CHAPTER 3

Methodology

Rationale

The Information Processing and Technology (IPT) syllabus for Senior students in Queensland (BSSSS 1991, 12) specifies that an aim of the Algorithms and Programming topic is to cultivate software development expertise in students with a focus on the solution of practical problems. It is also contended that secondary students of information technology courses should be involved in exciting and stimulating developmental work (Clarke 1992, 6; Newlands and Teague 1993, 15).

This study seeks to establish an effective way of achieving the above objectives by supplementing students' programming projects with a sophisticated user interface within either a procedural or an object-oriented programming environment. The traditional style of procedural programming is complemented with object-oriented programming because the evolving complexity of software interfaces makes object-oriented programming a necessity rather than an option (Cox 1986, 28; Martin 1993, 18).

The effectiveness of achieving the above objectives is to be measured within a framework of school learning as proposed by Bloom. Bloom (1976) accounts for variation in student achievement within a model of three interdependent variables. These variables are the provision of instruction, the treatment in this study; student cognitive entry behaviours, represented by general cognitive skills

within this study; and student affective characteristics which, within this study, are partially represented by student attitude towards programming and student persistence .

Research questions

The overall objective of the thesis

The research question may be broadly stated as: How do Year twelve students' achievement in procedural programming and achievement in object-oriented programming compare. The comparison being made on the basis of cognitive skill variables selected within a framework of school learning as proposed by Bloom (1976). A subsidiary question is whether the addition of a sophisticated user interface to students' programming projects improves students' attitudes towards programming?

Definition of dependent and independent variables

Independent variables

The instructional treatment - attaching a sophisticated user interface to students' projects within a procedural programming environment or within an object-oriented programming environment.

Learning style - the composite of characteristic cognitive, affective, and psychological factors that serve as relatively stable indicators of how a learner perceives, interacts with, and responds to the learning environment (Keefe and Monk 1990, 1). The appropriate factors are defined in table 3.

Dependent variables

Achievement (posttest only).

Attitude to(wards) programming (pretest and posttest).

Analytic skill	To identify simple figures hidden in a complex field; to use the critical element of a problem in a different way.
Spatial skill	To identify geometric shapes and rotate objects in the imagination; to recognise and construct objects in mental space.
Discrimination skill	To visualise the important elements of a task; to focus attention on required detail and avoid distractions.
Categorisation skill	To use reasonable versus vague criteria for classifying information; to form accurate, complete, and organised categories of information.
Sequential processing skill	To process information sequentially and verbally; to readily derive meaning from information presented in a step-by-step, linear fashion.
Simultaneous processing skill	To grasp visuospatial relationships; to sense an overall pattern from the relationships among component parts.
Memory skill	To retain distinct versus vague images in repeated tasks; to detect and remember subtle changes in information.
Persistence orientation	Willingness to work at a task until completion.

Table 3. The relevant subscales of the learning style profile (Keefe and Monk 1990, 5).

Hypotheses

1 Student achievement in adding a sophisticated user interface within a familiar procedural programming environment will be significantly greater than student achievement in adding a sophisticated user interface within a relatively unfamiliar object-oriented environment.

2 A significant positive relationship exists between each of the cognitive skill subscales of the Learning Style Profile, and students' achievement within each of the programming environments.

3 The addition of a sophisticated user interface to students' programming applications will improve students' attitude towards programming.

4 The magnitude of improvement in students' attitudes towards programming following the addition of a sophisticated user interface will not differ significantly between the procedural and object-oriented environment groups.

Experimental design

Subjects

The students involved in the study were all forty-eight Year twelve students in a rural high school studying the subject 'Information Processing and Technology' (IPT). The forty-eight IPT students were part of a Year twelve population of one hundred and twenty students. These students, at the end of Year ten, selected IPT from an offering of the 'Board approved' subjects IPT, Music and Biology, and school subjects Manual Arts A and Introduction to Catering. The majority of tertiary-oriented Year twelve students study IPT. Two teachers teach Year twelve IPT and work together as a cooperative unit.

Instructional resources

Two computer laboratories and a large open space teaching room were available. The licensed programming environments accessible at the school were Turbo Pascal 7.0 (Borland 1992), Turbo Vision - an object library (Borland 1992), TechnoJock's Object Toolkit - an object library (TechnoJock Software 1991) and TechnoJock's Turbo Toolkit - a procedural toolkit (TechnoJock Software 1989).

It was decided to use the two TechnoJock toolkits. Both toolkits, being devised by the same company and used in conjunction with Turbo Pascal, have similar philosophical structure to the extent of having parallel nomenclature for source

files. The essential difference between the two toolkits is that one is procedural and the other is object-oriented. The use of other user interface libraries may have introduced an extraneous variable into the study as a result of philosophical differences in the ways in which the libraries have been structured. The use of the TechnoJock toolkits also allows the avoidance of event-driven programming. Event-driven programming is a mandatory paradigm within the use of Object Windows Library and is a complication which is better avoided within the context of a study investigating the possible benefits of object-oriented programming for secondary students.

Explicit instruction involving the teaching of how to design application solutions, the provision of templates and the modelling of solution strategies, was used throughout the entire unit. This approach, as opposed to unguided discovery classes, is supported by a number of sources (Doyle 1983, Eylon and Helfman 1985, Dalbey and Linn 1985).

Instructional sequence and data collection

Because this study was conducted within the researchers' own institution, access was automatic and data collection was essentially unobtrusive.

Research indicates that the intrinsic motivational characteristics of a topic which adds value to learning are difficult to implement in a typical classroom environment (Lepper and Greene 1978; Deci and Ryan 1985; Malone and Lepper 1987). Good and Brophy (1991, 298) suggest that this difficulty is due to the fact that the whole curriculum must be taught, not just the appealing parts, and that while intrinsic factors should increase students' enjoyment of the activities, there is also a need to stimulate student motivation to learn - students might enjoy the task but not learn the appropriate knowledge and skills. This difficulty may be alleviated to some extent by firstly completing a course on procedural programming before introducing object-oriented programming concepts. It is

therefore expected that students will have experienced the basic knowledge and skills such as syntax and modularity concepts of procedures as shown by the schedule in table 4.

27-2-95 (8 weeks) 28-4-95	Introduction to programming. Control structures (sequence, selection, iteration and modularity). Data types (integer, real, boolean, character and string). Data structures (arrays, files and objects).
2-5-95 (7 weeks) 19-5-95 16-6-95	Software development applications (BSSSS 1991, 12). Student learning experiences involve observing, analysing and modifying existing solutions as well as developing solutions (BSSSS 1991, 15). Examples of applications are: School athletics records, overdue library books, video store, ticket booking office, and baby sitting service. Administration of Learning Style Inventory. Summative end of unit assessment. Attitude pretest.
3-7-95 (4 weeks) 28-7-95 1-8-95	**The study** One randomly assigned group receives instruction and laboratory experience on software development supplemented with an object-oriented toolkit. The other randomly assigned group receives instruction and laboratory experience on software development supplemented with a procedural toolkit. Students are presented with a solution to 'school athletics records' which now incorporates a sophisticated user interface. Students receive instruction, observe, analyse and modify the solution. Students also develop solutions to similar problems. Administration of the achievement instrument. Administration of attitude to(wards) programming posttest.
2-8-95 (4 weeks) 1-9-95	Repeat of the treatment Each group now receives the alternative treatment. The achievement instrument is again applied.

Table 4. The algorithms and programming unit schedule.

The relationship between data collection episodes and instructional sequences is also outlined in table 4. All data collection was scheduled to occur during normal lessons in the morning and under supervised conditions. The achievement instrument and the attitude to(wards) programming instrument were administered by typical school assessment procedures. The learning style profile instrument was administered as per instructions (Keefe and Monk 1990, 7). The experimental design is summarised in Figure 6.

The experimental design is summarised in Figure 6.

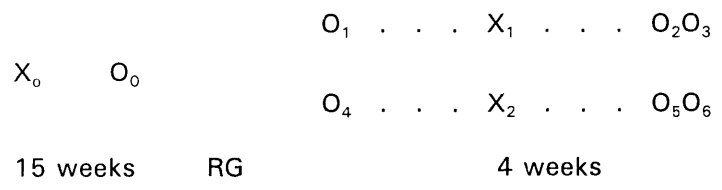

Figure 6. The experimental design.

X_0 All students were taught the normal algorithms and programming course as stipulated in the syllabus (BSSSS 1991, 12-16). The syllabus specified a maximum of twenty-five weeks be allocated to the Algorithms and Programming unit within IPT (BSSSS 1991, 5). IPT was timetabled at eleven forty minute periods per fortnight.

The first eight weeks were intended to introduce students to control structures (sequence, selection, iteration and modularity), data types (integer, real, boolean, character and string) and data structures (arrays, files and objects).

The next seven weeks involved application, within the software development cycle (BSSSS 1991, 12), to the management of school athletics records, overdue library books, video store, ticket booking office, and baby sitting service. Student learning experiences involved observing, analysing and modifying existing solutions. Students also developed solutions to similar applications (BSSSS 1991, 15).

O_0 All students sat for the formal summative end of unit achievement test. This achievement test was subject to regional review as part of the normal certification process. The regional IPT review panel, composed of practising IPT teachers, reviewed the school's assessment instruments and student responses with a view to monitoring comparability of student achievement. The achievement test is presented in Appendix B.

