N-S Chart

Honours Project Report

A. B. Couch 1988

A. Nassi-Shneiderman Cartographer

Supervisor: Dr. N. Churcher

Department of Computer Science
University of Canterbury
This report is one of three volumes that comprise the N-S Chart project report, and should be read in conjunction with the other two.

The N-S Chart Technical Report contains a critical evaluation of the programming environment used, and a listing of the program code for N-S Chart.

The N-S Chart User's Guide gives simple step-by-step instructions on how to use N-S Chart. Floppy disks containing the N-S Chart application, sample charts, and program code are also included in the back of the User's Guide.

Andrew Couch
# Table of Contents

1.0 Introduction ............................................................................. 5

2.0 Diagraming Methods ............................................................... 7
   2.1 The CAD / CASE Revolution ........................................ 7
   2.2 Better than a Thousand words ..................................... 7
   2.3 Changing Methods......................................................... 8

3.0 Computer Graphics Tools ....................................................... 9
   3.1 General Drawing Purpose Tools ................................. 9
   3.2 Tools for Using Dynamic Drawings ............................. 9
   3.3 Tools for Existing Methodologies ................................. 9
   3.4 Computer Enhanced Methodologies ............................ 10
   3.5 Existing Nassi-Shneiderman Diagrammers ............... 10

4.0 Criteria for good diagraming Methods ..................................... 12
   4.1 Attributes of good techniques .................................. 12
   4.2 Constructs to represent ............................................. 12

5.0 Nassi-Shneiderman Charts .................................................... 14
   5.1 Why use N-S Charts ? .............................................. 14
   5.2 What are N-S Charts ? ............................................. 15
   5.3 Nassi-Shneiderman symbols .................................... 15
      5.3.1 Sequence .................................................... 15
      5.3.2 Decision ..................................................... 16
      5.3.3 Selection .................................................... 16
      5.3.4 Iteration ..................................................... 16
      5.3.5 Block .......................................................... 17
      5.3.6 Parallel Execution ....................................... 17
   5.4 New Symbols ............................................................. 17
   5.5 Advantages of N-S Charts .......................................... 18

6.0 Why use a computer? ............................................................. 19
   6.1 The Benefits ............................................................. 19
      6.1.1 Seamlessness ............................................... 19
      6.1.2 Windows .................................................... 19
6.1.3 Consistent Style....................................... 19
6.1.4 Ease of Editing........................................ 20
6.1.5 Explosion / Implosion ............................ 20
6.1.6 Provision for Data ................................... 20
6.1.7 Potential for Interface.............................. 21
6.1.8 WIMP Environment ............................... 21
6.2 Disadvantages ............................................. 21

7.0 N-S Chart: The Application .............................. 23
7.1 Drawing Charts ............................................ 23
7.2 Editing Charts ............................................. 25
7.3 Navigating around Charts ............................ 25
7.4 File Operations ............................................. 26

8.0 User Interface Design .................................... 27
8.1 Point to Draw............................................. 27
8.2 Continual Chart Integrity ................................ 27
8.3 Multiple Charts .......................................... 28
8.4 Management of Complexity ............................ 28
8.5 Code Generation .......................................... 28
8.6 Structured Design ........................................ 28
8.7 Menus vs Tool Palette ................................... 29
8.8 Cursors ..................................................... 30
8.9 Text in symbols .......................................... 31

9.0 Implementation Details .................................. 32
9.1 Development Environment ................................ 32
  9.1.1 Lightspeed Pascal................................. 32
  9.1.2 Programmers Extender Libraries ............... 33
9.2 Basic Data Structures .................................. 33
  9.2.1 Chart Record ...................................... 33
  9.2.2 Symbol Record ................................... 36
9.3 Operations on the basic structures .......... 38
  9.3.1 Finding a Symbol .................................. 38
  9.3.2 Chart Walking ..................................... 40
  9.3.3 Drawing Symbols .................................. 40
  9.3.4 Text in Symbols .................................. 40
  9.3.5 Inputting Text ................................... 41
9.3.6 Tool Palette ............................................ 41
9.3.7 Case and Parallel Processing Symb ........... 42
9.3.8 Insert or Replace Mode............................ 42
9.3.9 Insert Before or After mode...................... 43
9.3.10 Selecting and Highlighting .................... 43
9.3.11 Cutting and Pasting............................... 44
9.3.12 Clipboard ............................................. 45
9.4 File Operations ........................................... 45
  9.4.1 File Format......................................... 45
  9.4.2 Printing Charts.................................... 45
9.5 Other Operations ......................................... 46
  9.5.1 On-line Help ....................................... 46

10.0 Possible Extensions ................................. 47
  10.1 Expand / Collapse .................................. 47
  10.2 Undo .................................................. 47
  10.3 Data Integrity ....................................... 47
  10.4 Diagrammer Integration ............................ 48
  10.5 Generating other Languages ..................... 48
  10.6 Database Orientation ............................... 48

11.0 Summary ................................................ 49

12.0 References ............................................. 50
1.0 Introduction

"Designers of computer hardware have had the benefit of computer aided design for many years. More recently the software industry is looking to computer applications to help in the design of software systems. This requires support for published methods for creating designs. The methods for "structured" software analysis and design often make use of diagrams as the notation for presenting, communicating and thinking about the design. There is therefore the need for design tools which support diagramming methods." [MACAD]

There are many reasons for making use of diagrams in the area of software engineering. Complex concepts and processes can be abstracted or simplified by describing them graphically rather than textually. Diagrams are clearer to read and enable the overall picture to be understood more rapidly. They can be used as a teaching aid and enhance learning speed.

Computer aided diagramming tools expand on these points and provide several additional advantages. Large diagrams can be worked with without having to shuffle great sheets of paper. The development becomes more flexible. Inevitable changes and modifications can be made to diagrams without needing to redraw the whole diagram by hand. Side effects of changes can be handled automatically by the computer. The integrity of a system may be maintained at a higher standard. The computer can enforce protocols and constraints that may otherwise be forgotten or overlooked despite the greatest of intent.

An alternative to using diagrams is to employ some other type of formal specification like denotational semantics. These notations only serve to add complexity to the task at hand for the average user.

This project involved the development of N-S Chart; a Nassi- Shneiderman Cartographer for the Apple Macintosh. N-S Chart is a tool for drawing and maintaining Nassi-Shneiderman diagrams, and incorporates the beneficial features of diagramming methods, and automated computer graphics tools discussed in chapters 2 and 3. The features of N-S Chart are discussed in chapter 7. Nassi-Shneiderman
charts (N-S Charts), an alternative to the flowchart, are described in chapter 5. A cartographer is someone who works with charts.

In the process of implementing the original Nassi-Shneiderman diagramming technique, modifications and improvements to the method were developed and implemented.

Flowcharts were designed in the days of 2nd generation languages. They do not support the constructs used in today's 3GLs and 4GLs. Nassi-Shneiderman charts do.

