conditional statement, alternate repetition condition, and a suggested solution algorithm.

The example problems were then followed by two more practice problems. The learning task section was presented as an interrogatory learning format (Fig:3). This type of instructional format reinforces and strengthens the learner's declarative, procedural and conceptual knowledge (Bagley 1990). Suggested solutions were given in a complete form for the first practice problem (Fig:7). However, the final task of writing a complete solution algorithm was left blank for the participant to finish in the second practice problem. Leaving gaps in this manner provides learners with an opportunity to practise their declarative knowledge.

Progressing through the problem examples, participants could check their procedural and conceptual knowledge against the suggested solutions. This technique is known as a coaching technique that points out key issues relating the problem to the definition, best examples, and non-examples (Bagley 1990).

<p><b>EXAMPLE 1: CITY LOOP TRAM SERVICE</b>            Repetition question:            What is the time?            .....            Repetition starting point:            Tram depot            .....            Repetition condition statement:            DOWHILE &gt;=8am and &lt;=6pm</p>	<p><b>PRACTISE PROBLEM1: SUPERMARKET SHOPPING</b>            Repetition question:            .....            Repetition starting point:            .....            Repetition condition statement:            .....</p>	<p><b>PRACTISE PROBLEM1 SOLUTION: SUPERMARKET SHOPPING</b>            Repetition question:            Have I finished shopping yet?            (Do I have more items on my list to buy?)            Repetition starting point:            New aisle            Repetition condition statement:            DOWHILE items on list            REPEAT ... UNTIL list completed</p>
<p><b>PRACTISE PROBLEM2: GRADUATION DAY</b>            Repetition question:            .....            Repetition starting point:            .....            Repetition condition statement. Write both a DOWHILE and REPEAT ... UNTIL statement:            .....            Solution algorithm            .....            .....</p>	<p><b>PRACTISE PROBLEM2 SOLUTION: GRADUATION DAY</b>            Repetition question:            Can I graduate?            Repetition starting point:            Enrolment day .. start of a new semester            Repetition condition statement. Write both a DOWHILE and REPEAT ... UNTIL statement:            DOWHILE subjects to complete            REPEAT ... UNTIL graduation            Solution algorithm            ** This time complete your own solution algorithms** ...            .....</p>	

Fig:7 Coaching Technique (McKay,1999)(b)

## 5. Validating the Learning Content

Learning content validity testing was performed on the pretest instrument and confirmed by independent content specialists. Variance on the pretest was established by Pilot-1, to determine scoring range for levels of prior domain knowledge; a pretest score of <28% for novice-programmers, while the experienced-programmer group's score was recorded as =>28% from the data obtained. The lower reliability recorded by the experienced-

programming group was related to the higher pretest scores, reflecting less change between the pre and post-test scores. These pretest results supported the Bagley, 1990 thesis. Accordingly, the pretest met the original objective of dividing the learners who passed the pretest after taking a programming course, and were, therefore, considered experienced-programmers.

## 5.1 Conceptual learning development

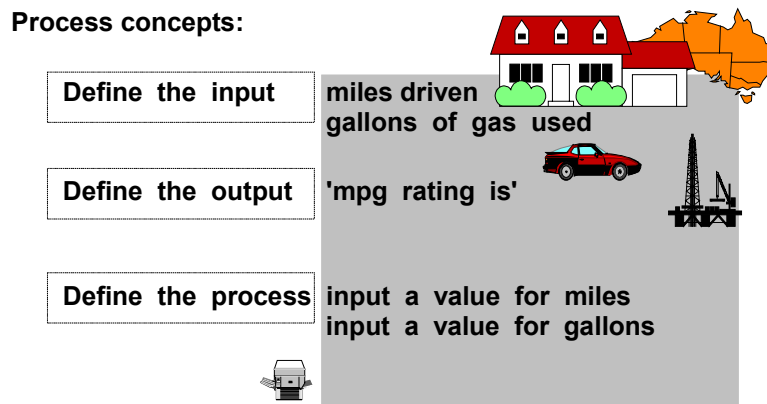
To facilitate the internal/external exchange process, the redesigned instructional content followed Klausmeier & Sipple's (1980) CLD model (Fig:8). A learning content analysis was performed on the test-items to identify the programming concepts required to achieve a correct answer. Consequently, the instructional outcomes, mapped directly to CLD, may be summarised as: (a) the discriminating (*concrete*) level, (b) distinguishing between different data types, and (c) the generalising (*identify*) level. The latter level depicts knowledge of program structure and procedure for output, formalising the knowledge gained from the previous levels, representing inclusive instructional outcomes, knowledge of logic patterns and knowledge of the (Bagley 1990) six-step problem solving process.

Klausmeier's CLD Attainment Levels	Definitions	McKay Implementation	
<b>Formal</b> <i>discriminate between newly encountered instances</i>	More than one example Observable 3-D examples Defined intrinsic and/or functional attributes	<b>Cognitive Strategy</b>  <i>Knows the complete "how"</i>  <i>Recall of simple prerequisite rules &amp; concepts</i>	<b>Procedural knowledge</b>  <i>Integrates learning from different areas into a plan for solving a complete problem</i>
<b>Classificatory</b> <i>generalize to newly encountered example</i>	(Newby&Stepich,1987) <i>Abstract concepts</i>  (Klausmeier&Sipple,1980) <i>Process concepts</i>		Identifies sub-tasks Recognizes unstated assumptions
<b>Identity</b> <i>recall of learned examples</i>	(Gagne,1985) <i>Concrete concepts</i>	<b>Intellectual skill</b>  <i>Discriminates: understands concepts &amp; principles</i>	<b>Declarative knowledge</b>  <i>Higher-Order-Rules: Applies basic concepts to new programs</i>
<b>Concrete</b> <i>recall critical attributes</i>	(Merril et al.,1992)	<b>Verbal information skill</b> <i>Knowing basic programming terms</i> <i>The Knowing of "that"</i>	<b>Declarative knowledge</b>  <i>States isolated rules</i>

Fig:8 McKay Concept Attainment Model

## 6. Measurable Instructional Outcomes

A three-level graded programming skills framework was applied to the test-items. Performances on concrete and process concepts were the two categories of instructional outcomes that were measured for Pilot-1. For instance, there was an expectation that the participants would master all three concrete PASCAL concepts (char, integer, and real). However, the conceptualization of these process concepts for programming requires special instruction, to assist with the notions of the internal/external exchange process. Subsequently, the instructional outcomes reflect the matching Gagne cognitive capabilities for knowing the following: the order of operations/evaluation of statements, concepts of logic flow, use of the accumulator for totalling values, and steps to use in creating a program's structure.