RG Random assignment of all forty-eight students to two groups of equal size.

O_1 O_4 The attitude to(wards) programming instrument was administered to all students as a pretest.

X_1 One group of students (24) received four weeks of further instruction and laboratory experience on software development supplemented with an object-oriented toolkit. Students were presented with a solution to 'school athletics records' which used the object-oriented toolkit to produce an acceptable user interface. The code is presented in Appendix C. Students received instruction, observed, analysed and modified the solution. Students also developed solutions to similar applications.

X_2 The other group of students (24) received four weeks of further instruction and laboratory experience on software development supplemented with a procedural toolkit. Students were presented with a solution to 'school athletics records' which used the procedural toolkit to produce an acceptable user interface. The code is presented in Appendix C. Students received instruction, observed, analysed and modified the solution. Students also developed solutions to similar applications.

O_2 Under supervised exam conditions, students modified and adapted the 'school athletics records' solution to provide a solution to a 'property rentals' application. The object-oriented toolkit was used to produce a similar interface.

O_5 Under supervised exam conditions, students modified and adapted the 'school athletics records' solution to provide a solution to a 'property rentals' application. The procedural toolkit was used to produce a similar interface.

O_3 O_6 The attitude towards programming instrument was again applied to all students as a posttest.

Threats to validity

Contamination and experimenter bias (Gay 1981, 219) was a definite possibility within this study because the researcher was the instructor of both the object-oriented treatment, X_1, and the procedural treatment, X_2. The rotation of each group through instruction and laboratory on each successive lesson may have provided some control while supported by a conscious effort to provide similar learning materials and similar learning experiences to both groups.

Internal threats to validity

Internal threats to validity arose from the influence of variables other than the independent variable on the dependent variable. Internal validity is a measure of the extent to which the results of the study are attributable to manipulation of the independent variable (Gay 1981, 211).

History. Interruption by school events was minimal. If a class period was lost then the rotation of the groups through instruction and laboratory was still able to be continued in the following class period ensuring that one group did not receive more time or less time than the other group. Critical events, such as the absence from class of the instructing teacher, did not occur.

Testing. The period of study was only four weeks and thus testing effects may have been a problem. The attitude to(wards) programming variable used the same test in both the pretest and the posttest and thus the testing threat will need to be considered during the analysis of the results. The achievement variable was measured by posttest only and thus was less susceptible to the testing threat to internal validity. The nature of continuous assessment within IPT promoted the view amongst students that testing was a naturally occurring event and thus was not likely to be noticed.

Maturation. The short period of the study, four weeks, should have hopefully controlled this threat to internal validity.

Instrumentation. The pretest and posttest for the attitude variable used the same measuring instrument and was applied to both groups at the same time. Similar conditions were applied for the posttest as for the pretest, same time of the day and day of the week. The achievement variable was treated as a posttest only and both O_2 and O_5 were almost identical in format.

Statistical regression. Because all students were randomly assigned to each of the two groups and not assigned on the basis of extreme pretest scores, it was expected that the threat posed by the tendency for regression would be controlled.

Differential selection of students. The random assignment of all IPT students to each of two groups controlled this threat to internal validity.

Mortality. No student dropped out of the subject within the short period of the study (four weeks).

Selection-maturation interaction, etc. The random assignment of all students to each of the two groups combined with the period by period rotation of each group through the instructing teacher and the laboratory supervising teacher controlled interactional threats to internal validity.

External threats to validity

External threats to validity arise from the inability to generalise results of the experimental study to situations outside of the experimental setting (Gay 1981, 212).

Pretest-treatment interaction. The pretest attitude to(wards) programming instrument combined with the short period, four weeks, of the treatment may have had some impact upon external validity. It was hoped that students will make little connection between the attitude instrument and the use of a procedural/object-oriented toolkit to enhance user interface appearance. The achievement instrument, being a posttest only, was expected to have little impact on this threat to external validity.

Selection-treatment interaction. The random assignment of students to each of the two groups was expected to provide some control of this threat. Credibility in being able to generalise the results of the study to the population of IPT students in Queensland was supported by the Queensland Core Skills Test (QCS). The QCS test, used for scaling purposes in producing tertiary entrance scores, suggested that the school was not atypical.

1995	number	mean	sdeviation
Stanthorpe SHS (all students)	74	134.2	29.0
State (all students)	27 425	131.2	30.0
Stanthorpe IPT students	47	137.5	31.7
State IPT students	3 230	138.2	28.1

Table 5. QCS comparison of school and State IPT students (BSSSS 1995).

Reactive arrangements. Control for the Hawthorne effect was attempted by providing similar materials and learning situations to each group, alternating the groups after the four week study so that both groups experienced both treatments, making all students aware of the work program at the beginning of the twenty-one week algorithms and programming unit, and by the students not knowing which section of the unit formed the study.

Multiple-treatment interference. The same group of students received each treatment in successsive four week blocks. This multiple treatment was applied more for reasons of equity than as condition of the study. The study comprised

only the first instance in which each group received one treatment. This single treatment should have helped to control for multiple-treatment interference.

Control of extraneous variables (Gay 1981, 220; Wiersma 1991, 93)
Teacher. Control of this extraneous variable was attempted by holding the condition constant for each group. The same teacher provided instruction to the procedural group, G_1, and to the object-oriented group, G_2. The other teacher supervised the laboratory activity of each group. Each group was rotated through instruction for one class period and through the computer laboratory for the successive class period.

Ability level. Students were randomly assigned to the two groups. If the groups were not essentially the same on the dependent variables then posttest scores will be analysed using analysis of covariance (Gay , 229). Gay (1981, 229) makes the point that there is no real advantage to random assignment of matched pairs to control for such variables as gender and ability. The variables are better controlled using other procedures such as analysis of covariance.

Gender of student. Students were randomly assigned to two groups. Analysis of covariance will be used if each group was not essentially the same on this extraneous variable.

Learning materials. Each group received similar learning conditions, similar materials, and teaching presented in a similar manner at similar times. TechnoJock toolkits for each of the procedural (TechnoJock's Turbo Toolkit 1989) and object-oriented (Technojock's Object Toolkit 1991) environments were chosen because of the similarity of the language and user interface appearance.

School. Reduced to a constant because students of only one school were included.

Data gathering instruments

Learning style profile

The learning style profile was produced by the National Association of Secondary School Principals (NASSP) (Keefe, Monk, Letteri, Languis, Dunn 1989). Learning style was defined by Keefe and Monk (1990, 1) as

> the composite of characteristic cognitive, affective, and psychological factors that serve as relatively stable indicators of how a learner perceives, interacts with, and responds to the learning environment. It is demonstrated in that pattern of behaviour and performance by which an individual approaches educational experiences. Its basis lies in the structure of neural organisation and personality which both moulds and is moulded by human development and the learning experiences of home, school, and society.

The learning style profile (LSP) contains 126 items measuring 24 subscales. Instrument validity is supported by the assertion that due attention has been given to face and content validity (Keefe and Monk 1990, 3). Construct validity of the LSP was established by the production of position papers and the extensive use of factor analysis to produce '24 relatively independent scales that assess elements of learning style' (Keefe and Monk 1990, 4). Concurrent validity was established for a number of the subscales (Keefe and Monk 1990, 4) by significant correlation of the LSP Analytic Skill subscale with the Group Embedded Figures Test (Witkin, Oltman, Raskin and Karp 1971); the LSP Perceptual Response subscales with the Edmund Learning Style Identification Exercise (Reinert 1980); the majority of the LSP Preference subscales with the Dunn, Dunn, and Price Learning Styles Inventory (Dunn, Dunn, and Price 1974). A case for concurrent validity of the LSP Simultaneous Processing Skill subscale is also proposed by comparison with the Kaufman Assessment Battery for Children (Kaufman and Kaufman 1983).

The LSP subscales relevant to the research question are the cognitive skill subscales (analytic, spatial, discrimination, categorisation, sequential processing, simultaneous processing and memory) and the Persistence orientation subscale. The instrument, containing some low reliability measures, may be stabilised by using group measures rather than individual measures (Cook and Campbell 1979, 43). For individual students the instrument has a diagnostic purpose and is better supplemented with more comprehensive style inventories (Keefe and Monk 1990, 6).

Keefe and Monk (1990, 12) suggest that, with the exception of categorisation skill and persistence orientation, analysis of the cognitive skills be based on the use of three ordinal groupings. Groupings of low, less than one standard deviation below the mean; average, within one standard deviation above and below the mean; and high, more than one standard deviation below the mean. Categorisation skill and persistence orientation are essentially ordinal but could be treated as interval because of the large number of levels. The LSP technical manual (Keefe and Monk 1988) provides norms for Year twelve students.

Analytic skill. The LSP items 25 to 29 comprise the analytic skill subscale and these items are similar to items found in the Group Embedded Figures Test (Witkin, Oltman, Raskin and Karp 1971). The items typically ask to select a correct form which is hidden within a complex figure. The internal consistency reliability (Cronbach's alpha) is reported as 0.56 with a ten day test-retest reliability of 0.54 (Keefe and Monk 1990, 3).

Responses to the five items are scored as correct or incorrect. The analytic skill measure is essentially ordinal ranging from analytic, a score of 5, to non-analytic, a score of 0.

