1996

An Investigation Into an Effective Method of Automatically Analysing Oracle Applications to Count Function Points

J. L. Wong
Edith Cowan University

Edith Cowan University

Copyright Warning

AN INVESTIGATION INTO AN EFFECTIVE METHOD OF AUTOMATICALLY ANALYSING ORACLE APPLICATIONS TO COUNT FUNCTION POINTS

BY

J.L.Wong

A Thesis Submitted in Partial Fulfilment of the

Requirements for the Award of

Bachelor of Science (Computer Science) Honours

at the

Faculty of Science, Technology and Engineering

Edith Cowan University

Date of Submission: June 1996

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

Abstract

Function Point Analysis (FPA) is a synthetic software estimation metric used for computing the size and complexity of applications. It was first introduced by Allan.J.Albrecht during the mid-seventies, as a result of a lengthy research based on applications that were developed using COBOL and PL/1 programming languages.

The purpose of this research is to investigate the possibility, and the most effective method, of automatically performing a Function Point Analysis on Oracle applications that consist of Oracle Forms and Oracle Reports.

The research revealed a seemingly lack of other researches on this topic. As FPA was invented a few years prior to the birth of Oracle, and consequently that of fourth-generation languages, it had to be tailored to suit the fourth-generation language Oracle tools used to develop the Oracle applications. This experiment provided a proof of concept and resulted in a software that achieved its objective of automatically calculating Oracle applications, consisting of Oracle Forms and Oracle Reports, in an a posteriori manner.

I certify that this thesis does not incorporate, without acknowledgment, any material previously submitted for a degree or diploma in any institution of higher education and that, to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where due reference is made in the text.

Signature...

Date..... June 1996. .....

Acknowledgments

There are a number of people who have contributed to this research and to whom I am deeply grateful. In particular, I thank:

Dr Ken Mullin, my supervisor, for his valuable time in providing the continual guidance, encouragement, support, and feedback. His support has been invaluable.

Dr Thomas O'Neill, the Honours co-ordinator, for assisting with the formalities, getting the honours research rolling, and for being efficient thus reducing any unnecessary delays.

Assoc. Prof. Robert Cross for reviewing and approving the initial thesis proposal and for providing valuable feedback.

Oracle, the world's leading supplier of information management software, for expanding my knowledge in an extremely dynamic field and for providing research material.

My family for bringing me thus far and teaching me the values of life.

My fiance, Ronnie, for his love, motivation, support and for the inspiration to accept challenges.

God and the ever presence of the Holy Spirit.

And to all whom I have met along the way...