*Fig:9 General Visual Representation*

Furthermore, this instruction concentrated on the critical, defining, and variable attributes (Klausmeier & Sipple 1980). For instance, in the case of the integer data type, the critical attributes were the rules associated with positive or negative whole numbers (like commas and decimal points), while instruction involving the defining attributes included other attributes of the data type, plus the critical attributes. Finally, the abstract concepts of programming constants and variables, the defining attributes, were the performance of arithmetic operations on numerical data only; or to perform the process of initialising the variable's name. The pictures chosen for the instructional format were visual representations of these identifiable attributes (Fig:9).

A Test Instrument Specification Matrix was devised for the second and final experiments (Fig:10). The horizontal axis was used to depict the instructional objectives, with the vertical axis used for the programming content (or learning domain). Programming of logic patterns are a significant sub-division of learning within programming. The learning domain is shown here as a continuum, beginning with simple programming concepts at one end, developing into more complex conceptual programming tasks at the other. There were nine-categories of learning identified along this continuum. The instructional objectives consisted of two-categories of specific programming knowledge.

The first category was declarative knowledge: it was divided into two-levels of skill: verbal information (*knowing isolated rules*), and intellectual (*knowing how to discriminate between concepts and principles*).

The second was procedural knowledge, divided into three-levels: intellectual skill (*higher-order-rules for problem solving*), cognitive strategy (*recognising sub-tasks*), and the ability to integrate learning across learning domains (*for implementing a comprehensive plan of action*).

Instructional Objectives : Programming Knowledge							
		Declarative		Procedural			
		Band-A	Band-B	Band-C	Band-D	Band-E	
		Verbal information skill	Intellectual skill	Intellectual skill	Cognitive Strategy	Cognitive Strategy	
		Concrete Concept Knows basic terms Knows <b>"that"</b>	Basic Rule Discriminates Understands concepts & principles	Higher-Order-Rule Problem solving Applies concepts & principles to new situations	Identify sub-tasks Recognizes unstated assumptions	Knowing the <b>"how"</b> Recall simple prerequisite rules & concepts Integrates learning from different areas into a plan for solving a problem	
Task No:	Learning Domain:						Totals:
9	Solution algorithm						
8	Conditional logic						
7	Until logic characteristic						
6	While logic characteristic						
5	Repetition question						
4	Logic patterns						
3	Basic mathematics						
2	Programming process						
1	Defining diagram						
	<b>Totals:</b>						

Fig:10 Test Instrument Specification Matrix

### 6.1 Programming knowledge performance bands (pkpb)

There were three knowledge bands identified to differentiate the instructional outcomes. There was a mix of declarative and procedural knowledge as the instructional outcomes from the lowest and middle cognitive performance bands, while procedural knowledge was at the top of the programming performance scale.

The instructional outcomes from the lowest-pkpb involved: *verbal information skill* (knowing the basic terms and rules), *intellectual skill* (understanding concepts), *cognitive strategy development* (a demonstrated ability for knowledge integration).

The middle-pkpb involved: *verbal information* (recalling body of previously acquired concepts), *intellectual skill* (a demonstrated ability to identify sub-tasks), and *cognitive strategy* (integrating the previous learning of higher-order rules).

The upper-pkpb involved: *cognitive strategy* (a demonstrated ability to understand the integration of concepts), and *intellectual skills* (higher-order rules including identifying sub-tasks).

## 7. Instrumentation

The assessment of cognitive performance was carried out using a pre and post-test. The pretest (used as a screening test for prior domain knowledge) was given to all participants before the instruction period. The pretest score data was used in the first exploratory study to classify the participants as novice- or experienced-programmers (Bagley 1990).

However, in the second and final experiments, the pretest was used to determine an individual's cognitive performance by recording the difference between the pre and post-test performance.

The following is a brief discussion involving the: pretest and post-tests, the need for common test-items, and the marking strategies.

### 7.1 Pretest

This test was carried out in the participants' normal tutorial classrooms. The first exploratory study used the Bagley 10-item test for both the pre and post-tests to test six abstract concepts written to test participants' knowledge of the four-levels of Bloom's (1956) learning taxonomy. Consequently, it targeted the comprehension, application, analysis, and synthesis levels: distinguishing between different data types, demonstrating the ability to represent data types, knowledge of program structure, knowledge of constants and variable values, a demonstrated ability to evaluate arithmetic expressions, the ability to display output on the screen, a knowledge of logic patterns, and a demonstrated ability to follow a six-step problem solving process.

A new 14-item pretest was designed and calibrated (tested on a group of 47-students not likely to participate in any of the experiments), for the second exploratory experiment. This new pretest was constructed using more of the easier calibrated test-items. In order to reduce the writing-time, it was decided to pick achievable items for the beginning of the pretest. It was also hoped that this would enhance the likelihood of obtaining meaningful answers from the non-programming participants. Reducing their stress was particularly important, considering this test was administered before they received the instructional material. The more difficult items were included to identify the experienced programmers.

The instructional outcomes expected in this pretest can be described in terms of increasing levels of programming skill.

In keeping with this, there were three-levels of cognitive performance on the test-items: *verbal information* (basic skill), *intellectual skills* (mid-range skill), and high-level cognitive strategies (advanced skill).

The test-items were designed to test cognitive performance in student learning of abstract programming concepts covered by the instruction booklets. Consequently, they were directly related to program logic patterns.