Spatial skill. The LSP items 36 to 40 comprise the spatial skill subscale. The items typically ask for the number of squares in a figure or to mentally fold and

rotate a sheet of paper. The internal consistency reliability (Cronbach's alpha) is reported as 0.60 with a ten day test-retest reliability of 0.77 (Keefe and Monk 1990, 3).

Responses to the five items are scored as correct or incorrect. The spatial skill measure is essentially ordinal ranging from strong spatial skills, a score of 5, to weak spatial skills, a score of 0.

Discrimination skill. The LSP items 7 to 11 comprise the discrimination skill subscale. The items typically ask to compare the size of the sample circle with the size of other circles around it, without measuring the circles. The internal consistency reliability (Cronbach's alpha) is reported as 0.51 with a ten day test-retest reliability of 0.53 (Keefe and Monk 1990, 3).

Responses to the five items are scored as correct or incorrect. The discrimination skill measure is essentially ordinal ranging from strong discrimination skills, a score of 5, to weak discrimination skills, a score of 0.

Categorisation skill. The LSP items 17 to 24 comprise the categorisation skill subscale. The following item is typical: About 300 new comic books have been written each year for the last 30 years. What do you think is the largest number of comics to be written in any one year during this time?

A. 380 comics C. 870 comics

B. 495 comics D. 620 comics

This item is scored as A = 0, B = 1, C = 3 and D = 2. The eight items thus produce a categorisation skill score ranging from 0, weak or broad categorisation skill, to 24, strong or narrow categorisation skill. The internal consistency reliability (Cronbach's alpha) is reported as 0.74 (Keefe and Monk 1990, 3).

Sequential processing skill. The LSP items 1 to 6 comprise the sequential processing skill subscale. The items compare two puzzles, made up of a number of shapes, and ask for the one shape that is missing from each puzzle. The internal consistency reliability (Cronbach's alpha) is reported as 0.72 with a ten day test-retest reliability of 0.54 (Keefe and Monk 1990, 3).

Responses to the five items are scored as correct or incorrect. The sequential processing skill measure is essentially ordinal ranging from strong sequential processing skills, a score of 6, to weak sequential processing skills, a score of 0.

Simultaneous processing skill. The LSP items 12 to 16 comprise the simultaneous processing skill subscale. The items show a complicated geometrical shape and ask which one of four shapes actually comes from the given shape. The internal consistency reliability (Cronbach's alpha) is reported as 0.86 (Keefe and Monk 1990, 3).

Responses to the five items are scored as correct or incorrect. The simultaneous processing skill measure is essentially ordinal ranging from strong simultaneous processing skills, a score of 5, to weak simultaneous processing skills, a score of 0.

Memory skill. The LSP items 109, 110, 112, 114, 116, 118, 119, 120, 121, 123, 124 and 126 comprise the memory skill subscale. The items typically present a picture and then after turning a page, ask whether the next picture is the same or different. The internal consistency reliability (Cronbach's alpha) is reported as 0.62 with a ten day test-retest reliability of 0.58 (Keefe and Monk 1990, 3).

Responses to the twelve items are scored as correct or incorrect. The memory skill measure is essentially ordinal ranging from strong memory skills, a score of 12, to weak memory skills, a score of 0.

Persistence orientation. The LSP items 68, 74, 84 and 91 comprise the persistence orientation subscale. The items are illustrated by (Keefe, Monk, Letteri, Languis and Dunn 1989, 16):

68. The harder the problem, the more likely I am to give up.
A. Always B. Usually C. Sometimes D. Rarely E. Never

This item is scored A = 5, B = 4, C = 3, D = 2 and E = 1. The four items thus produce a persistence orientation score ranging from 4, high persistence orientation, to 20, low persistence orientation. The internal consistency reliability (Cronbach's alpha) is reported as 0.67 with a ten day test-retest reliability of 0.65 (Keefe and Monk 1990, 3).

Attitude to(wards) programming scale

One of the research questions contained within this study is: Will students exhibit a better attitude to programming by been able to produce applications with sophisticated user interfaces? Measurement of the dependent variable, attitude, therefore required the development of an attitude to(wards) programming scale. It was decided to use a maximum of three constructs, these being the *liking of programming, difficulty of programming, and usefulness of programming*. These three constructs are linked to the research question and have been successfully constructed within a general computing environment (Sutton 1991; Kay 1993). The attitude to(wards) programming instrument is shown in Appendix E.

Design of the pilot instrument was essentially a balance between the number of items and instrument reliability. Increasing the number of items increases the reliability, but at the expense of having a longer test (Nunnally 1967, 223; Carmines and Zeller 1979, 45; Henerson, Morris, and Fitz-Gibbon 1987, 152). Reducing the number of items, and thus the test length, generally reduces the

reliability. Henerson, Morris, and Fitz-Gibbon (1987, 154) indicate that for attitude measurement a reliability of above 0.7 is desirable. While reliabilities below 0.7 are tolerated, this affects the confidence of decisions arising from the measurements. Carmines and Zeller (1979, 51) believe that if a scale is to be widely used then reliability should not be below 0.8.

Henerson, Morris, and Fitz-Gibbon (1987, 133) describe *instrument reliability* as being essentially a question of whether the instrument yields consistent results, that each administration of the instrument yields essentially the same results. Henerson, Morris, and Fitz-Gibbon (1987, 134) then appends the statement that demonstrating reliability does not prove validity. The question of validity of this instrument has been previously addressed. Henerson, Morris, and Fitz-Gibbon (1987, 134) indicate that an instrument demonstrated to be valid is likely to be reliable given the assumptions of the attitude itself being stable and that the respondents' answers have not being influenced by other unforseen factors.

Reliability, the extent to which an instrument produces consistent results, is affected by a number of factors. Henerson, Morris, and Fitz-Gibbon (1987, 147) indicate that the following sources of unpredictable errors should be considered:

> Temporary differences among respondents. These changes may be caused by illness, tiredness, emotional disturbances and affect the mood or attentiveness of the respondent (Henerson, Morris, and Fitz-Gibbon 1987, 147).
>
> Differences in the administration of the instrument. The differences may range from varying directions to respondents, test conditions and interfering distractions such as outside noise (Henerson, Morris, and Fitz-Gibbon 1987, 147).

Variations in the interpretation of results and errors in calculating scores (Henerson, Morris, and Fitz-Gibbon 1987, 147)

Random responses as respondents guess or mark responses without trying to understand items (Henerson, Morris, and Fitz-Gibbon 1987, 147).

The literature describes a number of methods of demonstrating reliability. Within the context of the current attitude to(wards) programming instrument, the test-retest reliability method would appear to be the most appropriate (Henerson, Morris, and Fitz-Gibbon 1987, 148.)

Ebel and Frisbie (1986, 76) indicate that the difficulties with test-retest and equivalent-forms of reliability coefficients have led to the contemplation of other methods of assessing reliability such as a method of internal analysis. A variation of the split half method, a method of internal analysis, is a statistical approach which is the average correlation obtained by calculating correlations for all possible split-half combinations. Such an average correlation coefficient is the coefficient alpha (Ebel and Frisbie 1986, 78) which is applicable for 'attitude scales that provide responses such as strongly agree and strongly disagree with intermediate response options'. The *liking programming subscale*, items 1 to 10, has an alpha of 0.95, the *programming difficulty subscale*, items 11 to 17, has an alpha of 0.91, and the *programming usefulness subscale* has an alpha of 0.88.

Henerson, Morris, and Fitz-Gibbon (1987, 133) describe *instrument validity* as being essentially a question of 'whether the instrument is giving the true story'. This definition is in close agreement with that proposed by Carmines and Zeller (1979, 12), that the validity of an instrument is established if it does what it is intended to do. Nunnally (1967, 76) and Henerson, Morris, and Fitz-Gibbon (1987, 146) suggest that in order that acceptable results be obtained from an instrument, it is necessary to demonstrate construct, content, or concurrent validity.

Construct validity is an appraisal of how well the instrument measures what it is said to measure (Henerson, Morris, and Fitz-Gibbon 1987, 146). Carmines and Zeller (1979, 23) maintain that the first step in construct validation is establishing a theoretical framework from which the constructs were conceptualised. This first step of validation has been previously outlined. The theoretical validation of the construct is then supported by investigation of the hypothesis and replication of these and similar studies (Carmines and Zeller 1979, 24; Henerson, Morris, and Fitz-Gibbon 1987, 146). While the theoretical validation of the constructs has been previously established, the hypothesis testing justification of the construct validation has yet to be addressed.

The difficulty in establishing content validation of the attitude to(wards) programming scale is in first agreeing that liking programming, programming difficulty and programming usefulness are concepts relevant to the domain content of attitude towards programming (Carmines and Zeller 1979,21). The literature does suggest that these constructs have been used in attitude towards computers scales but their exact definitions have not been described. Secondly, content validity is an assessment of how the items of the attitude scale appropriately reflect the components of the constructs (Carmines and Zeller 1979, 22; Henerson, Morris, and Fitz-Gibbon 1987, 146). This aspect of content validation is essentially an appeal to an appraisal of the adequacy of the match of the items to their construct.