In Memory of

DAD

Table of Contents

CHAPTER 1: INTRODUCTION 7

Function Point Analysis 7

Research Objective 13

Problem Question 14

CHAPTER 2: REVIEW OF THE LITERATURE 15

FPA and Oracle 20

CHAPTER 3: METHODOLOGY OF RESEARCH 21

CHAPTER 4: FP COUNTING IN RELATION TO ORACLE APPLICATIONS 22

Steps to manually calculate Function Points in Oracle applications 22

CHAPTER 5: FP COUNTING IN RELATION TO ORACLE APPLICATIONS

- ISSUES 28

The Input / Inquiry distinction 28

Categorisation of LOVs 30

Internal Logical Files & External Interface Files 32

CHAPTER 6: ANALYSE THE STRUCTURE OF ORACLE FORMS & REPORTS 34

Forms 34

Reports 35

CHAPTER 7: LIST & EVALUATE POSSIBLE AUTOMATIC COUNTING METHODS 37

METHOD 1: Using Designer/2000 (CASE) 37

Advantages 38

Disadvantages 38

*METHOD 2: Using SQL*Plus* 39

Advantages 39

Disadvantages 40

METHOD 3: Parsing the text files of Oracle Forms & Oracle Reports 40

Advantages *41*

Disadvantages *41*

CHOICE of Method *41*

Determination of a Suitable Language for Parsing *42*

CHAPTER 8: DESIGN AN AUTOMATIC COUNTING METHOD

- ISSUES WITH IMPLEMENTATION 43

Software Requirement *43*

Large File Size *44*

Truncation of Generated Document *45*

CHAPTER 9: DESIGN AN AUTOMATIC COUNTING METHOD

- THE COMPLEXITY DETERMINATION PROCESS FOR ORACLE APPLICATIONS 46

External Inputs (EI) *46*

External Outputs (EO) *47*

External Inquiry (EQ) *48*

Internal Logical Files (ILF) *48*

External Interface Files (EIF) *49*

CHAPTER 10: DESIGN AN AUTOMATIC COUNTING METHOD 51

- THE PARSING PROCESS 51

Base Tables Referenced *51*

Base Table Columns Referenced *52*

Non- Base Tables and Columns Referenced *52*

Enhancements *55*

CHAPTER 11: THE DESIGN 56

CHAPTER 12: RESULTS & CONCLUSIONS 58

REFERENCES 60

APPENDIX A: FPA REPORTS GENERATED BY DESIGNER/2000 63

APPENDIX B: THE SQL SELECT STATEMENT 87

APPENDIX C: DESIGN OF FUNCTION POINT COUNTING SOFTWARE 88

APPENDIX D: A SAMPLE FPA REPORT PRODUCED 89

APPENDIX E: SOURCE CODE FOR THE FUNCTION POINT COUNTING SOFTWARE 90

Chapter 1: INTRODUCTION

This paper documents the outcome of research into the use of Function Point Analysis (FPA) to evaluate those applications that have been developed through the use of a group of tools manufactured by Oracle Corporation. This introduction consists of the theory behind Function Point Analysis (FPA) and its history of usage from the beginning to the present time. Following this account, a review of the current literature relating to the theory of FPA, and its practices, especially as related to Oracle applications is surveyed. The next section concentrates on the preparation of an effective method to automatically analyse Oracle applications using FPA. This will involve an analytical discussion of the possible methods for implementing the automated analyser and the issues relating to this implementation. Finally, the results of the implementation will be presented, and an appropriate conclusion will be drawn from these results.

Function Point Analysis

Function Point Analysis (FPA) is a synthetic software estimation metric used for computing the size and complexity of applications. It is a measure of the functionality of an application, as perceived by the user. Since its birth to the IT industry, FPA has been successfully adopted by a number of large organisations (Heemstra, 1991) who varied the metric slightly to better suit their environment. The popularity of FPA has continued to grow steadily since the seventies and has become the predominant estimation method used in the IT industry. There are now many large databases of completed projects and their function point counts. (Weaver, 1989)

FPA was first introduced by Allan.J.Albrecht (IBM, 1975) during the mid-seventies, as a result of lengthy research on applications that were developed using COBOL and PL/1 programming languages (Ferens, 1992). The inspiration for his research was to originate an alternative method to the traditional Source Lines of Code (SLOC) metric which was prevalent, but seemingly inadequate, at the time. FPA was to be used as a more substantial metric to estimate the cost and effort required to complete an application.

Using the original FPA metric, the estimate of the cost and effort required for software development was derived from a calculation of the number of function points associated with the application to be sized. This function point value was calculated based on two groups of parameters that were deemed from the user's perspective to be influential on the estimate:

1. The application attributes and
2. The environmental factors

The five attributes relating to the program to be estimated, which have been identified by Albrecht are:

- the number of external input types (EI)
- the number of external output types (EO)
- the number of external inquiry types (EQ)
- the number of internal logical files (ILF) and
- the number of external interface files (EIF).