However, the Pilot-2 pre-test scores were relatively high with limited variation (McKay 1999)(b). Therefore, the assessment instruments were again modified for the final experiment. Both the pre and post-tests were expanded to comprise 20-test-items each, representing a wider range of difficulty.

## **7.2 Post-Test**

Immediately after completion of the instructional period, in each of the experiments participants were given the post-test. The difference between pre-test and post-test scores was used as the dependent variable. However, there was more emphasis placed on the mid-range skills for this test. The solution examples were measured on specific instructional content.

## **7.3 Graded skills**

A three-level skills grading scheme, devised from the programming skills hierarchy (Gagne 1985), was used to increase the range of difficulty (variance) on the pre and post-tests.

These post-test test-items were designed to target three-levels of performance outcomes: (a) *Skill level-1* comprising two-straight forward (or basic) test-items (like redefining the problem as input/processing/output concepts), (b) *Skill level-2* involving seven-mid-range test-items: (like knowing the 3-logic patterns, the repetition question and repetition starting point) and *Skill level-3* representing five-advanced (or procedural knowledge) test-items: (to show the depth of knowledge relating to the repetition conditional statement, the alternate repetition condition and the solution algorithm).

The lowest level test-item involved simple questions designed to elicit verbal declaration skill (knowing the basic rules). The mid-range test-item consisted of tasks which demonstrate intellectual skill (evidence of understanding the concept), while the advanced level test-item contained complicated tasks, which required integrating newly acquired programming concepts, in new situations (like writing a complete algorithm).

## **7.4 Common test-items**

Two medium level test-items were used as common-items, in both the pre and post-test instruments. They were worded the same in both instruments. The only difference between them was in their placement order within the test. Their test-item positions were: seven and eight in the pretest and five and six respectively in the post-test. The later placement in the pretest reflects the emphasis to construct a more achievable instrument for participants with no apparent exposure to programming. In the post-test, these more difficult concepts were placed earlier, with fewer straight-forward questions at the beginning.

## **7.5 Marking strategy**

The first exploratory study employed a dichotomous scoring strategy for both the pre and post-tests. This means the test-item answers were inspected for evidence of correct programming concepts, and deemed as being either right or wrong. However, as the focus had shifted from the broader nature of programming per se, to a more concentrated learning programme on control logic, the test instruments for the second and final experiments employed two types of scoring strategies:

### **dichotomous and partial credit**

Each test-item answer was given a numerical value of either 0, 1, 2, 3, or 4. In cases where the instructional outcome required a clear-cut answer (mostly found in verbal information), the dichotomous test-items were recorded as a 0-value or a 1-value. In cases, where there was a clear-cut division of instructional outcome (mostly found in the mid- and advanced skill ranges), a partial-credit marking strategy was implemented. These partial-credited test-items attracted either the value of: zero, 1, 2, 3 or 4. These values (scores) were then recorded to generate a computerised ASCII data file.

## **8. Innate Environmental Factors**

The bottom level of the research objectives, describes the environmental elements which affect the participation, and thereby, the outcomes of the final experiment. They are identified briefly below as the voluntary participation, and instructional conditions. First, it should be noted that due to the negative experiences of self-report visualization research (Thompson 1990), at no time did the facilitators refer to notions of learning through visualization or imagery.

### **8.1 Voluntary Participation**

To comply with the University's Ethics Committee's requirements, before inviting students to participate, they were issued with a Plain Language Research Description Document, which gave a brief overview of: the purpose for conducting the research, the sequencing of the experiment's stages (tests and instruction period), confidentiality issues, and notification that their participation was strictly voluntary.

In each experiment a small number of participants did not complete all stages of the experiment. The reasons given for not completing ranged from disinterest with the learning content or a lack of identified relevance, to having another appointment elsewhere. These reasons may be rationalised in terms of the requirement for a suitable attitude (Gagne learning capability) towards learning. However, the remaining participants were very keen to finish all stages, thereby demonstrating that their attitude was sufficient to motivate their interest through each stage of the experiment.

## **8.2 Sources of Data**

A total of 276-undergraduate business students participated in the three experiments (Pilot-1 (37); Pilot-2 (45); Final Experiment (194)), with each sample drawn from one particular information technology service subject, which delivers introductory database instruction and caters for a broad range of students. They were enrolled in a Bachelor of Business degree at an Australian University (they were not information technology students). Many of them had no previous computer-programming experience.

The sample was grouped, using their Verbal-Imagery Cognitive Style Analysis (CSA) ratio, to identify pairs of similar cognitive style. These pairs were then split to receive the instructional treatment booklet, with Treatment-1 being the text-plus-textual metaphor format, and Treatment-2 comprising the text-plus-graphical metaphors.

## **8.3 Instructional Conditions**

The instructional conditions involved two distinct components: *learning content* (programming logic flow concepts, and *instructional events* (pre-instructional events - to provide context, and the experiment representing the research data.

A number of changes were made to the sequencing of pre-instruction strategy, prior to conducting Pilot-2, and the final experiment. The learning content and instructional conditions were therefore radically different from Pilot-1.

The delivery method for the instructional events was changed to involve separate components: a one-hour lecture, and a two 2-hour practical tutorial sessions, held one week apart.

Changes were made to the sequencing of pre-instructional events involving the timing of the introductory lecture and Tutorial-1. The introductory lecture was run two to three weeks before the final experiment (Tutorial-2) to reduce the risk from the possible associations made between the new information given during this lecture, and the elaborative effect (Woloshyn, Wood & Willoughby 1994) of the pretest test-items. Consequently, this lengthened the total duration of the experimental components by a fortnight, requiring one month to complete the full study.

### **8.3.1 Pre-instructional events**

Due to the reduction in learning content, it was necessary to present the minimum number of programming concepts in the pre-instructional lecture. This delivery technique equates to a Gagne type of verbal informational session, whereby the lecturer only gives facts and rules. Introducing the concepts of programming in this manner provided the participants with the potential to progress through the lower stages of the CLD (discriminating and generalising equivalents across different contexts) (Klausmeier & Sipple 1980). Subsequent experience of the first practical learning session (Tutorial-1), justified progress to the higher levels of the CLD (Fig:10), thereby taking a wholistic approach to the design of the instructional conditions.