However the original Nassi-Shneiderman diagrams did have a few drawbacks which prevented them from becoming more widely used than they have been. The problems of data specification and drawability have been addressed in this project and successful solutions have been found. Chapter 8 describes aspects of user interface design. Symbols have been added to the drawing technique to support the specification of data inputs and outputs. Symbols have been modified to clarify charts and the drawing and maintenance process has been automated and simplified to an almost trivial level. Details of the implementation of N-S Chart are discussed in chapter 9 and possibilities for further extension are mentioned in chapter 10.
2.0 Diagramming Methods

2.1 The CAD / CASE Revolution

These days there is a new and very important reason for diagramming standards being well thought out. The job of the systems analysts and builders is undergoing revolutionary change. It is evolving from a pencil and paper activity to an activity of computer aided design [M&MC]. There is a general change from programmers writing code in third generation languages, to analysts and designers using tools that automatically produce code. We have CAI, CAD, CAM (computer aided instruction, computer aided design, and computer aided manufacturing). It is time for CASA and CAP (computer aided systems analysis and computer aided programming) or in other words CASE (Computer Aided Software Engineering).

Some diagramming techniques are more appropriate than others for computer automation. It is a great advantage if automation of diagramming leads to automatic generation of program code. Many diagramming techniques are not a sound basis for computerized design. They are too casual, unstructured, and cannot represent some of the necessary constructs. Many techniques perform well enough with small, model diagrams, but when used for designing large systems, can not handle the increase in complexity and size.

2.2 Better than a Thousand words

"If an appropriate diagramming technique is used, it is much easier to describe complex activities and procedures in diagrams than in text. A picture can be better than a thousand words because it is concise, precise, and clear." [M&MC].

One of the reasons why building and maintaining software systems is so expensive and error-prone is the difficulty that we have in clearly communicating our ideas to one another. Whether we are reading a functional specification, a program design, or a program listing, we often experience difficulty understanding what the author is telling us. Well-designed diagrams can greatly increase understanding.
2.3 Changing Methods

Diagramming techniques in computing are still evolving. Flowcharts have been shunned for a while now because they do not give a structured view of the program. Some of the early structured diagramming techniques need replacing because they fail to represent some important constructs.

The introduction of structured techniques into computing was a major step forward but the true structured revolution is that which makes the programmer unnecessary by the use of code generators working from structured designs.
3.0 Computer Graphics Tools

Computer graphics tools vary greatly in their capabilities and use to analysts and designers. They can be categorized as follows. [M&MC]

3.1 General Purpose Drawing Tools

These tools aid in the creation of static drawings. Various shapes can be drawn and perhaps manipulated on the screen. These tools range from programs like MacPaint, where the drawing created is nothing more than a bitmap, to tools like MacDraw, SuperPaint and Freehand, where once drawn, the shapes can be resized, shifted, grouped, combined and modified.

3.2 Tools for Using Dynamic Drawings.

"A static drawing is one without any built-in mechanisms. If one part of the drawing is changed or moved, it has no effect on any other part except perhaps to hide it or uncover it. A dynamic drawing is one with defined linkages between components or relationships between icons. When part of the drawing is changed or moved, other changes may occur automatically." [M&MC]

Relationships between various elements of the diagram may be defined and preserved. If the designer attempts to modify the diagram in a way that violates a specified constraint, she will be automatically warned, and perhaps not allowed to make the change.

The designer is able to build and manipulate diagrams on the screen faster than with static drawings.

3.3 Tools for Existing Methodologies.

Several graphics tools have been written for implementing many of the diagramming techniques that have emerged over the last 20 years. They generally provide the templates and icons from which these diagrams can be easily created. They use tools like menus, fill-in-the-blank panels and dialogues to help create the diagrams and enforce integrity constraints.
3.4 Computer Enhanced Methodologies.

These tools involve enhancements to existing diagramming methodologies or new methodologies that take full advantage of the computer. The computer has potential to store enormous libraries and use complicated algorithms that could never be used in pencil-and-paper methods.

Using a computer we “access large data models; automate the normalization and synthesis of these data models; extract and edit portions of the data model; create navigation paths through the data; generate screens reports and dialogs” [M&MC]. We can also add sufficient detail to permit code generation.

3.5 Existing Nassi-Shneiderman Diagrammers

Before starting any development for N-S Chart, a study was done of any previously available tools for developing Nassi-Shneiderman diagrams. Very few of these are available and only two could be found. They displayed undesirable features that it was felt could be improved upon in N-S Chart.

BLUES® Information have produced a tool called BLUE/30 which is a module for creating and maintaining Nassi-Shneiderman diagrams [BLUE]. This module can use information created by other BLUES modules and is designed especially for the Apple Macintosh. The main drawback of this tool is that diagrams are not automatically resized and formatted evenly when changes are made to the diagram. Also, text may overrun symbol lines, and only one diagram may be open at once. Symbols for supporting block structures and parallel processing have been left out. A typical BLUES N-S diagram would look like:
The second program found is called the Stylus Toolset System. It was written for an IBM PC type machine by Stylus Software [STYLUS]. The Nassi-Shneiderman diagramming technique has been adapted almost beyond recognition to allow them to be drawn with the line characters in the standard IBM character set. The style of a Stylus N-S Chart is shown below:

```pascal
SoundDemo

var 
    Freq: real;
    I: integer;

Freq := 32.3;
for i := 1 to Octave
    Freq := Freq * 2
If Duration <> 0
    Sound (0);
    Sound(round(Freq * Duration));
writeln('Press any key to stop');
Until KeyPressed
SoftAlarm;
```

These tools provided some insight into useful features of a Nassi-Shneiderman diagrammer. On the other hand they displayed several areas that could be greatly improved and gave ideas for expansion of the technique.
4.0 Criteria for good diagramming methods

4.1 Attributes of good techniques

Diagrams need to be user friendly. These days there is an increasing end-user involvement in computing. Users should be able to understand systems analysts plans.

Computer aided systems analysis and computer aided design speed up the building of systems, improve the quality of results, and make systems easier to test and change. Hence a diagramming technique should be able to be easily drawn and edited on a computer screen.

It is desirable for program code to be generated automatically where ever possible. A good diagramming method should lend itself to straight forward conversion to code.

It is also desirable to have a technique that can be used to represent the system at all levels - from the highest overview to the finest code detail. The method should encourage stepwise refinement of the design.

Productivity of application design becomes more important as computers out number programmers. Therefore a technique must be fast to use.

4.2 Constructs to represent

There are several basic constructs that a diagramming technique for procedural languages needs to represent. They can be summarized as:

Sequence
Selection and Decision
Iteration and Repetition
Parallel execution

Every computer language uses the notion of sequence. Basic operations are done one after another in a serial fashion.

Almost all tasks involve selection and decision. Choices must be made between different paths or different courses of action. A particular task may or may not be done.
Most operations will require iteration and repetition. Statements may be repeated a certain number of times, or until a condition is met.

There is also a requirement these days to represent parallel execution. Several languages (especially simulation languages) allow the specification of tasks to be done simultaneously.