This first equation assumes that all the attributes have an "average" rating and is computed as follows:

$$\mathbf{BFP = 4EI + 5EO + 10ILF + 7EIF + 4EQ}$$

where BFP is the "Basic Function Points". (Behrens as cited by Ferens)

To refine the estimation technique further, Albrecht proposed two areas of enhancements. Firstly, each item belonging to an attribute is classified as having "low", "average", or "high" complexity and then an appropriate weighting is assigned. For example, when examining the External Input component, the level of complexity would be determined using the following table:

DET (# columns) FTR (# tables)	1-4	5-15	16+
0-1	Low	Low	Average
2	Low	Average	High
3+	Average	High	High

Table 1

Note that the Data Element Type (DET) refers to the number of attributes/columns used by the module and the File Type Referenced (FTR) is "counted for each entity, table or file" used by the module. (Oracle, 1995)

The table shows that an EI module with five to fifteen data element types has a *low* level of complexity if it contains zero to one file type referenced, an *average* level of complexity if it contains two FTR, and *high* complexity if it contains more that three FTR. In relation to Oracle applications, the number of FTR maps to a count of the number of relevant tables, and the number of DETs maps to the associated table columns referenced. A detailed discussion is provided in the next chapter.

Using the complexity rating, the weighting for each of these application components can be obtained by referring to the following table (Ferens, 1992),

Attribute	Low	Average	High
Inputs (EI)	3	4	6
Outputs (EO)	4	5	7
Data Files (ILF)	7	10	15
Interfaces (EIF)	5	7	10
Inquiries (EQ)	3	4	6

Table 2

This results in the "Unadjusted Function Points" (UFP).

Once the environmental factors are applied to the UFP, the final adjusted function points (FP) will be obtained. The adjusted function points is obtained by summing up the ratings of the environmental factors, totalling to fourteen different characteristics, which results in a value known as the "total degree of influence" (TDI).

The fourteen characteristics comprising the group of environmental factors, with definitions provided by Dreger (1989, pp.63-4), are listed as follows:

1. Data Communications - "means that data or control information used in the application is sent or received over data communication facilities - including not only various networks, concentrators, multiplexers,and private lines, but also the terminals locally connected. On-line systems will always have at least some data communication influence."
2. Distributed Data/Processing - "indicates the application uses data stored, accessed, or processed on a storage or processing system other than the one used in the main program

routines. Note that presence of this factor increases the data communications influence previously defined."

3. Performance Objectives - "influence system design, development, implementation, and support when specific, user-approved demands for exceptionally high throughput or fast response times have been made."
4. Heavily-Used Configuration - "this factor is especially important to a user already lacking computer capacity but unable to purchase or acquire more hardware or upgraded software."
5. Transaction Rate - "a high transaction rate can occur when the network consists of many data entry or inquiry terminals, when each screen transmitted contains a lot of input information, or when the frequency of screen transmission is high."
6. On-Line Data Entry - "(including control and security functions) are always more difficult to accommodate than similar batch systems; hardware, application software, and operating system software are all affected by the additional requirements of an on-line system."
7. End-Use Efficiency - "human-factor features..designed to increase the level of "user-friendliness" and include such things as conventional data entry (requiring multiple sequenced screens), help screens, "next format" fields, paging capabilities, more descriptive documentation (including users manuals and "learner-friendly" training materials), second-language input/output screens and messages, and additional edit, error, and exception handling routines."
8. On-Line Update - "as are on-line inquiry and data entry more difficult than batch, so is on-line update of files and data sets more difficult because of the short turnaround time and its widespread effects on all system design components."

9. Complex Processing - "refers to the situation in which an application requires substantially greater than average difficult in input or output processing; in logic file, or numeric manipulation; or in exception handling routines."
10. Reusability - "refers to the situation in which some of an application's routines, subroutines, or other procedures have been designed or written with uses in mind other than just the program under evaluation."
11. Conversion/ Installation Ease - "increases the difficulty of application development but reduces the number and severity of problems in testing and implementation."
12. Operational Ease - "is not the same as end user efficiency." The purpose of this factor is "to provide effective but easy startup, backup, error recovery, and shutdown procedures, and to minimize such manual activities as mounting tapes or special forms, handling paper, or responding to requests for information at the operator console."
13. Multiple Site Use - "when the application has been specifically designed, developed, and supported for installation at multiple sites, for multiple organizations, additional co-ordination, review, and approval is required even if no site-unique code needs to be written."
14. Facilitate Change - "when the application has been specifically designed, developed, and supported to facilitate change, it requires increased attention to and planning for future maintenance and modification needs."