### 8.3.1 (a) One-hour lecture

To enable the participants to work on their own, in both of the subsequent two-hour tutorials, an introductory one-hour lecture was designed, involving: why PASCAL was chosen, data types, PASCAL structure, input/output issues, and a brief outline on control structures.

### 8.3.1 (b) Tutorial-1

The *learning content* for this tutorial was considered as a *pre-instructional event*, and was designed to assist the participants in developing a positive *attitude* towards the programming environment.

Therefore, the instructional components involved: a revision of the lecture material (especially relevant for participants who may have missed the earlier 1-hour lecture), as familiarisation with PASCAL data types, and accessing the PASCAL editor on the University's network.

Instructional Format		
	Treatment 1	Treatment 2
	Text-plus-textual metaphor	Text-plus-graphical metaphor
Cognitive style measure (Verbal-Imagery)		
Verbal	Group 1 (T1csa1)	Group 2 (T2csa1)
Imagery	Group 3 (T1csa2)	Group 4 (T2csa2)

Fig:11 Factorial Design

## 9. Experimental Design

The 2 x 2 factorial design is shown in Fig:11 (Gay 1992). However, ANOVA was considered inappropriate (Izard 1999). The extent of the design effect was unknown. As a result, simple random sample statistics could not be used.

**The CSA was conducted before the experiment. Furthermore, it was decided to use another statistical measurement tool better suited to analyse the test data for cognitive performance. Since the research questions were interested in the magnitude of effect, was used instead of ANOVA (Cohen 1977).**

## **9.1 Measurement**

The cognitive performance was, therefore, measured using the QUEST Interactive Test Analysis System. Given that in both cases, the number of participants and the tested instructional outcomes had increased since the second exploratory study, it was now possible to plan for a more thorough data analysis of the interacting independent variables (instructional format/cognitive style) (Fig:11), with the dependent variable (cognitive performance of the instructional outcomes).

**Central to this research, was be the ability to establish the relationship between cognitive performance and instructional outcome.**

Therefore, both these topics will be discussed next.

### **9.1.1 Cognitive performance**

In order to evaluate the performance of each participant, the QUEST Interactive Test Analysis System (Adams and Khoo 1996) was used. This test measurement application allows for improved analyses of an individual's performance relative to other participants. This program establishes a model, which allows the best estimate of the probability of an individual making a certain response to a test-item – low to high. Rasch test-items are designed to go from easy to difficult, with the test-item difficulties and performance measures are calibrated on the same scale. For a full description of QUEST and its capabilities, see Adams & Khoo (1996), and (Izard 1995).

As measuring cognitive performance was crucial to the experiment, it was necessary to separate out: assessment of cognitive skills, scaling the performance, test review process and assessment validity.

#### **9.1.1 (a) Assessment of cognitive skills**

The classical approach to assessment identifies test-items which do not distinguish between high and low scores, whereas QUEST relies on a type of assessment that is similar to a criterion measure. According to Griffin & Nix (1991), this means an observation is:

*“directly compared to a single, fixed level of performance, or pre-specified criterion, and is interpreted as either mastery or non-mastery”* (Griffin and Nix 1991):264

It is now possible to look at a QUEST Variable Map to determine the behaviour of test-items and participants (or cases), compared to each other. Reviewing the QUEST Item-fit Map enables a designer to check for unexpected results. Furthermore, examination of the QUEST Fit t-values, (obtained from the QUEST Test-Item Estimates Table) empowers a designer to either reject unstable test-items or include additional test-items.

#### **9.1.1 (b) Scaling the performance**

Item Response Theory (IRT) describes the assessment method used by QUEST. It compares actual patterns of responses from the interactions of test-items with

participants compared with a model pattern. This becomes the fit-statistic. Unusual patterns can be investigated. Central to QUEST is a measurement model developed in 1960 by a Danish statistician called Georg Rasch. QUEST develops an uni-dimensional scale with equal intervals along each axis, to measure performance and test-items together. This scaling feature enables a developmental sequence of learning tasks to be arranged from simple to more complex. It is also possible to locate an individual at different skill levels along this scale. The assessment of particular skill levels relies on choosing tasks that can provide evidence, which effectively distinguishes between those who have the required knowledge and those who have not.

### **9.1.1 (c) Test review process and assessment validity**

Test-item analysis results in allowing instructional strategy decisions to be made about the validity of the test-items: either to retain, modify, or discard. This forms the basis of a test blueprint (or specification) of the testing instrument. The associated Test-item Specification Grid (an instructional design tool which places learning tasks and instructional objectives in a matrix) must show enough items remain to give a range of test-item difficulties within each cell (Fig:10). These decisions fall into three categories (Izard 1995): too easy, wrong choice, or not possible, may not discriminate, correct choice too obvious, and may be too difficult, or more than one answer is correct.

### **9.1.2 Instructional outcomes**

Evidence of a weak variation of overall performance emerged from the results of Pilot-2. Consequently, the testing instrumentation was changed again, to include a larger number of test-items in each of the instructional outcome categories. Examination of the Test Instrument Specification Matrix (Fig:10), revealed a number of areas where the combination of instructional objectives and learning content were not included in the earlier testing instruments.

This meant there was a bunching effect of test-items at either end of the programming difficulty scale. For instance, too many basic test-items, which may account for the lack of variance in performance outcomes in the two exploratory studies. Consequently, both the pre and post-tests were increased from 14-test-items in Pilot-2 to 20-test-items in the final exploratory study, to provide for a broader range of instructional outcomes.

### **9.1.2 (a) Validity**

The new test-items were calibrated manually. They were tested on a sample of first year students (not likely to participate in the experiment). The calibration of these new test-items was necessary to ensure consistency with the existing test-items used successfully in Pilot-2. Again there were three-levels of performance on the pre and post-tests: basic (*verbal information*), mid-range (*intellectual skills*) and advanced (*high-level cognitive strategies*). There were six-new test-items added to each test instrument for the final experiment. The programming knowledge levels of these new test-items included: two-basic level, two-mid-range, and two-advanced.