If a diagramming technique is to be successful it needs to be able to clearly represent these basic constructs.
5.0 Nasi-Shneiderman Charts

5.1 Why use N-S Charts?

Nasi-Shneiderman charts were introduced 15 years ago as an alternative to the flowchart. "Unfortunately the conventional flowchart language has aspects that make it both too powerful and yet too simple a language to model current programming techniques. These techniques tend toward a more restrictive control structure which the flowchart can not describe nicely. Certain control structures in programming languages, such as iteration, have no direct translation to flowchart language and must be built from simpler control structures, thereby losing the forest in the trees." [N&S]

Nassi-Shneiderman charts, however, provide a structured, hierarchical view of the program. Their main advantages over conventional flowcharts and similar diagramming techniques are:

- The scope of iteration and If-Then-Else clauses is well defined and visible. Alternative paths from If-Else and case statements are clearly separated, but parallel for easy comparison.
- The conditions on process boxes embedded within compound conditionals can easily be seen from the diagram.
- Blocks are representable and the scope of local variables is immediately obvious.
- Arbitrary transfers of control are impossible. There is no representation for a Jump or Goto. This supports the idea of structured programming [DD&H].
- Complete diagrams can and should fit on one page (i.e. there are no off page connectors).
- Recursion has a trivial representation.

However, N-S Charts as originally proposed do have some disadvantages. They have no facility for specifying data and data types. There is no provision made for links to a data model or a data dictionary. They are less appropriate for showing the high-level hierarchical control structure of a system. N-S Charts are easy to read, but they are not always easy to draw manually. Editing them on paper becomes very messy.
As will be revealed below, changes have been made to the diagrams to help solve some of these problems. The result of these changes is that the tool has become even more powerful and useful.

5.2 What are N-S Charts?

"Nassi-Shneiderman charts (N-S charts) represent program structures that have one entry point and one exit point and are composed of the control constructs or sequence, selection, and repetition. Whereas it is difficult to show nesting and recursion with a traditional flowchart, it is easy with an N-S chart. Also, it is easy to convert an N-S chart to structured code." [M&MC]

An N-S chart consists of a rectangular box representing the logic of a program module. This box is intended to be drawn on one page. When a N-S chart becomes too large, subfunctions should be separated out and drawn on another N-S chart.

Each basic control construct used in structured programming can be represented by an N-S symbol.

5.3 Nassi-Shneiderman symbols

The following paragraphs show how Nassi-Shneiderman symbols represent the basic constructs introduced above.

5.3.1 Sequence

Sequence is shown by a vertical stack of process boxes:

```

```

The process symbol is used to represent assignment, input and output statements as well as procedure calls and returns. A process symbol represents operations of arbitrary complexity. Hence any process symbol could be replaced with any other sequence of symbols, or even another whole N-S Chart.
5.3.2 Decision

Selection, If-Then Else or decision is shown by the If-Else symbol:

![Diagram of If-Else symbol]

The central triangle contains a boolean expression, while the left and right triangles contain a True or a False (or other notation) to represent the possible outcomes. The process symbols (shaded rectangles) contain the operations to be performed depending on the outcome of the test. The column of the left shows the actions for one outcome, and the column on the right the actions for the other. As indicated above, these process symbols can be replaced with any combination of other symbols.

5.3.3 Selection

The Case structure in which a selection is made from multiple mutually exclusive choices, is shown here:

![Diagram of Case symbol]

It may be noted that the Case symbol and the IfElse symbol look rather similar. This is intentional as they represent similar concepts. However a decision was made to alter the Case symbol slightly for reasons elaborated below.

5.3.4 Iteration

Repetition or iteration is indicated by the DoWhile or DoUntil symbols:

![Diagram of DoWhile symbol]

The body of the iteration is a structure of arbitrary complexity. The vertical part of the symbol to the left of the body, extends for the full length...
of the symbols within the loop. This part of the symbol will be referred to as the upright.

5.3.5 Block

A block can be represented by the Begin-End symbol:

```
[-----------------------------]
```

Again the upright on the left extends for the full length of the symbols within. A block can be used to represent scopes of local variables or even begin-transaction end-transaction pairs.

5.3.6 Parallel Execution

Parallel processing can be handled with the symbol:

```
[-----------------------------]
```

As with the Case symbol, any number of alternatives may be used inside the symbol. It expresses the idea that each alternative is started and executed concurrently.

5.4 New Symbols

Two more symbols have been added to the technique to accommodate the specification of input and output data. The symbol to specify data input to the routine is:

```
[-----------------------------]
```

and the symbol to indicate the data produced by the routine is:

```
[-----------------------------]
```

The rounded corners help to remind the user that these are special symbols, and may not be used just anywhere. Each chart has just one Input symbol at the start and one Output symbol at the end. Hence the
rounded corners fit into the scheme of symbols and serve to neatly round the corners of the whole chart.

5.5 Advantages of N-S Charts

N-S Charts exhibit most of the good characteristics of diagramming techniques mentioned above.

They are user friendly and easy to read. The shapes of symbols reflect their meaning and suggest how they should be used in a diagram. The rectangular nature of N-S Charts lends itself to be rapidly drawn by computer. Most symbols are made up of simple lines.

Program code can generated in a straightforward way directly from the diagrams. Each symbol can be directly mapped to a construct in a programming language. Perhaps most importantly, they are orientated to structured, stepwise refinement design. Any particular process symbol may represent another arbitrary complex N-S Chart.
6.0 Why use a computer?

6.1 The Benefits

Interactive diagramming on a computer screen has major advantages. It speeds up the process greatly. It enforces standards. It can automatically balance, tidy, format and arrange diagrams. The computer may apply many checks to what is being created. It can automate the documentation process. The computer enforces discipline and permits types of cross-checking, calculation, and validity checks that people often can not be bothered to apply. Even the best of intentions may be subverted by time constraints, project deadlines, etc.

6.1.1 Seamlessness

The diagram drawn may be arbitrarily large - without seams. Drawing is not restricted by manageable paper sizes. When the diagram is printed it may be tiled to fit onto several pages for printing. However it is better if the diagramming technique enforces the use of one page diagram units. Larger diagrams can also be scaled and sized to fit on to particular size page. There is no need for an array of disjointed off-page connectors.

6.1.2 Windows

The whole of a diagram may be seen by scrolling. Scrolling may be by line, page or section at a time. A good diagramming tool will allow the user to view the whole diagram (reduced to fit on the screen) and perhaps allow it to be moved and positioned on the page.

There may be multiple windows onto a large diagram. Several parts of a large chart may be visible at once which removes the need to flick back and forth between pages.

6.1.3 Consistent Style

Diagrams drawn with an automated tool will all be drawn in a single style. This makes understanding the diagrams much simpler since there are no variations made to the technique (intentionally or unintentionally)
by the analyst or designer. A degree of flexibility and provision for
customisation is possible to avoid needless rigidity.

6.1.4 Ease of Editing

One of the greatest advantages of computer based diagramming tools
is the ease with which diagrams may be edited. Symbols may be inserted
and deleted without having to redraw or move other parts of the diagram
(since they are automatically moved by the application). Pieces of the
diagrams may be copied and pasted elsewhere, hence saving the
redrawing of parts of the diagram that are repeated or similar.

With a technique like N-S Charts where symbols are nested inside
each other, editing diagrams manually on paper can become a
nightmare. If symbols are to be inserted into the middle of a deeply nested
structure, each layer of symbols outside the insertion point need to be
modified to accommodate the change.