Based on the degree of influence that one expects from each of the characteristics, a rating of 0 (no influence) to 5 (highly influential), with an average influence rating of 3, is performed, preferably by the system user. Dreger (1989) suggests the inclusion only of those factors that:

"

- clearly benefit the user,

- are specifically approved by the user, and

influence to a measurable degree the design, development, implementation, or support of an application."

Once the ratings of the fourteen characteristics have been summed to derive the TDI, the following equation is used to compute the adjusted function points. Note that the TDI can alter the UFP by up to 35% in either direction.

$$\mathbf{FP = UFP * (0.65 + .01 * TDI)}$$

Function points were derived as a means of assessing the functionality of an application. Subsequent research (Ferens et al, 1992; Kansala & Kitchenham, 1993) showed the measure to correlate well with the effort required to develop the application, provided development environment and individual skills were similar. Thus FPs are a useful measure of effort as well as functionality.

Research Objective

The objective of this research is to investigate how FPA can be tailored to count the number of function points in given Oracle applications that have been developed using Oracle Forms and/or Oracle Reports. Once the investigation has been completed, and a clear plan has been devised, an application will be developed for the automatic calculation of function points for a given Oracle application. The resulting application will serve as a highly useful tool for its users. In particular, it

- Will eliminate "laborious hand counting of function points." (Internet: Funcnet)

- Will provide a consistent means of estimating the size of different Oracle applications. (Low, 1990)
- Will be independent of the technology that has been used for its development.
- Will enable lecturers to assess the effort that went into student Oracle projects.

Note that to satisfy the final point, the resulting application will be required to execute a function point analysis in an "a posteriori," that is, after the system development phase. (Hignite, Johnson, Foster, 1993).

Upon completion, this will be one of the few pieces of research that focuses on the usage of FPA to automatically count the number of function points in an *Oracle* application.

Problem Question

The research question pertaining to this project is as follows:

> **What is a most effective way of automatically counting the function points in an Oracle application consisting of forms and reports?**

Chapter 2: REVIEW OF THE LITERATURE

Prior to the invention of Function Point Analysis (FPA), the primary software estimation metric used was the Source Lines of Code (SLOC) metric. The major drawback with this metric is that it does not measure software productivity, which the standard economic definition describes as, "Goods or services produced per unit of labor and expense." Relating to this economic definition, the SLOC metric fails to measure software productivity due to the following reasons (Jones, p.45):

1) Lines of code are neither 'goods' nor 'services'. Thus, measuring the lines of code does not provide a good measurement of software productivity.
2) Lines of code are not the primary deliverable for customers. Customers are not concerned with the number of lines that comprise a completed piece of software, nor are they interested in the programming language used for the source code. In fact, if a piece of software could be developed in a higher-level language, thus generating less code in the final product to provide cost reduction benefits, it would serve as a preferred option, from the customer's point of view.

The deficiency in the SLOC metric inspired the emergence of the Function Point metric. The function point computation is based on those components deemed as important, or of interest, to the customers, and qualify as a quantifying characteristic of the term 'goods' that exists in the economic definition of productivity.

Since its emergence, a number of research projects have been conducted on FPA, focussing on the comparison to SLOC, with favourable outcomes. As an example, research performed by

Kremer (1987), revealed that the two function point models used, ESTIMACS and an Albrecht-derived model, produced estimates that were much more accurate than the two SLOC-based models, PRICE-S and SLIM. The research was based on the comparison of the estimated figures with the actual effort of fifteen, mostly COBOL, applications.

This observation is supported by similar research, conducted by Low & Jeffery (1990), on COBOL and PL/1 business programs which indicated that the function point metric was a more consistent size estimator than the SLOC metric. The function points counted correlated with the effort. This makes FPA a good estimating tool. (Ferens, 1992)

Although many authors, such as (Yau, 1995) , (Tsoi, 1995), and (Heemstra, 1991), agree that FPA is widely used and is also a successful method (Betteridge, 1993) for software estimation, a number of significant issues and possible areas of improvement also exist and should be addressed. These areas include:

1) The need for an easier method of defining and counting the application components. One of the major obstacles associated with counting the number of function points within a program is the identification and calculation of the number of inputs, outputs, data files, inquiries, and interface files. A variation of the original Albrecht's FPA is *Mark II Function Points* which was developed by Symons (Symons, 1988). It attempts to simplify the original method of FPA by using only three of the program attributes, namely the inputs, the outputs, and the entity references of each logical transaction (Betteridge, 1992). Ferens explains that the absent attributes can be neglected since the "external interfaces and inquiries are treated as inputs or outputs, and internal files are 'replaced' by a measure of entity types referenced by transactions."