Henerson, Morris, and Fitz-Gibbon (1987, 146) distinguish between concurrent validity and predictive validity in that concurrent validity is supported when the attitude measure is highly correlated with the results of another associated measure. Predictive validity is established when the results of the instrument correlates with another valid measure of the behaviour that the instrument aims to measure. Nunnally (1967, 77) contends that the distinction between predictive and concurrent validity does not imply different validation procedures

because in each case the attitude instrument is correlated with a criterion measure. It does not matter when the data is obtained.

The criterion measure in this case is achievement on the programming end-of-unit test held just prior to the administration of the fifty-three item attitude towards programming pool. The final attitude instrument, and in particular the programming difficulty items, would be expected to correlate with achievement. The Pearson product-moment correlation is, unfortunately, unable to be used because the respondents to the attitude to(wards) programming scale remain anonymous and thus it is not possible to match the attitude scores with the achievement scores. One possible indication of concurrent validity is to test the 'goodness of fit' between the attitude score distribution and the achievement score distribution (Hays and Winkler 1975, 822). The null hypothesis that there is no difference in the two distributions is not rejected. This does provide some concurrent validity to the attitude towards programming scale.

Henerson, Morris, and Fitz-Gibbon (1987, 135) propose that the validity of an instrument is established by anticipating arguments that may be used to question the results of the instrument.

Rebuttal of arguments that may be used to question results and threaten validity (Henerson, Morris, and Fitz-Gibbon 1987, 145) are summarised by :

> response bias. The strong emphasis and demonstration that anonymity will be maintained has hopefully allowed respondents to express their true feelings and beliefs;
>
> lack of comprehension. The effort in interviewing potential respondents has hopefully produced item statements which respondents understand and recognise;

administration. All respondents were treated in the same manner and under the same circumstances.

too few items. The final instrument of twenty-four items resulting from a purification of an initial fifty-three items has hopefully been demonstrated to be sufficient.

The derivation of the final instrument from theory-based constructs and item analysis leads to the hypothesis that there are three factors involved. From an hypothesis perspective, Nunnally (1967, 342) suggests the use of the multiple-group method, otherwise known as the group-centroid method, of testing for the presence of the hypothesised three factors. The factor analysis produced the factor loadings shown in table 6.

	Factor A	**Factor B**	**Factor C**
Item 1	0.88	0.67	0.61
Item 2	0.84	0.55	0.58
Item 3	0.79	0.35	0.75
Item 4	0.80	0.50	0.57
Item 5	0.82	0.48	0.58
Item 6	0.74	0.38	0.39
Item 7	0.78	0.35	0.50
Item 8	0.86	0.74	0.53
Item 9	0.86	0.64	0.68
Item 10	0.86	0.57	0.51
Item 11	0.43	0.78	0.14
Item 12	0.65	0.84	0.37
Item 13	0.33	0.65	0.22
Item 14	0.40	0.79	0.19
Item 15	0.50	0.71	0.52
Item 16	0.48	0.65	0.46
Item 17	0.55	0.84	0.31
Item 18	0.42	0.35	0.68
Item 19	0.70	0.41	0.78
Item 20	0.43	0.20	0.67
Item 21	0.40	0.28	0.68
Item 22	0.44	0.20	0.70
Item 23	0.45	0.30	0.71
Item 24	0.64	0.38	0.81

Table 6. Factor loadings on the hypothesised three attitude factors. Multiple-group method (Nunnally 1967,344).

Programming achievement test

This study compares the attitudes and achievement in procedural programming and object-oriented programming in relation to cognitive skills of Year twelve students. The programming environment involves the attaching of sophisticated user interfaces to students' applications.

Explicit instruction, within this study, involves the provision of templates and the modelling of solution strategies. Language templates are briefly defined as programming plans (Soloway and Ehrlich 1984). This approach, as opposed to unguided discovery classes, is supported by a number of sources (Doyle 1983; Eylon and Helfman 1985; Dalbey and Linn 1985; Linn, Sloane and Clancy 1987).

The two groups of students are presented with a full solution to 'school athletics records' incorporating the use of a procedural or object-oriented toolkit to produce an acceptable interface. Students within each group receive instruction, observe, analyse and modify the solution. Modification of 'school athletics records' to develop a solution to 'overdue library books' is explicitly demonstrated.

The achievement tests for both groups, procedural and object-oriented, involves the development of a solution to a 'property rentals' application and is included in Appendix D. The format is a one and a half hour pencil and paper test. Students are permitted the use a full printed solution to the 'school athletics records' application together with a summary of the appropriate toolkit library functions.

The achievement test assesses three programming component skills contained within Shneiderman's syntactic and semantic model of programming competence (Shneiderman, 1980, cited in Foreman 1988, 6). Knowledge of syntax, the ability to design and generate code (program composition) and the ability to

restructure a program by adding and changing code (program modification) are inherent within the achievement test (Foreman 1988, 8).

The following scoring system was adopted:

Each *syntax* error was circled and the number of errors totalled, providing a score ranging from ten (no syntax errors) to zero (10 or more syntax errors).

The student's solution required *modification* to four major components. The score ranging from a maximum of twenty (all modifications correctly made) to a minimum of zero. The method of scoring for each major component is indicated below.

MenuChoice: Seven modifications were required. A maximum score of five was given with one subtracted for each incorrect modification and a minimum of zero recorded.

CreateFile: Twenty-one modifications were required. A maximum score of five was given with one subtracted for each incorrect modification and a minimum of zero recorded.

AlterRecord: Eleven modifications were required. A maximum score of five was given with one subtracted for each incorrect modification and a minimum of zero recorded.

ShowRecords: Nine modifications were required. A maximum score of five was given with one subtracted for each incorrect modification and a minimum of zero recorded.

The student's solution required the *composition* of a 'ShowVacantProperty' component. Each logic error was crossed, providing a score ranging from ten (no logic errors) to zero (more than ten logical errors).

Ethical considerations

The intention to manipulate and control the treatment of human participants in this study necessitates some attention to ethical concerns. Tuckman (1978, 15), Gay (1981, 63), Wiersma (1991, 297) and Maxwell (1992, 9) outline a number of ethical concerns involved in the process of research, and the following statements are responses to these considerations.

Initial permission to conduct the study within the school and to use students as subjects was granted by the Principal. The Principal has been informed of all activities throughout the study, to the extent that copies of the research proposal, lesson plans, student activities, measuring instruments, data analysis and the thesis have been furnished.

A counterbalanced instructional design, in which each group received all treatments, but in a different order, was deliberately selected so that no student was disadvantaged or felt that some form of instruction had been missed.

No data was collected without students' permission. All students were informed of the general purpose of the study, the instruments to be used, and were provided with feedback upon request. Students' individual data was treated as strictly confidential and was protected in the following manner:

> Student responses to the Learning Style Profile and Attitude Towards Programming instruments were identified by a code, with an elected student holding access to the code/student identification;
>
> Data was presented in group statistics. Access to individual data will be provided only to the supervisor of this study and to the Principal of the school. The thesis contains only group statistics .

Assumptions and limitations

Major assumptions of this study are that the constructs measured by the instruments provide sufficient basis to answer the research questions, and that the students are honest and conscientious in completing the instruments.

Delimitations are that the study is limited to the students enrolled in Year 12 Information Processing and Technology at Stanthorpe State High School in 1995.

Limitations of the study are that all instruments are self-report pencil and paper measures; the Information Processing and Technology (IPT) students are not a representative sample of students from the school's population in that IPT students chose to study IPT from a selection of other subjects; and the collection of learning style information is restricted to the use of the Learning Style Profile.

CHAPTER 4

Analysis of data

Introduction

The research question essentially involves a comparison of procedural programming achievement and object-oriented programming achievement. The comparison is made on the basis of variables selected within a framework of school learning as proposed by Bloom (1976).

Independent variables

Instructional treatment: a nominal variable based on instruction in the creation of a graphical user interface within either

a) a procedural programming environment, or

b) an object oriented programming environment.

Cognitive skills: a composite variable measured by the Learning Style Profile and transformed by factor analysis to a two factor interval scale. Subsequent analysis requiring conversion to a two-point ordinal variable.

Dependent variables

Achievement, measured on an arbitrary interval scale, in three areas of programming competence:

knowledge of syntax

program modification

program composition

Attitude to(wards) programming, measured on an arbitrary interval scale by pretest/posttest, in three areas:

liking programming

programming difficulty

programming usefulness

Overview of analyses

The analyses essentially involve:

A test of difference of achievement, in the three areas of programming competence, in each instructional treatment.

A correlational analysis of learning style factors and programming achievement subscores within each of the treatment environments.

Three ANOVAs with each of the three achievement scores as dependent measure and with the treatment groups (two level) and factor scores (each of the two factors in two levels) as independent measures. Lack of correlation across the three achievement scores formed the justification for not using a MANOVA.

An ANCOVA with pretest attitude to(wards) programming as covariate, posttest attitude to(wards) programming as dependent variable, treatment groups (two level) and factor scores (each factor in two levels) as independent variables.

Descriptive statistics

Learning style instrument

Assumptions of normality, required for subsequent statistical analysis, of the learning style profile subscales are generally supported by stem-and-leaf plots

and measures of kurtosis and skewness shown in Table 7. The analytical skill data, simultaneous processing skill data, and sequential processing skill data casts considerable doubt about assumptions of normality.