The N-S Chart application developed in this project automatically
resizes symbols and adjusts the chart after a change has been made.

6.1.5 Explosion/Implosion

The design process involves stepwise refinement, moving down
through the levels of hierarchy. Sometimes tasks at a particular level
turn out to be too simple for a level of their own, or may be too complex and
so have to be split into an extra level. This involves moving parts of a
diagram up one level, or down one level in the hierarchy.

This process can be greatly simplified in an automated tool with
options for collapsing parts of a diagram into a sub-diagram, or for
automatically including a sub-diagram into the level above.

6.1.6 Provision for Data

Manual diagramming methods often have little or no provision for
specifying data items. Those that do tend to make use of constructs that
seem unnatural or 'tacked on'.

An automated tool can add facilities to allow data to be specified and
gives potential for interface to other system development tools.
6.1.7 Potential for interface to larger environment

There are very few (if any at all) diagramming techniques that are good for diagramming every stage and level of system development - from a high level overview of the structure to detailed program logic and from data flows between units to detailed data structure design. It is better for diagramming techniques appropriate to the task to be used.

However, this causes problems with interactions between the various techniques. Automating these techniques provides the opportunity to produce integrated diagramming tools that use several diagramming techniques while enabling them to be interfaced in an appropriate way.

The Input and Output symbols added to Nassi-Shneiderman diagrams in this project were designed to enhance this potential while at the same time providing a means to specify inward and outward data flows. N-S Charts could be integrated with other automated tools like Entity Relationship diagrammers and Dataflow diagrammers. Dataflow diagrams describe the flow of data throughout the system. At the lowest level, data is input to a process and is transformed to produce output data. N-S Charts would be excellent for describing this transformation.

6.1.8 WIMP Environment

The environment of Windows, Icons, Mouse and Pull down menus makes a diagramming tool easier and more intuitive to use. These days EUC (End User Computing) is becoming more and more feasible as systems become more user friendly and easier to use. It is important for the user to be able to specify the requirements she has for a new system. The WIMP environment helps to make this more of a reality.

6.2 Disadvantages

The advantages of automating a diagramming technique greatly outweigh the problems that arise. However the disadvantages should not be overlooked.

A computerized method loses a certain amount of flexibility. A system will need to apply constraints at some point. Two examples of this can be seen in the N-S Chart application that has been developed in this project.
In N-S Chart the width of each alternative in a Case, IfElse or Parallel symbol must always be equal. Often one side of a IfElse symbol is just an exception case and so could be made rather narrower, leaving more room for the main alternative. However it was decided to forfeit this option in favour of automatic resizing. Having even-width columns allows new widths of columns to be automatically calculated and guaranteed to be consistent. If different widths were allowed, fractional widths have to be stored and rounding discrepancies cause problems aligning symbol columns exactly.

The other example is the decision to limit symbol heights to an integer number of 'lines'. This was a result of searching algorithm design and is discussed more fully in chapter 9.

Another problem with computer generated diagrams is that of display. Until recently computers capable of displaying graphics and multiple sized text were rather rare. Also not many commonly available printers were capable of producing high quality graphics diagrams. However with the advent of computers like the Macintosh this is becoming more standard.

On the other hand minor display problems still prevail. It is still more difficult to fit a piece of text into a small irregular shape on a computer screen, than is it to do it by hand on paper. A change was made to the Case symbol to help solve this problem in N-S Charts.
7.0 N-S Chart: The Application

The paragraphs below describe the features of N-S Chart; the automated diagramming tool developed in this project. Details of the implementation of each of these features can be found in the following chapters.

For full details on how to use N-S Chart, the reader is urged to read the N-S Chart User's Guide [NSCUG].

7.1 Drawing Charts

Charts are drawn by selecting a symbol tool from the tool palette and clicking on the diagram to draw symbols. When a symbol is drawn it is automatically sized and fitted into the diagram. If the symbol is inserted into an alternative of a IfElse, Case or Parallel symbol, the filler symbols in the other alternatives are sized automatically to keep the diagram even.

Chart B shows how the diagram is changed from Chart A after a process symbol is inserted into the DoWhile symbol. Notice that the DoWhile symbol is enlarged to incorporate the new symbol, and a new filler symbol has been added to alternative 2 of the case symbol. The other two fillers have been enlarged to even up the bottom line. Details of the routine to adjust these filler symbols are given in chapter 9.

The Insert / Replace mode may be changed by clicking on one of the two icons at the bottom of the tool palette. Insert Before /
After mode can be specified by choosing **Insert Before** of **Insert After** from the **Settings** menu.

Insert/Replace mode specifies whether new symbols added or pasted should be inserted after (or before) symbols selected in the chart, or whether they should replace the selected symbols. Insert Before/After mode indicates whether the inserted symbols should be placed before or after the selected symbols.

Text may be added to symbols with the text tool. Clicking on a symbol opens up a dialog box that allows text to be entered and edited.

Cases may be added to and deleted from Case and Parallel processing symbols with the Add Case and Remove Case tools. To use these tools the user clicks on any symbol in the alternative in question. The Add Case tool will add a blank alternative after (or before depending on the insert before/after mode) the one clicked on. The chart C below shows Chart B after a new alternative has been added to the case statement. The dot shows where the mouse was clicked. As might be expected, the Remove Case tool deletes the case clicked on.

Symbols may be made taller or shorter with the Shrink and grow tools. Clicking on a symbol with the grow tool increases its height by one line. The Shrink tool shortens a symbol by one line. Symbols may not be made shorter than one line high.

On-line help about N-S Chart may be obtained any time by clicking the Help button on the **About N-S Chart** dialog. A window with a list of topics is displayed from which help subjects may be selected.
Paralleling the idea of structured design, charts may be built in a top-down tree structure. Subcharts may be opened below the parent chart, and sub-subcharts from the children, etc.

A subchart may be opened below a process symbol by selecting that symbol and choosing **New Subchart** from the **Chart** menu. A subchart may be removed in the same fashion by choosing **Remove Subchart**. A gray border is put around a process symbol to indicate that it represents a whole subchart. Chart D above shows this border.

### 7.2 Editing Charts

Full editing facilities are provided for charts. Most editing operations rely on certain symbols being selected.

Symbols are selected with the Grab tool by either simply clicking on a symbol or by clicking down and dragging the mouse up or down the chart. Only symbols at the same level may be selected together and when a symbol is selected, all symbols inside it are also selected. A whole chart may be selected by choosing **Select All** from the **Edit** Menu.

Selected symbols may be **Cut**, **Copyied** or **Cleared** by choosing the appropriate item in the **Edit** menu. The **Erase** option removes all symbols from a chart except for the basic Input and Output symbols.

Symbols that are cut or copied are put in the clipboard. The contents of the **Clipboard** may be viewed at any time by choosing this item from the **Windows** menu.

To paste the contents of the clipboard into a chart, a symbol is selected and the **Paste** item chosen. The clipboard symbols are either inserted after the selected symbols or in place of the selected symbols - depending on the insert / replace mode.

### 7.3 Navigating around Charts

As mentioned above, charts may be built in a hierarchical tree structure. For example, a system to be written in Pascal should be
diagrammed with one procedure per chart. When working on a chart, the
designer will want to move up and down the hierarchy of charts, or
perhaps just go directly to a particular chart.