To facilitate the measurement of cognitive performance, some of the test-items were used in both the pre and post-tests. In all, there were 8-common test-items for the final experiment (there were only 2 in Pilot-1 and Pilot-2). These common test-items were designed to flush out specific instructional outcomes that were directly linked to the critical attributes identified in the instructional metaphors. The emphasis of the instructional outcomes on the post-test also shifted to concentrate on the mid-range and advanced instructional (performance) outcomes.

The same combination of dichotomous and partial credit scoring methodology was used. It has been shown that this type of scoring methodology is more appropriate for evaluation of the differences in cognitive performance levels.

## **10. Final Experimental Procedure**

The final experiment was conducted in week-10 of the academic semester. Overall, there were a total of 11-tutorial groups participating. As before, each treatment group was separated for the duration of the experiment. A monitor observed the participants throughout. Due to the voluntary nature of the involvement, if any participant wished to withdraw (for any reason), this was permitted. Consequently, there were a small number (four-participants) who did not complete the full experiment. The attrition was due partly to the voluntary nature of the participation, and to a belief that programming concepts would not be included in the final exam for the subject.

### **10.1 Screening Tests**

It was necessary to categorize the participants according to two characteristics: their levels of prior knowledge of computer programming, and their cognitive style. This categorical data was again gathered before each of the experiment's instructional periods.

#### **10.1.1 (a) Cognitive style**

Within the framework of the internal/external conditions-of-the-learner, the Riding & Cheema (1991) cognitive style construct has proved to be a useful cognitive modelling tool. Therefore, the position of the participants on the Verbal-Imagery and Wholist-Analytic dimensions was identified, using the computer-presented CSA (Riding and Cheema 1991).

The only change to the screening for cognitive style was to alter the location from a private room to the normal classroom. Therefore, the CSA was conducted during each Tutorial-1 session. This was necessary due to the inconvenience of running separate screening tests on a large number of individuals. To further reduce the time for this process, two computers were set up with the CSA software at the back of the classroom (a computer laboratory). Participants were called to the back of the tutorial classroom (two at a time). They took the cognitive screening test, sitting at computers on opposite sides of the room. These computers were positioned such that the rest of the class could not view their screens.

### **10.1.1 (b) Pretest**

This was conducted at the beginning of Tutorial-2 to determine levels of prior domain knowledge. It was carried out in the participants' normal tutorial classrooms. The pretest was constructed using more of the easier pre-trialed test-items. In order to reduce the writing-time, it was decided to pick achievable items for the beginning of the pretest. It was also hoped that this would enhance the likelihood of obtaining meaningful answers from the non-programming participants. Reducing their stress was particularly important, considering this test was administered prior to them receiving the instructional material. The more difficult items were included to identify the experienced programmers.

## **10.2 One-hour lecture**

The lecture was given earlier in the semester (week-7), to reduce the likelihood of creating a ceiling effect (a possible reason for many high performing results on the Pilot-1 and Pilot-2 pretests).

### **10.2.1 Instructional period**

This instructional event involved the running of the actual experiment. At no time were the participants given any explanation as to the likelihood of differences in instructional strategies. They were, however, reminded that the skills gained during the experiment would be transferable to other contexts in other tertiary level subjects.

Although there was no official recording of the test results in the normal assessment for the participants, they were advised that the (general) knowledge they gained would be useful in completing the higher-grades of their database assignment for the subject.

There were no changes made to the instructional content for the final experiment, with the learning content remaining exactly the same as Pilot-2. However, there were changes made to the timing or sequencing of the pre-instructional activities.

## **10.3 Cognitive performance**

Each participant was given the post-test on completion of the self-paced instructional booklet, in Tutorial-2. The conditions for this test were the same as Pilot-2 (see McKay 1999 (b)), with test modifications as described previously. The purpose of this test was to identify improvement in cognitive performance from the pre-test scores obtained prior to the instruction period.

## 10.4 Results

The *QUEST Item Fit Map* (Fig:12 ) provides a *visual check* of magnitude of the *fit statistic* for the *test-items*. Overall, these *test-items* fit the Rasch model, with the exception of *test-item-34*, which falls outside of the left-hand vertical dotted line, indicating a mean square that is 30% below its expected value.