Analytical skill

Procedural		Object-oriented
0	0	0
1	0	
2	0	2
33	0	
4444444	0	4444444444444
555555555555	0	555555555

4.04	Mean	4.13
1.37	Std Dev	1.12
2.75	Kurtosis	8.06
-1.75	Skewness	-2.52

Memory skill

Procedural		Object-oriented
	0	3
55	0	4445
7776666	0	666777777
99999888888	0	8888999
100	1	0
2	1	22

7.96	Mean	7.21
1.81	Std Dev	2.32
-0.20	Kurtosis	0.08
0.26	Skewness	0.27

Categorical skill

Procedural		Object-oriented
322	0	1
65444	0	666
999877	0	7778888899
2110	1	0112
54433	1	344
7	1	89
2	1	4

8.71	Mean	10.17
4.38	Std Dev	4.96
-1.04	Kurtosis	1.74
0.09	Skewness	1.10

Discrimination skill

Procedural		Object-oriented
0	0	0
11	0	
2222	0	222
333333	0	3333333
4444444	0	44444444
5555	0	55555

3.17	Mean	3.50
1.37	Std Dev	1.22
-0.24	Kurtosis	1.51
-0.55	Skewness	-0.95

Sequential processing skill

Procedural		Object-oriented
	0	3
	4	0
5555	0	
6666666666666666	0	66666666666666666666

5.75	Mean	5.88
0.53	Std Dev	0.61
4.14	Kurtosis	24.00
-2.13	Skewness	-4.90

Spatial skill

Procedural		Object-oriented
22	0	2
333	0	
4444444444	0	44444444444444
555555555	0	5555555

4.08	Mean	4.13
0.93	Std Dev	0.74
0.22	Kurtosis	1.76
-0.89	Skewness	-0.91

Simultaneous processing skill

Procedural		Object-oriented
	2	0
	3	0 3
	4	0 4444
555555555555555555555	0	5555555555555555555

4.75	Mean	4.75
0.74	Std Dev	0.53
9.37	Kurtosis	4.14
-3.11	Skewness	-2.13

Persistence

Procedural		Object-oriented
78	0	699
221100	1	0012
5555544443	1	33444444445
986666	1	66667
	2	0

13.58	Mean	13.38
2.96	Std Dev	3.05
-0.00	Kurtosis	0.68
-0.51	Skewness	-0.41

Table 7. Learning style profile: Stem-and-leaf displays, kurtosis, and skewness of responses of the forty-eight Year 12 IPT students to each of the eight cognitive skill variables.

A comparison of sample size, means and standard deviations of the Year twelve IPT students with American Year twelve normative data is shown in Table 8. The Year twelve normative data is provided in the learning style profile technical manual (Keefe and Monk 1988). The reliability indices are essentially of the same magnitude with the exception of spatial skill.

	Year 12 IPT				US Year 12			
Subscale	n	mean	sd	alpha	n	mean	sd	alpha
1 Analytical skill	48	4.08	1.23	.65	893	2.97	1.52	.56
2 Spatial skill	48	4.10	.83	.12	899	3.04	1.45	.60
3 Discrimination skill	48	3.33	1.29	.46	902	2.96	1.50	.51
4 Categorization skill	48	9.44	4.69	.72	902	2.96	1.50	.74
5 Sequential processing skill	48	5.81	.57	.53	902	5.27	1.20	.72
6 Memory skill	48	7.58	2.09	.45	874	6.13	2.67	.62
7 Simultaneous Processing skill	48	4.75	.64	.52				
11 Persistence orientation	48	13.48	2.97	.79	889	12.79	2.68	.67

Table 8. Learning style profile: subscale means, standard deviations and Cronbach's alphas of Year twelve IPT students and Year twelve American students.

There are significant differences in the means of the two populations (t-statistic, 0.05 significance) as illustrated in Table 9. The explanation is that the Year 12 IPT student group is an above-average, self-selected group of students. This above average grouping is supported by the QCS data of Table 5 (page 42). Violations of the assumption of normality have only trivial effects on the level of significance and power of the t-test (Glass and Stanley 1970, 297).

Subscale	**t-statistic**	**p**
1 Analytical skill	+6.01	**0.00**
2 Spatial skill	+8.21	**0.00**
3 Discrimination skill	+1.92	0.06
4 Categorization skill	+9.55	**0.00**
5 Sequential processing skill	+5.90	**0.00**
6 Memory skill	+4.61	**0.00**
7 Simultaneous Processing skill		
11 Persistence orientation	+1.58	0.11

Table 9. Differences in the learning style profile means of Year 12 IPT students and Year 12 American normative data.

The NASSP considered eight cognitive skill variables necesary to build a picture of the demands of learning. The wealth of this information, together with the somewhat vague nature of the collected data, clouds an ability to understand the cognitive demands of each treatment. Researchers such as Fowler (1980), Brumby (1982), and Riding and Buckle (1990) have suggested that a number of types of cognitive skills are actually different conceptualisations of each other. It therefore appeared appropriate to undertake exploratory factor analysis in order to reduce the number of cognitive skill variables to a more manageable number (Amick and Walberg 1975, 115). The two factor model exhibited promise and after deleting items which showed low correlation to both factors, the two factor solution of Table 10 was accepted. The two factors accounted for 33 percent of the total variance.

Cognitive Skill		Factor A	Factor B	h^2
Analytical	Item 25	0.62		0.39
	Item 26	0.39		0.17
	Item 27	0.81		0.67
	Item 28	0.58		0.37
	Item 29	0.50		0.29
Sequential Processing	Item 1	0.39		0.21
	Item 2	0.37		0.15
	Item 6	0.74		0.57
Discrimination	Item 9	0.59		0.40
Spatial	Item 39	0.42		0.17
	Item 40	0.55	0.45	0.51
Categorisation	Item 18	0.46		0.22
Simultaneous Processing	Item 12		0.70	0.49
	Item 13		0.72	0.52
Persistence	Item 74		0.65	0.43
	Item 84		0.41	0.17
	Item 91	0.30	0.41	0.26
Memory	Item 112		0.50	0.25
	Item 114		0.38	0.15
	Item 116		0.35	0.14

Table 10. Varimax factor loadings of the eight NASSP cognitive skill variables on two factors. Loadings less than 0.30 have been extracted. Communalities (h^2) are also indicated.

A factor score for each of the forty-eight students was arrived at by multiplying each item score by each factor score coefficient and then summing the products (Hedderson 1991, 179). The distribution of student factor scores is illustrated in Table 11.

Factor A			**Factor B**		
Procedural		Object-oriented	Procedural		Object-oriented
00	0			0	1
	1	1	74	1	
	2	9		2	5
5	3	69	8	3	6
987431	4	5789	751	4	114457
876541	5	0111223666	88776643311	5	001222344579
21111110	6	0011122	4322221	6	129
47.50	Mean	52.50	52.15	Mean	47.85
54.98	Median	52.94	56.56	Median	52.02
19.33	Std Dev	8.55	13.21	Std Dev	16.60
2.68	Kurtosis	1.15	3.57	Kurtosis	10.25
-1.86	Skewness	-1.08	-1.92	Skewness	-2.77
0.23 (p=0.00)	K-S	0.12 (p>0.2)	0.18 (p=0.04)	K-S	0.19 (p=0.03)

Table 11. Two factor cognitive skill profile: Stem-and-leaf displays, kurtosis, and skewness of scores of the forty-eight Year 12 IPT students. Standardised to a mean of 50 and a standard deviation of 15.

The cognitive skill factor distributions demonstrate kurtosis and skewness indices which suggest non-normal distributions. The computed values of the Kolmogorov-Smirnov (K-S) test statistic which assesses normality (Jobson 1991, 61), are significant at the 0.05 level in most instances supporting concern about non-normality of the data. The K-S statistics are shown in Table 11.

Jobson (1991, 56) suggests that transformations should be applied to small samples distributions which have outlier and skewness/kurtosis problems. The purpose of the transformation being to transform the scale of measurement using a nonlinear transformation to obtain a distribution shape that is more normal-like. The transformed data, using a Box-Cox λ power transformation ($\lambda = 4$) to remove skewness, is shown in Table 12 (Jobson 1991, 68).

Factor A			Factor B		
Procedural		Object-oriented	Procedural		Object-oriented
544	2	9	76	2	59
9973	3	35	85	3	478
8432	4	02356777999	8720	4	0013668889
85431	5	445	776644100	5	012359
43333222	6	0223355	9644442	6	34
49.25	Mean	50.75	52.16	Mean	47.84
52.47	Median	49.52	54.69	Median	48.40
13.64	Std Dev	10.35	11.96	Std Dev	11.89
-0.93	Kurtosis	-0.68	-0.23	Kurtosis	1.49
-0.55	Skewness	-0.24	-0.68	Skewness	0.65
0.13 (p>0.2)	K-S	0.12 (p>0.2)	0.08 (p>0.2)	K-S	0.12 (p>0.2)

Table 12. Two factor cognitive skill profile: skewness removed by a Box-Cox transformation and standardised to a mean of 50 and standard deviation of 15.

Attitude to(wards) programming scale

Descriptive statistics of the pretests and posttests of the attitude to(wards) programming subscales are listed in Table 13. Assumptions of normality of the attitude to(wards) programming data, an assumption required for following statistical analysis, are illustrated to some extent by the shape of the stem-and-leaf plots and supported by the Kolmogorov-Smirnov (K-S) test statistic for assessing normality (Jobson 1991, 61).