A subchart may be opened from it's parent by double clicking on the
symbol that represents it. The double click opens a window with the
subchart in it.

Movement back up the chart structure can be achieved by
using the 'Up One' tool. Clicking on this icon closes the current chart and
opens its parent.

All top level charts appear in the Windows menu and may be opened
by selecting their name. Other subchart windows may be opened from the
Window List dialog. This dialog is obtained by choosing the Window List
option in the Windows menu.

7.4 File Operations

Charts may be Saved and old charts Opened using options from the
File menu. When Save is chosen from the File menu, the whole of the
current chart is written to the file specified.

Charts may also be saved as pictures in the standard PICT format.
To do this choose Save Picture... from the File menu.

A chart may be printed by choosing Page Setup... and Print... from
the File menu. A short cut option, Print All..., prints all charts in the
hierarchy from the current subchart down.

Before printing, the whole chart may be viewed and positioned on the
page with the Page View window. This window is opened by choosing
Page View from the Windows menu. A scaled picture of the chart may be
dragged around the window to the desired printing position.

Pascal pseudo code may be generated by choosing Generate Pascal
from the Chart menu. This option opens up a dialog to allow the user to
select various parameters for generating the code. A file name can then
be specified to save the the code to. This file as saved as text and so can be
read into any text editor for editing, or any application for compilation and
execution.
8.0 User Interface Design

How a system looks and feels has a great deal to do with its success. A system with excellent features can fail if it is difficult to use or takes too long to learn. Operations that are used very often should be quick to execute and even seldomly-used commands should not be hard to find. If the interface is user friendly and intuitive to use, the tool becomes more useful.

Before any coding was started for the N-S Chart application, much thought and research was put into the design. How should the interface look? How would it feel to use? The main decisions about the general design are described here.

8.1 Point to draw

When drawing with pencil and paper, you put the pencil on the paper, and drag to draw a line. This idea has been followed in drawing applications like MacPaint and MacDraw. Lines and shapes are drawn by clicking the button and dragging the mouse.

With a diagramming technique like Nassi-Shneiderman charts, there is no need to draw individual lines. There is a small finite set of symbols that may be drawn and no other shapes are allowed. Hence after a symbol type has been selected, one click is all that is required to draw that symbol. No dragging of the mouse is needed as all symbols are sized automatically by the program.

8.2 Continual chart integrity

N-S Charts are made up of many different sized rectangles. When drawing charts these rectangles tend to become uneven sizes and do not naturally match up. Therefore an important aspect of the design was to ensure that diagrams stay even and balanced. When parts of a diagram are cut and pasted, they should be sized to fit and the rest of the diagram altered accordingly.
8.3 Multiple charts

Typically there will be more than one person developing a particular system. Hence different parts of the system will be diagrammed separately and saved in different files. It is conceivable that parts of diagrams or even whole subcharts may need to be copied from a chart in one file to a chart in another.

Hence it was decided that the program should allow the user to work on more than one chart at once. This would allow symbols to be copied between charts and also provides a more flexible working environment where parts of a diagram may be temporarily put in a separate chart while other work on the main chart is being done.

8.4 Management of complexity

An important consideration was how the application would behave as the charts being worked with become very large (real size). Speed and performance should not be down-graded so that the tool becomes unbearably slow to use. Many diagramming tools fail in this respect. Redrawing large diagrams takes a significant amount of time and so the application should be designed to handle operations efficiently.

8.5 Code Generation

An important advantage of automated diagramming tools is their potential to generate code from the diagrams. Hence it was decided that N-S Chart should be able to generate code from Nassi-Shneiderman charts. The language chosen was Pascal because of its widespread use in academic circles (where it is most likely this diagrammer would be used).

There is potential for other languages to be generated with a minimum of extra effort.

8.6 Structured Design

The benefits to be gained from structured design and top-down programming are well known and accepted. Hence it was decided the style of drawing charts should parallel and support the ideas of structured design. The notion of stepwise refinement is supported by allowing a hierarchy of charts.
A subchart may be opened on any process symbol. That symbol then represents the entire subchart. The subchart may be opened at any time from the parent by double clicking on the process symbol that represents it.

**8.7 Menus vs Tool Palette**

In most Macintosh applications actions are performed by either choosing items from menus, or selecting icons on a window. Much consideration was put into determining what operations should be performed by choices from menus, and what would be better done by selecting icons from a palette like the one on the left. A tool palette takes up room on the screen and the icons can be confusing if they are not designed well. However a palette make the selection of options much faster than choosing from a menu, and has the added advantage of showing the current selected tool at all times.

It was decided to use a palette to select between the seven N-S symbols for drawing, and for selecting various other tools for manipulating the charts. A summary of the meaning or each of the symbols can be found in chapter 3 of the N-S Chart User Guide [NSCUG].
The tool palette (on the left) can be thought of as being composed of 3 main sections. Only one icon may be selected from each section at any one time. Section 1 contains the seven symbol icons (1A) and icons for other often used operations. The highlighted (inverse) icon shows the current tool. Section 3 contains icons to select the current Insert or replace mode (refer below).

8.8 Cursors

When the action of clicking on a window can do many different tasks depending on the current tool, it is very easy to forget what tool you are using and click with the wrong tool. Even though the current tool icon is highlighted in the palette, the eyes of the user are fixed on the cursor pointer.

Hence it was decided to change the cursor shape to a smaller representation of the tool icon whenever a new tool is chosen from the palette. The cursors used are:

- Standard arrow pointer used when selecting from the palette, menus etc. Also used for dragging windows and scrollbars.

- The Finger cursor is shown when using one of the symbol tools. Point where you want the symbols, click, and it appears!

- The Grow cursor is used when the Grow tool has been selected.

- When the Shrink tool has been selected, the Shrink cursor is displayed.

- The Hand cursor is used when selecting symbols. It is also shown on the palette window where the reduced view of the chart may be clicked down on and dragged around.

- This is the Add Case cursor. Click on the top symbol of the alternative you want the case inserted before or after.

- The Remove Case cursor deletes an alternative of a Case or Parallel symbol.
The inspect tool cursor has not been implemented.

By simply looking at the cursor, the current tool type can be determined.

**8.9 Text in Symbols**

One problem with drawing N-S symbols is that it is difficult to fit text into the triangles of IfElse and Case symbols. The Case symbol is particularly awkward when it contains many alternatives. For this reason the symbol was modified slightly to aid the presentation of the alternative captions.

The Case symbol was changed from the one on the left to the symbol on the right which ensures that each alternative has at least one whole line to write the alternative caption. As the symbol is expanded, the text spaces automatically grow in proportion:
9.0 Implementation Details

9.1 Development Environment

N-S Chart was written in Lightspeed™ Pascal on the Apple Macintosh with the aid of the Programmer's Extender™ libraries of routines.

9.1.1 Lightspeed Pascal

Lightspeed Pascal provides an integrated programming environment. It includes a compiler, a fast linker, comprehensive debugging tools and a text editor combined with an intelligent code management system [LSP].