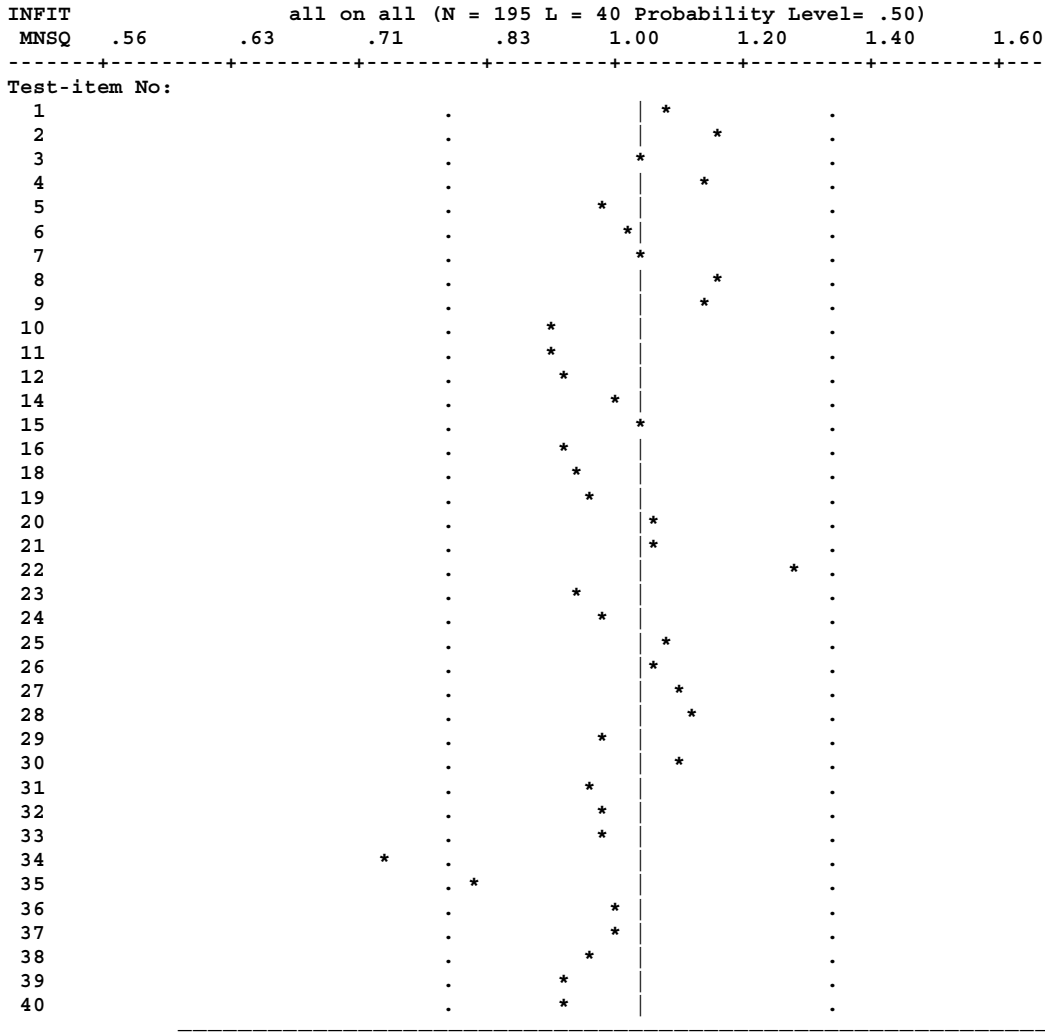


Fig: 12 *QUEST Test-Item Fit Map*

The traditional summary of the *test-item* estimates (thresholds) identified a mean of .00, the standard deviation as 1.40, and the standard deviation (adjusted) 1.40, with separation reliability of 1.00 (Wright and Masters, 1982).

Each *test-item* has been designed to test for specific types of knowledge. In general, the higher the *programming-knowledge* required to correctly answer a question, the higher on the *common-QUEST logit scale (Qls)* the *test-item* will appear.

Fig:13 locates the *cognitive performance* fit statistic for *four participants*. In this example, notice *two Wholist-Imagers* fit the Rasch model (shown as asterisks lying between the two vertical dotted lines), while *two Analytic-Imagers* do not.

```

:studyctl.ct1: OVERALL PERFORMANCE RUN2 : ANCHOR ON
-----
Participant Fit In input Order                                9/ 7/98 16:42
all on all (N = 195 L = 40 Probability Level= .50)
-----
INFIT
MNSQ      .38      .45      .56      .71      1.00      1.40      1.80      2.20
-----+-----+-----+-----+-----+-----+-----+-----+
Participant
  Detail
-----
Input   Group
Order   Code
-----
   6    0106MAI      .      |      .      *
  43    0308MAI  *      .      |      .
 129    0901FWI      .      |      *      .
 185    1202MWI      .      |      *
-----
Group Code: numerics, gender, ICS Sub-Group (AI = Analytic-Imager; WI = Wholist-Imager)
=====

```

Fig:13 QUEST Participant Fit Map

Traditional analysis of the participant (Case) statistical estimates identified a mean of  $-.89$ , standard deviation as  $.53$ , and the standard deviation (adjusted)  $.47$ , with separation reliability of  $.79$  (Wright and Masters, 1982).

It was found that graphical representations of abstract concepts (Fig:9), when added to textual learning material, have a positive effect on learning outcomes (McKay 1999(b)). Moreover, the results from Pilot-1 (illustrated on the left of Fig:14) supported the Bagley hypothesis that novice computer-programmers prefer a structured approach to learning abstract programming concepts, whereas experienced computer-programmers perform best with unstructured material. More specifically, there was strong evidence in both Pilot-1 (with dichotomous scoring), and Pilot-2 (with dichotomous and partial credit scoring), that the best performances were achieved by the Verbalizers using the text-plus-graphics instructional material, and a weaker confirmation that the Imagers performed best using the text-only instructional material (Fig:14).

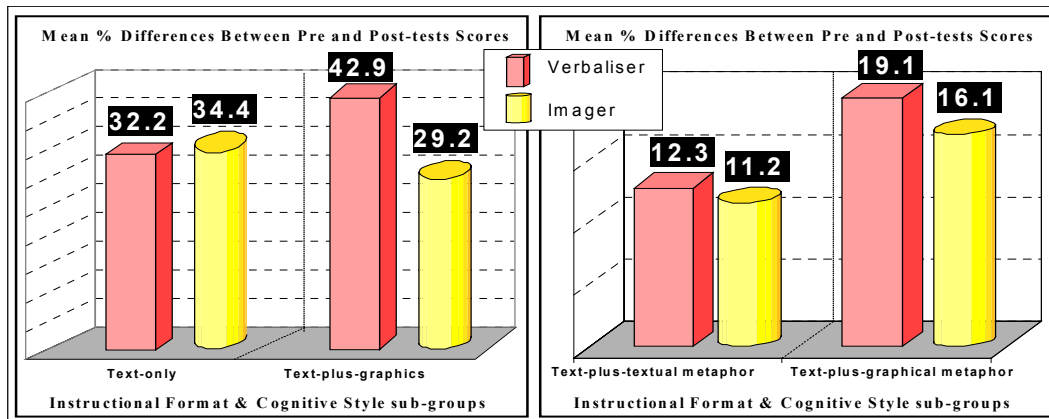


Fig:14 Variance in Cognitive Performance (means)

However, this result contradicts popular belief, that Verbalisers learn best from words and text, and that Imagers learn best from pictures or graphic representations (Douglas and Riding 1993); (O'Halloran and Gauvin 1994). Even though different measurement methodologies were used for both experiments (statistical means for Pilot-1, and the QUEST interactive test analysis system for Pilot-2); the findings were at variance with the expected outcome. Subsequently, to confirm the interactive relationship between instructional format and cognitive style, the third and final experiment was conducted on a large data set (195-participants) (McKay 2000; 2002). The increased sample size facilitated the analysis of the effects of the Wholist-Analytic cognitive style dimension, and the gender variable.