Pretest	Posttest
Liking programming	
Procedural Object-oriented 1 3 98766544443 2 24568899 988443310 3 00134456 5420 4 005566 5 0	Procedural Object-oriented 8 1 3 988750 2 1235699 9887644433210 3 00024678888 2000 4 0168 5 0
31.71 Mean 33.29 30.50 Median 32.00 6.98 Std Dev 8.96 -1.03 Kurtosis -0.14 0.48 Skewness 0.03 0.13 (p>0.2) K-S 0.10 (p>0.2)	32.75 Mean 33.08 33.50 Median 33.00 6.31 Std Dev 8.99 0.10 Kurtosis -0.11 -0.69 Skewness -0.11 0.08 (p>0.2) K-S 0.09 (p>0.2)
Programming difficulty	
Procedural Object-oriented 0 7 4433331 1 344 999865 1 55556789 43322110 2 000124 755 2 678 3 013	Procedural Object-oriented 0 7 4 1 4 99887666 1 55667789 422111000 2 01122344 77755 2 66 0 3 0123
18.75 Mean 20.00 19.00 Median 19.50 4.67 Std Dev 6.53 -1.23 Kurtosis -0.40 -0.03 Skewness 0.36 0.14 (p>0.2) K-S 0.13 (p>0.2)	21.04 Mean 21.21 20.50 Median 21.00 4.18 Std Dev 6.37 -0.52 Kurtosis -0.07 0.41 Skewness 0.11 0.13 (p>0.2) K-S 0.08 (p>0.2)
Programming usefulness	
Procedural Object-oriented 1 1 2 97 1 689 44222 2 1123 98777776665 2 55667889999 22100 3 02223	Procedural Object-oriented 1 2 21 2 22234 99888888777666665555 2 555677888899 10 3 001333
25.46 Mean 25.50 26.50 Median 26.50 4.90 Std Dev 5.51 2.12 Kurtosis 0.06 -1.22 Skewness -0.77 0.09 (p>0.2) K-S 0.09 (p>0.2)	26.71 Mean 26.67 27.00 Median 27.50 2.27 Std Dev 4.61 1.03 Kurtosis 3.23 -0.65 Skewness -1.25 0.12 (p>0.2) K-S 0.08 (p>0.2)

Table 13. Descriptive statistics, including stem-and-leaf display, of measurements obtained using the attitude to(wards) programming instrument (forty-eight responses).

The reliability, alpha correlation coefficients, of the attitude to(wards) programming scale is listed in Table 14. The coefficients suggest that the instrument yielded consistent results and that each administration of the instrument yielded essentially the same results.

Subscale	Pilot	Pretest	Posttest
Liking programming	0.95 (46)	0.93 (48)	0.93 (48)
Programming difficulty	0.91 (46)	0.89 (48)	0.87 (48)
Programming usefulness	0.88 (46)	0.88 (48)	0.82 (48)

Table 14. Reliability indices (Cronbach's alpha) and student numbers for the attitude to(wards) programming subscales.

Programming achievement instrument

Descriptive statistics of the programming achievement test are shown in Table 15. Data for program modification, a measure of the ability to change code and add to code, supports assumptions of normality as does program composition data, a measure of the ability to design and generate code. The knowledge of syntax measurement data demonstrates a distribution which is not normal. Analytical conclusions about student syntax knowledge will need to be approached with caution.

Knowledge of syntax (S)

Procedural		Object-oriented
1	0	
5	0	
66	0	6
7	0	777
8888	0	8
99	0	99
0000000000000	1	00000000000000000
8.54	Mean	9.29
2.23	Std Dev	1.27
4.71	Kurtosis	1.10
-2.02	Skewness	-1.58
0.26 (p=0.00)	K-S	0.29 (p=0.00)

Program modification (M)

Procedural		Object-oriented
0	0	
55	0	4
97	0	77999
21	1	222
55543	1	334444
87777766666	1	667788888
9	1	
13.46	Mean	13.29
4.88	Std Dev	4.07
1.30	Kurtosis	-0.44
-1.41	Skewness	-0.62
0.17 (p=0.07)	K-S	0.12 (p=0.20)

Program composition (C)

Procedural		Object-oriented
000	0	0
322	0	3
55	0	4455
7776666	0	666677
998888888	0	888889999
	1	000
5.67	Mean	6.88
2.93	Std Dev	2.49
-0.39	Kurtosis	0.97
-0.93	Skewness	-0.95
0.13 (p>0.20)	K-S	0.10 (p>0.20)

Table 15. Descriptive statistics, including stem-and-leaf display, of measurements obtained using the programming achievement instrument (forty-eight responses).

Analytical procedures

Statistical hypothesis 1

Hypothesis: Student achievement in adding a sophisticated user interface within a familiar procedural programming environment will be significantly greater than student achievement in adding a sophisticated user interface within a relatively unfamiliar object-oriented environment.

Null A: The mean achievement of knowledge of syntax within a procedural programming environment, μ_1, is equal to the mean achievement of knowledge of syntax within an object-oriented programming environment, μ_2. H_o: $\mu_1 = \mu_2$.

Null B: The mean achievement of program modification within a procedural programming environment, μ_1, is equal to the mean achievement of of program modification within an object-oriented programming environment, μ_2. H_o: $\mu_1 = \mu_2$.

Null C: The mean achievement of program composition within a procedural programming environment, μ_1, is equal to the mean achievement of program composition within an object-oriented programming environment, μ_2. H_o: $\mu_1 = \mu_2$.

Descriptive statistics of the programming achievement test are shown in Table 15.

The Student's t statistic for independent samples provides an appropriate basis upon which to accept or reject the null hypotheses ($t_{critical,0.05,46df,one\text{-}tail} = 1.68$). Glass and Stanley (1970, 297) claim that violations of the assumption of normality, as suggested by the knowledge of syntax data, have only trivial

effects on the level of significance and power of the t-test. Because n_1 and n_2 are equal within this study, violations of the assumption of homogeneous variances are unimportant (Glass and Stanley 1970, 297).

The null hypotheses of no difference of achievement means is accepted for each area of programming competence. The t statistics and p-values are shown in Table 16.

Knowledge of syntax	**Program modification**	**Program composition**
$t_{critical} = 1.68$ $t_{calculated} = -1.43$	$t_{critical} = 1.68$ $t_{calculated} = 0.13$	$t_{critical} = 1.68$ $t_{calculated} = 1.54$

Table 16. Student's t statistics comparing achievement means of each instructional treatment ($t_{0.05,46df,one\text{-}tail}$)

Statistical hypothesis 2

Hypothesis There is a significant interaction between instructional treatment and learning style factors with respect to achievement in each of the areas of programming competence.

Null A: There is no relationship between programming achievement within a procedural programming environment, as measured by achievement scores, and each of the cognitive skill factors.
Correlational analysis, $t_{0.05,n-2,upper\text{-}tail}$. H_o: $\rho = 0$.

Null B: There is no relationship between programming achievement within an object-oriented programming environment, as measured by achievement scores, and each of the cognitive skill factors.
Correlational analysis, $t_{0.05,n-2,upper\text{-}tail}$. H_o: $\rho = 0$.

Null C: The differences in programming achievement, measured by each of the achievement scores, of students with varying cognitive skills (low, high) are independent of the programming environment (procedural programming, object-oriented programming). ANOVA 2x2 (α = 0.05).

Correlational analysis was chosen to test *null hypothesis A* and *null hypothesis B* based on the need to initially assess the *degree* to which the two variables, programming achievement and cognitive skill, are related rather than *how* the two variables are linearly related.

The Pearson correlation coefficients, together with probabilities, are listed in Table 17. Within a procedural programming environment the null hypothesis is rejected for the relationship between syntax achievement and factor B, and for the relationship between composition achievement and factor A. Significant relationships exist between each of the three achievement scores and factor B within an object-oriented programming environment.

Cognitive Skills	Syntax	Modification	Composition
Procedural programming			
Factor A	-0.00 (p = 0.49)	0.32 (p = 0.06)	0.41 (p = **0.02**)
Factor B	0.36 (p = **0.04**)	0.30 (p = 0.07)	0.05 (p = 0.40)
Object-oriented programming			
Factor A	0.04 (p = 0.43)	-0.04 (p = 0.42)	0.01 (p = 0.48)
Factor B	0.38 (p = **0.04**)	0.63 (p = **0.00**)	0.66 (p = **0.00**)

Table 17. Pearson coefficients of correlation measuring the strength of the linear relationship between each of the cognitive skill factors and student programming achievement within each of the programming environments.

It would be useful to follow the correlation analysis with a regression analysis to determine the extent to which the previously established significant relationships contribute to explanations of variance in student achievement. The coefficient of

multiple determination, R^2, is the ratio of the explained variance to the total variance and gives an indication of the contribution of cognitive factors to student programming achievement. R^2 and the F-statistic, measuring overall goodness of fit of the regression, are listed in Table 18.

The regression analysis indicates that cognitive factor B significantly contributes to program modification achievement and program composition achievement within an object-oriented programming environment. Cognitive factor A explains seventeen percent of the variation in program composition achievement within a procedural programming environment.