Lightspeed Pascal uses a Project Document to organize the files that make up a program. Source files (units) must be added to a project before they can be compiled and run. The project indicates what order the code modules should be linked, and also specifies in what segment of the Macintosh's memory the code files should be placed. When the Go command is given, each file unit is complied, linked and the project is executed. The system is intelligent so that only those files that have been changed, or that are effected by changes, are recompiled.

The debugging facilities allow the program to be halted, stepped through and resumed. 'Stops' may be put into the code at any time to halt the program. Any variables in scope can be examined and evaluated with the full interactive use of built in Pascal functions. The values of variables may even be changed, and extra bits of code may be executed while the program is halted to 'patch' an error before the program is resumed. These facilities simplify greatly the process of tracking down errors.

However the system is not perfect, and it does have the odd bug. The most annoying glitch is the tendency for the linker to occasionally mess up and effectively destroy the project. The dialog below shows one of the nauseating link errors that can not be resolved - the project has to be rebuilt again from scratch.

9.1.2 Programmers Extender Libraries

The Programmer's Extender libraries are a set of routines that provide an interface between the programmer and Toolbox routines. Extender routines tend to call frequently used combinations of toolbox routines and often do extra error checking. The N-S Chart Technical Report [NSCTR] mentions various advantages and disadvantages found with using the Extender libraries.

9.2 Basic Data Structures

All operations on charts revolve around the two basic data structures used: the chart record and the symbol record.

The word 'Chart' tends be overloaded with two meanings in this chapter. The word 'Chart' will be used to mean a whole chart - top level diagram and all it's children, it's children's children, etc. The word 'subchart' will be used to mean a single diagram at a particular level.

9.2.1 Chart record

A chart record contains data about a subchart. Data is stored here about the subchart in general, or about all of its symbols. The record has the following structure:
The subcharts title
Top left corner of the subchart in the window
Top left corner of the subchart on paper
Overall subchart width
The height of one line in the subchart
Width of a DoWhile, DoUntil or Case symbol left side
Text size used in the subchart
Number of the point size item in the style menu
Number of the font used in the subchart
The fonts item number in the font menu
Justification for the subcharts captions
Does this chart record store a valid chart?
Amount the subchart has been scrolled
Has this subchart been saved yet?
Has the subchart been modified since last save?
Pointer to the first symbol (record) in the subchart
The subchart's window and graph port
Handle to the picture representation of the chart
Pointer to the symbol this subchart represents
The number of this subcharts very top level ancestor
This subchart's immediate parent
Pointer to a list of this subchart's children

The title is set whenever a Save As... is done or when it is changed explicitly with Chart Name. The PointMItem and FontMItem fields are used to check and uncheck items in the menus after changing between charts. The Justify field is an integer (rather than say an enumerated type) for compatibility with toolbox routines.

The exists boolean is used when scanning the chart array for a vacant spot to slot in a new chart. Windo is variant record of two types of pointers, which allows fast access to fields in the charts window record and the associated graph port record. The topChart field gives direct access to the subcharts top level ancestor. This is used whenever whole chart operations like Save... and Generate Pascal... are chosen. The childCharts list is used for quick navigation down the hierarchy of charts.

The top level chart record of each chart is stored in an array. If the chart has any subcharts, they are stored in a tree structure hanging off the top level chart record. If N-S Chart had two charts open; the first a simple single level chart, and the second with a hierarchy like this:
Links are also kept in both directions to the symbol the subchart represents. For example in the above diagram, parentsym field in Chart 2.2's chart record would point to the symbol in Chart 2 that represents it. The subchart field in the symbol's record (q.v) would point to Chart 2.2.

Charts are stored in a tree structure that parallels the hierarchical nature in which they are built. If charts were allowed to be built in a more flexible network structure, problems with integrity would arise. In particular, major problems start appearing if a chart were allowed two or more parents. For example if you change the inputs of a subchart that has several parents (some of which you are unaware of) the integrity of the chart is immediately in jeopardy.
9.9.2 Symbol record

The symbol record stores information about a single symbol. The data stored here is different for every symbol in the chart. It looks like this:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>symb</td>
<td>The symbol type. Process, IfElse etc.</td>
</tr>
<tr>
<td>CaptStyle</td>
<td>Symbol caption text style</td>
</tr>
<tr>
<td>symtoplen</td>
<td>Length of top part of symbol</td>
</tr>
<tr>
<td>insidelen</td>
<td>Total length of symbols &quot;inside&quot; this symbol</td>
</tr>
<tr>
<td>symbotlen</td>
<td>Length of bottom part of symbol</td>
</tr>
<tr>
<td>totsymlen</td>
<td>Total length of whole symbol</td>
</tr>
<tr>
<td>width</td>
<td>Width of the symbol</td>
</tr>
<tr>
<td>across</td>
<td>Position across horizontally in the chart</td>
</tr>
<tr>
<td>kind</td>
<td>The kind of symbol: normal, filler or dummy</td>
</tr>
<tr>
<td>NoAlts</td>
<td>Number of alternatives following the symbol</td>
</tr>
<tr>
<td>caption</td>
<td>The symbol caption</td>
</tr>
<tr>
<td>Alterns</td>
<td>Array of alternative records (see below)</td>
</tr>
<tr>
<td>parentsym</td>
<td>The symbol this symbol is inside</td>
</tr>
<tr>
<td>lastsym</td>
<td>Symbol before this one in the chart</td>
</tr>
<tr>
<td>nextsym</td>
<td>Symbol after this one in the chart</td>
</tr>
<tr>
<td>subchart</td>
<td>Pointer to the subchart this symbol represents</td>
</tr>
</tbody>
</table>

The alternative record stores the name of the alternative and pointers to the first and last symbols of the alternative:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>capt</td>
<td>The Alternative caption</td>
</tr>
<tr>
<td>firstsym</td>
<td>Pointer to the first symbol of the alternative</td>
</tr>
<tr>
<td>fillersym</td>
<td>Pointer to the filler (last) symbol of the alternative</td>
</tr>
</tbody>
</table>

Symbol records are linked together in a rightward spreading structure that is a cross between a linked list and a tree. It could perhaps be thought of as a linked list of trees of linked lists of trees ... etc. Consider the simple chart on page 38 (which iteratively calculates a factorial) and the corresponding structure of symbols (on page 37) used to store it:
The Dummy symbols before each list of symbols in an alternative are to simplify insertion of symbols at the start of the list. Dummy symbols are never displayed.

The filler symbols at the end of each list are to keep the lengths of each alternative the same. In the example, the filler after the `NFact:=1` symbol has a length of two, whereas the filler after the `DoWhile` symbol has a length of zero. This makes the total length of each alternative after the `IfElse` symbol equal to three.

As with the structure of chart records, the tree structure of symbols parallels the hierarchy of symbols.

### 9.3 Operations on the basic structures

#### 9.3.1 Finding a Symbol

One of the main problems to be solved was to find a particular symbol in the data structure from a mouse click on the window. From the mouse click position (in local coordinates of the window) the position in the diagram must be found (after adjustments for scrolling, boarders etc). The coordinates of the point in the diagram must then be converted to a particular symbol in the chart.