Analysis of the single category of cognitive style (SCCS) groups based on their mean-QUEST difference logit value (Qdlv), indicated that all groups performed better with the graphically enhanced instructional treatment (Treatment-2) (McKay 2000).

Although the effect size (ES) between instructional treatment groups is small ( $\sim 0.2$ ), all analysis measurements show a preference to the Treatment-2 material (top half of Table 1). This result is contrary to predictions (Riding 1980; 1989; 1990). The Analytic and Imager groups although showing only modest improvement with Treatment-2; the Treatment-1 group actually performed lower on the post-test than the pretest relative to the other participants (Fig:15).

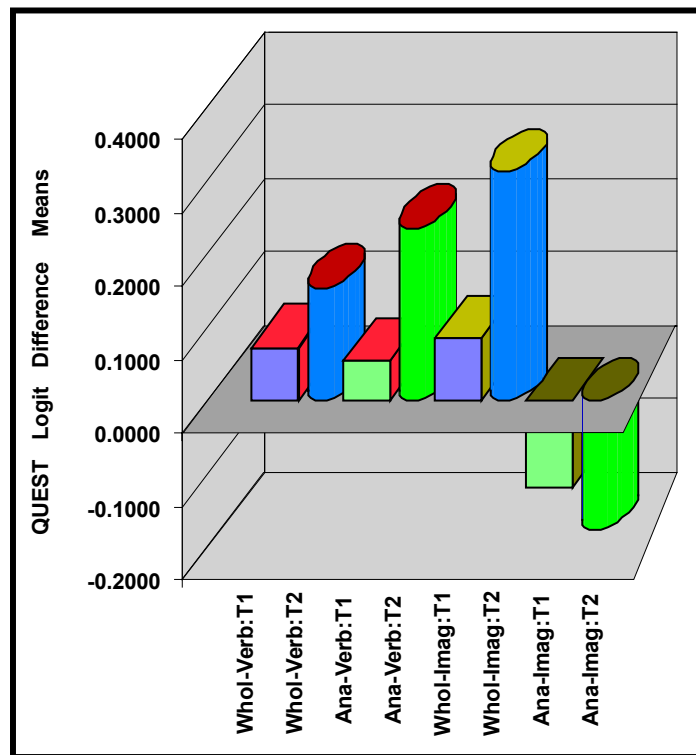


Fig:15 Variance in Cognitive Performance (means)

The QUEST data analysis suggested there were trends of the form: Verbalizers learn abstract concepts differently from Imagers and benefit from the text-plus-graphical metaphor enhanced instructional format. Furthermore, Verbalizers learn how to apply abstract concepts in varied contexts differently from the Imagers. They appear to be better at transferring their conceptualisation's of the graphical representations into new computing contexts. Therefore, Verbalizers would learn new programming concepts better with a text-plus-graphical metaphor than the text-plus-textual metaphor material.

Given the relatively poor performance of the Analytic and Imager SCCS groups, it is not surprising that the Analytic-Imager ICS sub-groups performed badly. Their mean-Qdlv's (Fig:15) are both negative with the Treatment-2 sub-group (-0.175) obtaining a lower value than their Treatment-1 counterparts (-0.118). This would suggest that an Analytic-Imager would perform best with the textual treatment.

## 11. Summary

The experimental design focussed on the strategic knowledge requirements for the acquisition of complex programming concepts in a computer-mediated environment. Major influencing factors were shown to be: type of content, existing knowledge, and the students' information processing ability. This design validates and exemplifies the three level knowledge acquisition framework deduced from the selected research methodology.

The first level, defined the research process necessary to construct the context for conducting the study. The scope of the research was threefold: to determine the specific knowledge requirements for the acquisition of programming concepts, to identify the cognitive strategies employed by novice-programmers, and to develop the theoretical foundations to support a model for describing meta-knowledge acquisition, in particular for the acquisition of complex programming concepts. The key research parameters were shown to involve two distinct components involving the external-conditions-of-the-learner, namely: the method of delivery (relating to the learning content, instructional conditions, presentation mode, and specific learner characteristics), and the measurable instructional outcomes (MIOs) (relating to the instructional method and cognitive style construct).

The second level specified the nature of the variables involved in the interaction of cognitive style and instructional strategies and described the research parameters for the acquisition of programming knowledge. Sampling adequacy of the learning content was pre-determined to ensure content validity. The instructional mechanisms employed to draw out the specific programming performance expectations were shown to be grounded in instructional design principles (Klausmeier & Sipple 1980, Gagne 1985, Merrill et al. 1992). A concept attainment model (Fig:8), was devised to formalise the range of programming knowledge performances, as MIOs (Fig:10). The QUEST Interactive Test Analysis System provided a measuring tool. The Rasch measures do not eradicate all errors of measurement; they reduce the noise. Therefore, reliability of the test instrument is assured through the calibration techniques utilized by the QUEST estimate (separation index for the test-items was 1.00, while the participant (cases) was .79), thereby validating the statistical basis of the research design. Replication of the results is possible through the relative absence of the distortions otherwise encountered with observational techniques.

The third level described the final experiment, setting out the limitations of the environmental factors which may affect results, including: voluntary participation, sources of data, instructional conditions, experimental design, and measurement.

## Reference List

- Adams, R. J. and Khoo, S.-T. (1996). *Quest: The Interactive Test Analysis System*. Melbourne, Australia, Australian Council for Educational Research.
- Bagley, C. A. (1990). Structured Versus Discovery Instructional Formats for Improving Concept Acquisition by Domain-Experienced and Domain-Novice Adult Learners. *Faculty of the Graduate School*. Minnesota.
- Cohen, J. (1977). *Statistical Power Analysis for the Behavioral Sciences*. New York, Academic Press.
- Dick, W. and Carey, L. (1990). *The Systematic Design of Instruction*. Florida, Harper Collins.
- Douglas, G. and Riding, R. J. (1993). The Effect of Pupil Cognitive Style and Position of Prose Passage Title on Recall. *Educational Psychology* 3(3 & 4): 385-393.
- Gagne, R. M. (1985). *The Conditions of Learning: And the Theory of Instruction*. NY, Holt/Rinehart/Winston.
- Gay, L. R. (1992). *Educational Research: Competencies for Analysis and Application*. New York, MacMillan.