Cognitive Factors	Syntax	Modification	Composition
Procedural			
Factor A			$R^2 = 0.17$ $F = 4.50$ ($p = \mathbf{0.05}$)
Factor B	$R^2 = 0.13$ $F = 3.36$ ($p = 0.08$)		
Object-oriented			
Factor A			
Factor B	$R^2 = 0.14$ $F = 3.60$ ($p = 0.07$)	$R^2 = 0.39$ $F = 14.29$ ($p = \mathbf{0.00}$)	$R^2 = 0.44$ $F = 17.18$ ($p = \mathbf{0.00}$)

Table 18. Linear regression analysis quantifying the contribution of cognitive skill factors to programming achievement.

The model used to test *null hypothesis C* was a two-way analysis of variance. This model is appropriate because the hypothesis wishes to investigate the relationship between the dependent achievement variables (knowledge of syntax, program modification, and program composition), and two independent qualitative variables. The qualitative variables are cognitive skill factors (low, high) and programming environment (procedural programming, object-oriented programming). Figure 7 is a plot of the group means.

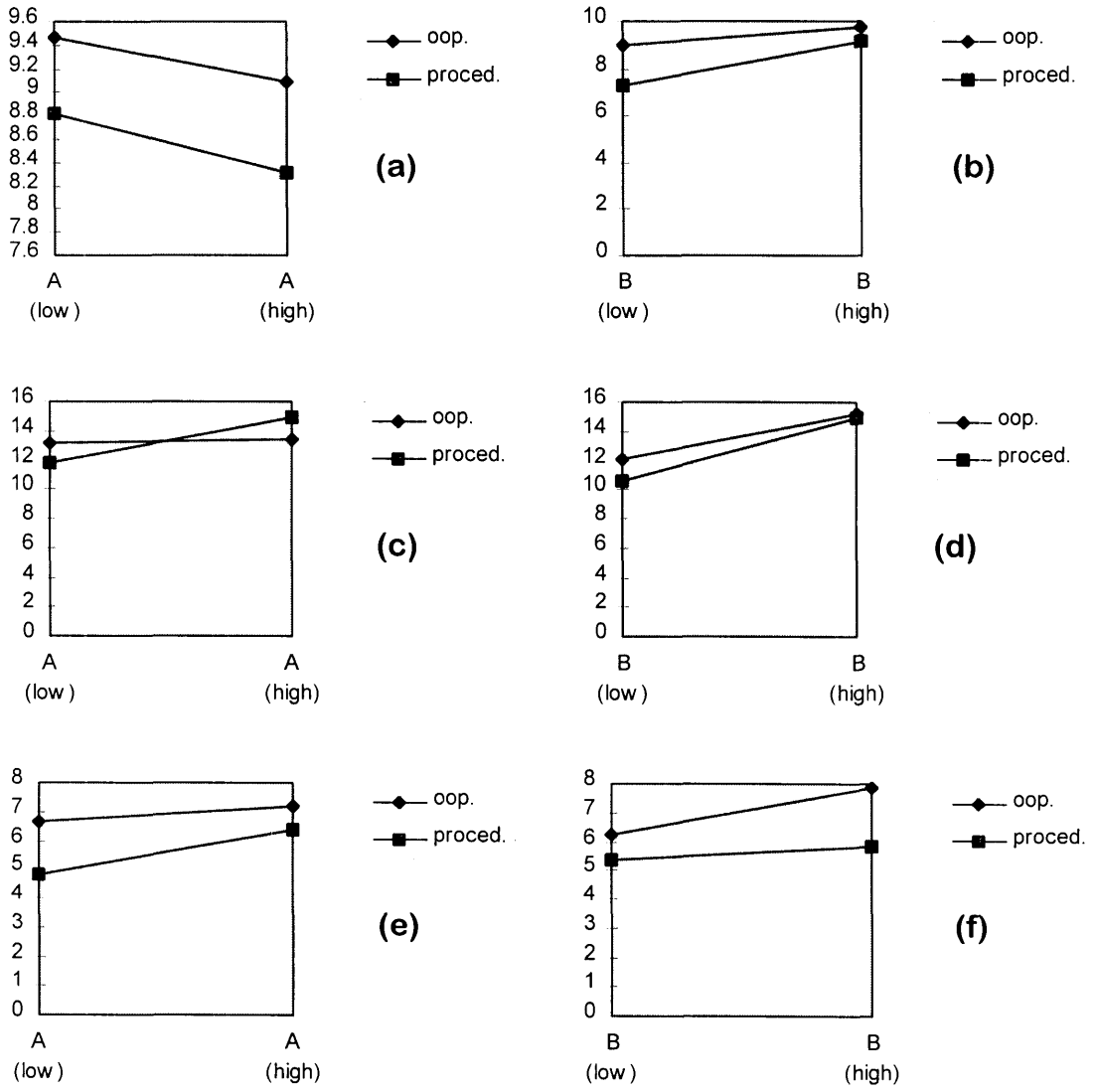

Figure 7. **Plots of mean programming achievement (syntax (a) and (b), program modification (c) and (d), and program composition (e) and (f)) in cognitive factors A and B (low , high).**

The ANOVA analysis, summarised in Table 19, indicates that the null hypothesis of no interaction is to be accepted in all cases. Student achievement, with relation to cognitive skills, is independent of the programming environment. In the absence of interaction it is interesting to note that the main effects of programming environment is negligible and that the main effects of the individual cognitive skills are similar in significance to the correlational analysis of hypothesis 1.

Source	F	p-value
Knowledge of syntax		
Cognitive skill Factor A	0.69	0.41
Programming environment	1.80	0.19
Interaction	0.02	0.90
Cognitive skill Factor B	6.97	**0.01**
Programming environment	5.17	**0.03**
Interaction	1.27	0.27
Program modification		
Cognitive skill Factor A	1.85	0.18
Programming environment	0.00	0.99
Interaction	1.27	0.27
Cognitive skill Factor B	8.33	**0.01**
Programming environment	0.53	0.47
Interaction	0.21	0.65
Program composition		
Cognitive skill Factor A	1.84	0.18
Programming environment	2.73	0.11
Interaction	0.41	0.53
Cognitive skill Factor B	1.58	0.22
Programming environment	3.27	0.08
Interaction	0.52	0.47

Table 19. Two-way ANOVA of student programming achievement. One factor being cognitive skill (factor A, factor B) and the other factor being programming environment (procedural , object-oriented).

Statistical hypothesis 3

Hypothesis There is no significant overall relationship between attitude changes on any of the three scales and the instructional treatment.

Null: The difference between the mean pretest attitude scores and the mean posttest attitude scores are independent of the programming environment (procedural programming or object-oriented programming). ANOVA 2x2 (24 replications, $\alpha = 0.05$).

An analysis of variance was used to test for factor interaction. The two levels of instructional treatment (procedural programming and object-oriented programming) combined with two levels of attitude measures (pretest and posttest) established an analysis by means of a balanced two-way factorial model with twenty-four replications. Figure 8 is a plot of the group means,

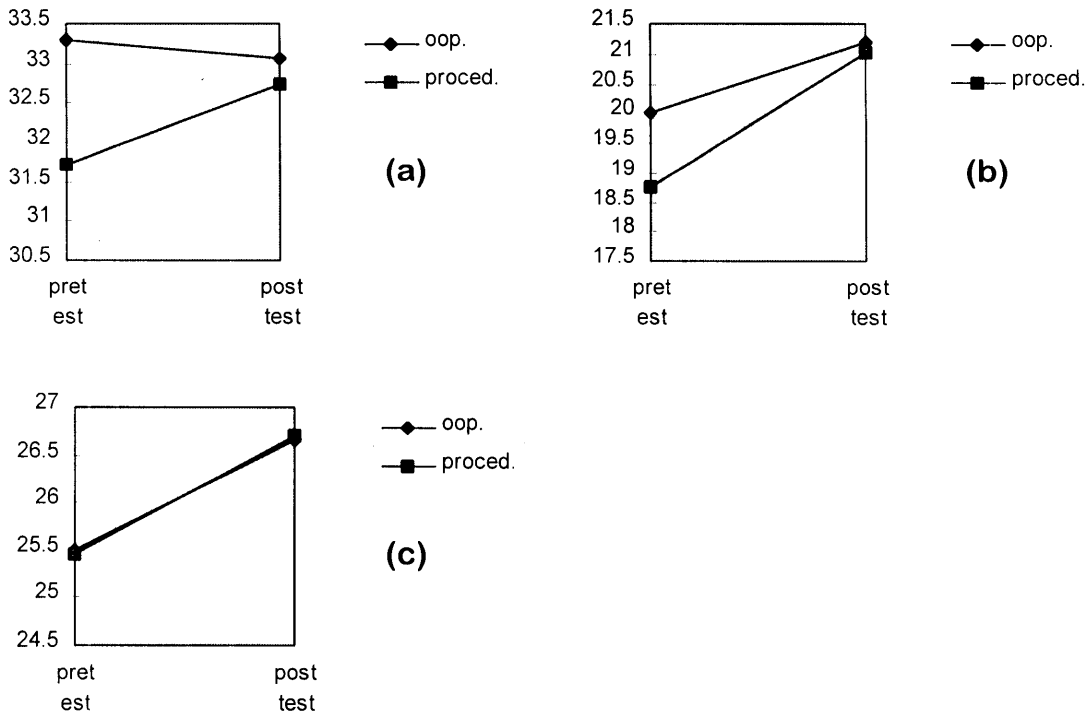

Figure 8. Plots of mean attitude (pretest, posttest) of programming environment in liking programming (a), programming difficulty (b), and programming usefulness (c).