This problem was made somewhat simpler by limiting the height of each symbol to an integer number of lines. The line height may be changed to any number of pixels, but the height of a symbol is limited to a whole number of lines. This meant that a point on the chart could be converted to a line number (simply by dividing by the line height). The symbol at this line could then be found.
The first approach to solve this problem was to have an array (or perhaps linked list) of pointers (one for each line of the chart) pointing into the symbol tree structure at the symbol in that place in the diagram. This proved to be impractical due to the time it would take to update this array (or list) every time the chart was changed.

The method finally chosen was to search the chart from the start each time to find the symbol. The recursive algorithm moves down the chart skipping over symbols, until the point is found to be inside the current symbol. The alternative within the symbol that contains the point is then found, and searching continues down the symbols in that alternative.

The Nassi-Shneiderman chart below illustrates this algorithm:

**The FindSym Algorithm**

```
Input line, across, sline, sym
InSym := is the point within symbol sym ?
then
    FinSym := the current sym
PastSym := is the point past the end of this symbol?
then
    if not lastsym
        FinSym := the current symbol following this one
    else
        i := 1
while point is further right
    [i := i + 1]
    FinSym := in altern i
else
Return symbol clicked on, and line of symbol
```

At first this approach may seem to be rather slow. However it should be pointed out that the maximum length of a subchart (in symbols) is 66 (length of a page / average line height) with the average being closer to 15. The depth of nesting of symbols is only practical to about six levels due to the reduction in width. Hence on average for a large diagram, the search should only involve skipping along 16 links at most (if the symbol is at the bottom of the chart). Trials on large charts have shown that this time is negligible.
9.3.2 Chart walking

Many routines have cause to do something to every symbol in a subchart, or part of a subchart. They all use this general algorithm to skip through the tree:

1. Input a symbol
2. Do the operation to the current symbol
3. for each alternative the symbol has
   a. Call this routine for the first symbol in the alternative
5. then
6. if not last symbol
7. Call this routine for next symbol

Hence the tree structure is traversed in preorder fashion. The deletion and disposing routines do this in a postorder pattern so as not to cut one's own rope (i.e. delete the parent before the child - and so lose the links to the child).

9.3.3 Drawing Symbols

Charts are drawn by walking the symbol structure (with a routine like the one above) drawing each symbol in turn. Getting each symbol exactly aligned with the symbols above, below, left and right took much adjustment. The routine had to be able to draw any of the nine symbols at any height and width.

Process symbols are drawn in several different ways. Dummy symbols and zero length filler symbols are not drawn at all. Fillers may be drawn in outline or shaded to help the user distinguish them from empty process symbols.

9.3.4 Text in Symbols

It was decided that a set amount of text would be stored for each symbol, but only the text that could be fitted in the symbol would be shown. To see the rest the user can examine the symbol with the text tool.
The alternatives were to restrict text being entered to that which could actually fit in the symbol, or display all text entered. The first option is undesirable because this would severely limit the text space in some captions. Also the text capacity of symbols changes as symbols change in overall size. Drawing all text entered would lead to a very cluttered diagram with text overwriting symbol boundaries. A way to solve this problem would be to automatically enlarge symbols to accommodate any text entered. However different size fonts lead to different sizes and this approach would make narrow symbols disproportionately long.

9.3.5 Inputting Text

There were several ways to go about letting the user enter text into symbols. The first method that seemed reasonable was to bring up a dialog for text to be entered whenever a new symbol was added. The problem with this is that often it is desirable to sketch the outline of a diagram by just drawing the symbols, and then go back and add the text captions latter. It also makes the drawing process rather slow.

The nicest way to enter text would be to type it directly on the diagram using a text tool. This turned out to be rather difficult to accomplish. Having too many text fields mingled with the picture also would make the scrolling and redrawing of charts very slow.

The method finally chosen was to ignore text when the symbols are drawn, and to have a text tool to allow text to be entered into symbols. The text pointer is clicked on a symbol and a dialog box opens to allow the text of a symbol to be edited. This approach makes the drawing of symbols very fast (just one mouse click).

9.3.6 Tool Palette

Implementing the tool palette proved to be more difficult than it at first looked. It was first designed to be a separate window that could be moved around the screen to where the user preferred it. However because of the way the Macintosh window manager works this proved to be impractical.

Many windows may be open at once, but there is always one front window that is the active window. It is not possible to have more than one window active at once. (Some applications like Pagemaker get around this but it could not be determined how this was done). Hence if the palette was to be a window, the current window needed to be changed back and forth.
between the chart window and the palette window. Without any intervention, this would mean two clicks on an icon in the palette to select it - one to activate the window and one to actually select the tool.

The second approach was to automatically activate the palette window when the pointer was moved over it. This solved the double click problem, but caused another problem just as undesirable. When a window is deactivated, the toolbox automatically unhighlights its title bar and scroll bars. No way could be found to avoid this. Hence the window's bars flickered from highlighted to unhighlighted as the pointer was moved across the screen.

Eventually the window was done away with and the palette was drawn directly on the screen as a simulated window. This gives the effect of having two highlighted windows at once. The disadvantage of drawing the palette direct to the screen is that it is wiped out by windows being put on top of it. Because of this it was necessary to write a new window dragging routine so that none of the N-S Chart's windows could be dragged over the top of the palette. Any time a dialog wipes part of the palette, it is redrawn afterwards. The Palette option was added to the Window menu to enable the user to redraw the palette in case it is wiped by a desk accessory or screen saver.

9.3.7 Case and Parallel symbols

Much consideration was put into how best to draw Case and Parallel Processing symbols. Should the designer be prompted for the number of alternatives to draw initially, or should the symbol be drawn with some default number? Once the symbol is drawn, do you allow the number of alternatives to be changed, or should the symbol just be deleted and another one put in its place?

The method used in answer to the first question was to draw the symbols with a default number of alternatives. The default value may be set by the user to make it more appropriate to the task being diagrammed.

Once a Case of Parallel symbol has been drawn, cases (alternatives) may be added or deleted with the Add Case and Remove Case tools.

9.3.8 Insert or Replace Mode

When editing text in a word processor, you usually insert text before or after existing text. However you sometimes replace text by selecting it,
and typing the new. It was decided that this style of editing would be used for editing charts.

This led to the Insert and Replace modes. When in Insert mode, symbols are added after (or before) the symbol clicked on (q.v.).

Symbols can be replaced by selecting replace mode. In this mode, instead of a new symbol being inserted into the diagram, it replaces the selected symbols.

Insert and Replace modes also apply to Paste.

**9.3.9 Insert Before and After mode**

The parallel idea to entering text needed to be slightly modified due to the nature of the symbols. A diagram has no blinking cursor to indicate a current insertion point. It was deemed impractical to insert a symbol between two others by clicking on the joining line. Macintoshes have a fairly sharp screen but trying to click on a single pixel line all the time would still involve a reasonable amount of squinting! If the inserting area was enlarged to be in the 'general area' of the line, the user would be unsure if she has the exact place.

The solution to this dilemma was to make an entire symbol the area to click on. Hence when inserting into a chart, some convention must be established as to whether the symbols will be added before or after the symbol clicked on. This is the reason for the Insert Before / After mode. Most of the time Insert After is all that is required. Insert before is only needed when symbols are to be added into the first place in an alternative.