- Griffin, P. and Nix, P. (1991). *Educational Assessment and Reporting: A New Approach*. Sydney, Harcourt Brace.
- Hennefeld, J. (1989). *Using Turbo Pascal 3.0, 4.0, and 5.0*. Massachusetts, PWS-Kent.
- Howell, A. D. (1972). An Electroencephalographic Comparison of Lowenfeld's Haptic Visual and Witkin's Field-Dependent-Independent Perceptual Types. Indiana, Ball State University.
- Hummel, J. E. and Holyoak, K. J. (1997). Distributed Representations of Structure: A Theory of Analogical Access and Mapping. *Psychological Review* 104(3 July): 427-466.
- Izard, J. (1995). Trial Testing and Item Analysis. Melbourne, International Institute of Educational Planning.
- Izard, J. (1999). Some Potential Difficulties in Educational Research Studies (and How to Avoid Them), Paper Written for the Third Elementary Education Project, Philippines.
- Klausmeier, H.J. & Sipple, T.S. (1980). *Learning and Teaching Concepts: A Strategy for Testing Applications of Theory*. NY, Academic Press.
- Lukose, D. (1992). Goal Interpretation as a Knowledge Acquisition Mechanism. *Faculty of Science and Technology, School of Computing and Mathematics*. Deakin University, Melbourne.
- Mager, R. E. (1988). *The New Six Pack : Making Instruction Work*. California, Lake.
- McKay, E. (1999)(a). An Investigation of Text-Based Instructional Materials Enhanced with Graphics. *Educational Psychology* 19(3): 323-335.
- McKay, E. (1999)(b). Exploring the Effect of Graphical Metaphors on the Performance of Learning Computer Programming Concepts in Adult Learners: A Pilot Study. *Educational Psychology* 19(4): 471-487.
- McKay, E. (2000). Instructional Strategies Integrating the Cognitive Style Construct: A Meta-Knowledge Processing Model (Contextual Components That Facilitate Spatial/Logical Task Performance): An Investigation of Instructional Strategies That Facilitate the Learning of Complex Abstract Programming Concepts through Visual Representation. *Applied Science (Computing and Mathematics Department)*. Waurm Ponds, Geelong, Australia, Deakin University: 3 Volumes.
- McKay, E. (2002). Cognitive Skill Acquisition through a Meta-Knowledge Processing Model. *Interactive Learning Environments* 10(3): 263-291 <http://www.szp.swets.nl/szp/journals/il103263.htm>.
- Merrill, M. D. (1994). *Instructional Design Theory*. New Jersey, Educational Technology Publications.
- Merrill, M.D., Tennyson, R.D. & Posey, L.O. (1992). *Teaching Concepts: An Instructional Design Guide*. NJ, Educ. Tech. Pubs.
- Moore, D. M. and Bedient, D. (1986). Effects of Presentation Mode and Visual Characteristics on Cognitive Style. *Journal of Instructional Psychology* 13(1).
- O'Halloran, A. M. and Gauvin, L. (1994). The Role of Preferred Cognitive Style in the Effectiveness of Imagery Training. *International Journal of Sport Psychology* 25(1 January-March): 19-31.
- Perkins, D. N. (1991). Technology Meets Constructivism: Do They Make a Marriage? *Educational Technology* (May): 18-23.
- Preece, J. (1994). *Human-Computer Interaction*. Harlow, England, Addison-Wesley.
- Reigeluth, C. M., Ed. (1983). *Instructional-Design Theories and Models: An Overview of Their Current Status*. NJ, Erlbaum.
- Riding, R. & Cheema, I. (1991). Cognitive Styles - an Overview and Integration. *Educational Psychology* 11(3&4): 193-215.
- Riding, R. J. and Caine, R. (1993). Cognitive Style and Gcse Performance in Mathematics, English Language and French. *Educational Psychology* 13(1): 59-67.
- Riding, R. J. and Rayner, S. (1998). *Cognitive Styles and Learning Strategies*. United Kingdom, Fulton.
- Robertson, L. A. (1994). *Simple Program Design*. Melbourne, Nelson.
- Romiszowski, A. J. (1981). *Designing Instructional Systems*. United Kingdom, Kogan Page.
- Salomon, G. (1979 Reprint 1985). *Interaction of Media, Cognition, and Learning*. California, Jossey-Bass.
- Savitch, W. J. (1989). *Turbo Pascal 4.0/5.0: An Introduction to the Art and Science of Programming*. California, Benjamin/Cummings.
- Thompson, S. V. (1990). Visual Imagery: A Discussion. *British Journal of Education Psychology* 10(2).
- Tversky, A. (1977). Features of Similarity. *Psychological Review* 84(4 July): 327-352.
- Wileman, R. (1993). *Visual Communicating*. New Jersey, Educational Technology Publications.
- Winn, W. (1982-a). Visualization in Learning and Instruction: A Cognitive Approach. *ECTJ* 30(1): 3-25.
- Woloshyn, V. E., Wood, E. and Willoughby, T. (1994). Considering Prior Knowledge When Using Elaborative Interrogation. *Applied Cognitive Psychology* 8(1)(Feb): 25-36.
- Wright, B. D. and Masters, G. N. (1982). *Rating Scale Analysis: Rasch Measurement*. Chicago, MESA Press.
- Yost, J., Varecka, A. F. and Marefat, M. M. (1997). Content-Based Visualization for Intelligent Problem-Solving Environments. *International Journal of Human Computer Studies* 46(4 April): 409-441.
- Zemke, R. and Kramlinger, T. (1982). *Figuring Things Out: A Trainer's Guide to Needs and Task Analysis*. Massachusetts, Addison-Wesley.