The testing of the null hypothesis was carried out using the F-statistic ($F_{critical,0.05,1,92} = 3.92$). The null hypothesis of no interaction between attitude to(wards) programming and the programming environment is accepted ($F_{calculated} < F_{critical}$) for each of the three attitude scales. Table 20 summarises the analysis.

Source	df	SS	MS	F	p-value
Liking programming	1	4.17	4.17	0.07	0.80
Instructional treatment	1	22.04	22.04	0.35	0.55
Interaction	1	9.37	9.37	0.15	0.70
Error	92	5738.25	62.37		
Programming difficulty	1	73.50	73.50	2.40	0.13
Instructional treatment	1	12.04	12.04	0.39	0.53
Interaction	1	7.04	7.04	0.23	0.63
Error	92	2815.42	30.60		
Programming usefulness	1	35.04	35.04	1.73	0.19
Instructional treatment	1	0.00	0.00	0.00	1.00
Interaction	1	0.04	0.04	0.00	0.96
Error	92	1858.25	20.20		

Table 20. Two-way ANOVA of student attitude to(wards) programming. The factors programming liking, difficulty, and usefulness each at two levels (pretest, posttest) and the factor programming environment at two levels (procedural, object-oriented).

The absence of factor interaction leads to a consideration of main effects. From the values of the F-statistics in each case it may be concluded that there are no significant differences among the attitude means ($F_{critical,1,92} = 3.92$) nor among the programming environment means ($F_{critical,1,92} = 3.92$) at the 0.05 level.

Statistical hypothesis 4

Hypothesis There is a significant interaction between instructional treatment and cognitive skill factors with respect to attitude changes on any of the three scales.

Null: There is no interaction between cognitive skill factors (two factors each categorised as low, high) and instructional treatment (procedural programming and object-oriented programming) in their effects on attitude. ANOCOVA 2x2 ($\alpha = 0.05$).

One way to examine the question of whether the treatment affects attitude is to control for pretest attitudes by including the pretest attitudes as a covariate. An ANOCOVA, analysis of covariance, is necessary to analyse the effects of the treatment on student attitudes to(wards) programming because of the absence of an attitudes control group in the experimental design. An ANOCOVA, analysis of covariance, removes the effects of the predictor variable, attitude pretests, on the dependent variable, attitude posttests, through regression methods. An ANOVA is then performed on the corrected posttest attitude scores to analyse the effects of the instructional treatment on attitudes to(wards) programming.

A plot of the posttest attitude means, adjusted for the pretest attitude covariate is displayed in Figure 9 and the ANOCOVA is summarised in Table 21. There is no significant interaction. The main effects of instructional treatment and cognitive skill factors on attitude changes are also insignificant.

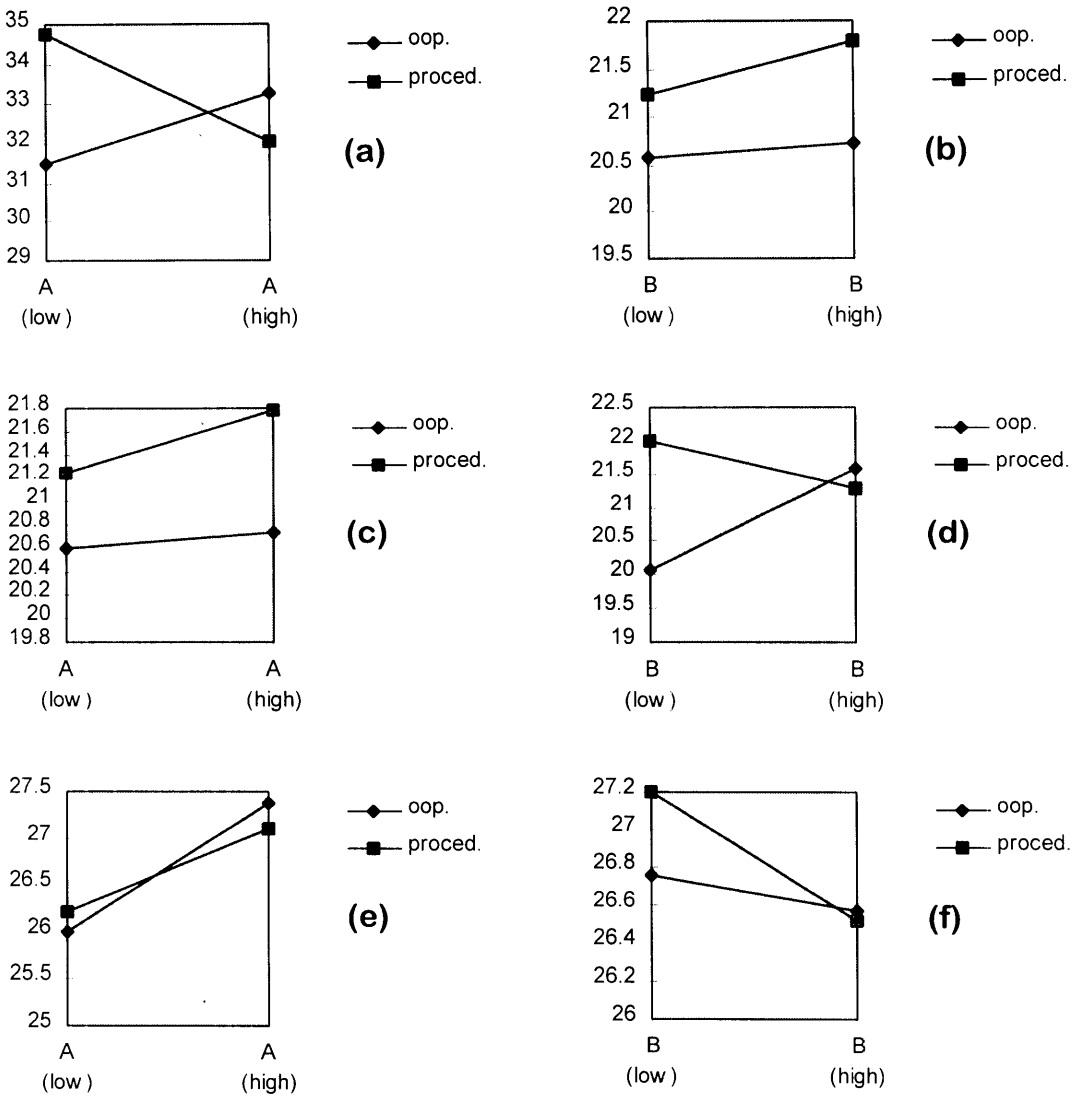

Figure 9. **Plots of mean posttest attitude, adjusted for the pretest attitude covariate, (liking programming (a) and (b), programming difficulty (c) and (d), and programming usefulness (e) and (f)) in cognitive skill factors A and B (low, high).**

Source	SS	DF	MS	F	Sig. Of F
Liking pogramming					
Within cells	853.36	43	19.84		
Regression	1873.52	1	1873.52	94.41	0.00
Constant	90.03	1	90.03	4.54	0.04
Instructional treatment	12.20	1	12.20	0.61	0.44
Cognitive Skill Factor A	2.48	1	2.48	0.13	0.73
Interaction	56.66	1	56.66	2.86	0.10
Within cells	838.22	43	19.49		
Regression	1792.55	1	1792.55	91.96	0.00
Constant	151.73	1	151.73	7.78	0.01
Instructional treatment	10.28	1	10.28	0.53	0.47
Cognitive Skill Factor B	0.23	1	0.23	0.01	0.91
Interaction	72.93	1	72.93	3.74	0.06
Programming difficulty					
Within cells	280.11	43	6.51		
Regression	988.07	1	988.07	151.68	0.00
Constant	85.95	1	85.95	13.19	0.00
Instructional treatment	8.43	1	8.43	1.29	0.26
Cognitive Skill Factor A	1.035	1	1.35	0.21	0.65
Interaction	0.46	1	0.46	0.07	0.79
Within cells	266.42	43	6.20		
Regression	1048.25	1	1048.25	169.19	0.00
Constant	89.21	1	89.21	14.40	0.00
Instructional treatment	7.25	1	7.25	1.17	0.29
Cognitive Skill Factor B	1.63	1	1.63	0.26	0.61
Interaction	13.58	1	13.58	2.19	0.15
Programming usefulness					
Within cells	382.67	43	8.90		
Regression	214.97	1	214.97	24.16	0.00
Constant	455.40	1	455.40	51.17	0.00
Instructional treatment	0.01	1	0.01	0.00	0.97
Cognitive Skill Factor A	15.10	1	15.10	1.70	0.20
Interaction	0.72	1	0.72	0.08	0.78
Within cells	395.75	43	9.20		
Regression	209.74	1	209.74	22.79	0.00
Constant	480.75	1	480.75	52.24	0.00
Instructional treatment	0.42	1	0.42	0.05	0.83
Cognitive Skill Factor A	2.06	1	2.06	0.22	0.64
Interaction	0.64	1	0.64	0.07	0.79

Table 21. Two-way ANOCOVA analysis of student posttest attitude to(wards) programming with pretest attitude as covariate. One factor being instructional treatment (procedural, object-oriented) and the other factor being cognitive skill (factor A, factor B).