Insert Before / after mode also applies when adding alternatives to Case and Parallel symbols. An alternative may be added before the first column by selecting **Insert Before** and clicking on the first alternative with the Add Case tool.

**9.3.10 Selecting and highlighting**

It was decided that it should only be possible to select symbols at a single level. Selecting symbols from different levels causes all sorts of integrity problems when actions are to be performed upon the selected symbols. Consider the selection drag path, and two possible symbol selection patterns.
Cutting or Copying the selected symbols on the left would be a little strange but could be accomplished. However, what would you do with the extra 2 symbols on the top if it was pasted elsewhere? This would also cause problems with replacing selected symbols with others. For these reasons when the mouse is dragged over nested symbols, the parent at the same level as the rest of the selected symbols is selected (along with all the symbols within).

9.3.11 Cutting and Pasting

The method behind cutting and pasting symbols is rather different to that for cutting and pasting text. Instead of just a simple buffer of characters in the clipboard, a whole tree structure of symbols need to be stored.

A problem was found with cut, copy and paste where symbols with subcharts were selected. If a symbol that represents a subchart is is deleted, its subchart should be deleted. When it is cut to the clipboard, the subchart should be carried with it so that when it is pasted, the same substructure is restored. However this causes a problem if it is allowed to be pasted again. The same problem arises with copy.

If two copies of a symbol with a subchart are pasted, the subchart then has two parents and the structure is no longer a tree (see Chart Structure above). There are several possible solutions to this dilemma.

One is to forbid the cutting and copying of symbols with subcharts. This would become very annoying and impede the development process. An intermediate solution would be to allow these symbols to be cut but not copied. The clipboard could then be restricted so that the symbol in question may be pasted only once.

The solution finally chosen was to make a complete copy of the whole subchart structure whenever it is pasted. This leads to several copies of the same diagram, but maintains better chart integrity. A copy and paste
is then a true copy at all levels rather than just at the level being worked on.

Copying a subchart may involve copying several descendants of a chart as well. For each chart, a new window must be created and their names inserted into the window list.

However, due to time limitations, this has not been implemented. In N-S Chart, if symbols with subcharts are copied, only the symbols themselves are copied, and not their subcharts.

### 9.3.12 Clipboard

The clipboard proved quite difficult to maintain correctly. The problem arises because it is possible to have the clipboard window open on the screen when a cut or copy is done. The clipboard window should then be immediately updated to reflect the changes to the clipboard.

This problem was solved by making the clipboard a special case of a chart window. Hence all picture-drawing routines and screen-updating routines written would work for the clipboard as well.

### 9.4 File Operations

#### 9.4.1 File Format

When Save is chosen from the File menu, the whole of the current chart is saved to a file. The charts are saved by traversing the tree structure of subcharts, writing each subchart's chart record in turn. Within each subchart, the symbol record of each symbol is written to file. Only necessary parts of the records are written and all text fields are saved with a length byte and all blanks removed. This compression saves a great deal of file space since text fields take up over 90% of a symbol record in memory.

When the symbols and charts are read in again, the structure is restored and all pointers connected by the order in which the symbols are read in. This avoids the need for any unique ID numbers and tables of links being saved to the file.
9.4.2 Printing Charts

Printing is done by calling Extender routines to print the Quickdraw picture of the chart. There appears to be an error in memory management in one of the library routines which causes an 'Out of Memory' error when printing multiple charts. For this reason the Print All option has been unhighlighted to avoid a system error in the application.

9.5 Other Operations

9.5.1 On-line Help

The on-line help system is modelled after the successful system found in Microsoft's Word and DiskFit programs. It involves a two stage help process. The first is a list of topics with short comments after each title. This may be enough to prompt the user's memory about a particular command. The second stage is another page of information about the command.

The help information is stored in a separate text file. The type and creator of this file has been changed so it can not be edited by mistake. The topic headings are stored in the start of the file and are read in to make the list in the topics' dialog. When help is requested on a particular topic, the file is scanned for the help page on that topic.

Sequential access to the file makes finding the topics at the end of the file a little slow but meant that help topics could be simply added to the file without having to alter the code or calculate indexes into the file.
10.0 Possible Extensions

10.1 Expand / Collapse

Commands for expanding and collapsing groups of symbols could be implemented. The collapsing action would involve selecting a group of symbols, and choosing collapse from the chart menu. The symbols selected would be replaced in the chart by a single process symbol that represents them all. The symbols would not be removed from the chart structure, but just effectively removed from view.

The opposite operation, expand, would reverse this process. The placeholder process symbol would be removed, and the symbols it stood for would be redrawn in the chart.

These operations are distinct from the Make Subchart and Include Subchart commands in that the symbols are not actually made into a new subchart, but just temporarily hidden.

The advantage of these operations would be that large charts could be made more manageable by hiding parts that are already complete. It would make the visible size of the chart smaller so that less scrolling around the window would be needed and redrawing the chart would be faster.

10.2 Undo

Perhaps one of the most useful extensions would be to enable operations to be undone. This would involve remembering details of the last action so that it could be reversed. There are several problems involved in storing these details due to the large range and types of actions that it would be useful to reverse.

10.3 Data Integrity

The Input and Output symbols added to Nassi-Shneiderman diagrams in this project could be developed further by deciding on a specific format for defining data flows. Checks could then be added to N-S Chart to determine whether the inputs for subcharts are available and
that the outputs are indeed created by the routine. There is potential for adding checks to ensure data conservation.

10.4 Diagrammer Integration

One advantage of automated diagramming techniques is the potential for integration with many other techniques. This was kept in mind throughout the design of N-S Chart.

N-S Chart could be combined with other diagrammers to make a much more powerful diagramming tool. The data specified in the Input and Output symbols could be used to connect N-S Charts with other techniques such as dataflow diagrams.

10.5 Generating Other Languages

Pascal was chosen as the language to be generated partly because of the possibility of using N-S Chart to write parts of itself! i.e. parts of N-S Chart could have been developed using N-S Chart generated code. This was attempted and proved quite feasible.

There is no reason why generators for other languages could not be added to the tool. Some 4GLs such as Ingress's OSL would be very useful.

10.6 Database Orientation

N-S Chart could be orientated more towards database manipulation diagramming. Block symbols could be used to represent begin-transaction end-transaction pairs. Process symbols could be extended, with perhaps variations being developed for Create, Read, Update and Delete operations.
11.0 Summary

The aim of this project was to overcome the limitations of 'pencil & paper' Nassi-Shneiderman diagrams by constructing a Nassi-Shneiderman diagrammer for the Macintosh. The Macintosh application N-S Chart was successfully developed with many features that improve greatly on tools previously available.

Modifications and additions to the Nassi-Shneiderman diagramming technique made it more appropriate for computer automation and integration with other diagramming techniques. The changes address the problems of data specification and system development tool integration. The additional goal of generating procedural program code from Nassi-Shneiderman diagrams was successfully reached.
12.0 References


[IMV*] Apple Computer, Inside Macintosh, Vols 1, 2, 3, 4 and 5. Addison Wesley 1987.