2) A need to ascertain the accuracy of estimation models using function points. Rask, Laamanen, and Lyytinen (1993, p.661) stated that "the quality of a cost estimate is a function of how it compares with the actual result." The observation made by Ferens (p.635) on Kremer's study involving the application of the function point models ESTIMACS and an Albrecht-derived model on fifteen (mostly COBOL) programs revealed that "for even the most accurate model, ESTIMACS, the estimates averaged 85% higher than actual levels of effort." However, Ferens' own study of applying three FP models, the SPANS (Tecelote Software Program Acquisition Network Simulation) model by Tecelote Research Inc., the Checkpoint model by Software Productivity Research, and the Costar model by Softstar Systems, to estimate thirty-six (mostly COBOL) business programs appeared to indicate that the "calibration of models, or adjusting models to a particular environment, appears to be a worthwhile endeavour if greater accuracy is sought." This is confirmed by Betteridge's (1992) study which compared the results derived from an FPA method, with the managers' estimates and the actual expenditure. Betteridge (1992) concluded that "the results give some cause for optimism in the use of the function point model that was used (Mark II)."

3) FPA requires an assessor, commonly the main user(s), to rate a set of 14 general system characteristics. These characteristics, including *Reusability, Facilitate Change, Performance,* are all subjective elements. Even though these subjective elements are used in FPA, the result given does not show the statistical confidence interval, that is the assessor's confidence level of the general system characteristics being rated subjectively. To overcome this inability to assess the confidence level of the estimate, Tsoi & Yau (1995) introduced fuzzy logic to the FPA model, that is, a "fuzzified FPA" (FFPA).

The FFPA method, proposed by Tsoi et al (1995), is based on the traditional FPA. The contrast lies in the determination of the Technical Complexity Factor which is derived from an evaluation of the ratings given to the fourteen general system characteristics.

The fourteen general system characteristics are rated not only on a score of 0 (no influence) to 5 (Highly influential), as exists in the traditional FPA, but also on the assessor's linguistic degree of certainty rating of *Almost Certain, Very Likely, Probably, Unlikely,* and *Extreme Unlikely.* For example, an FFPA assessor may give an estimate of "Very Likely 3" to the *Performance* system characteristic and "Unlikely 1" to the *Reusability* system characteristic.

For each of the estimates given by an FFPA assessor, an Fuzzified score for General System Characteristics (GSC) can be obtained by referring to the following table (Tsoi et al, 1995).

	Score 0	**Score 1**	**Score 2**	**Score 3**	**Score 4**	**Score 5**
Very Likely	from 0 to 0.25	from 0.75 to 1.25	from 1.75 to 2.25	from 2.75 to 3.25	from 3.75 to 4.25	from 4.75 to 5
Probably	from 0 to 0.5	from 0.5 to 1.5	from 1.5 to 2.5	from 2.5 to 3.5	from 3.5 to 4.5	from 4.5 to 5
Unlikely	from 0.75 to 5	from 0 to 1.75; from 1.75 to 5	from 0 to 1.2; from 3.75 to 5	from 0 to 2.25; from 3.75 to 5	from 0 to 3.25; from 4.75 to 5	from 0 to 4.25;
Extreme Unlikely	from 1 to 5	from 2; to 5	from 0 to 1; from 3 to 5	from 0 to 2; from 4 to 5	from 0 to 3;	from 0 to 4

Table 3 : Fuzzified Score for GSC

The final Fuzzified FPA count will consist of a range of values, to reflect the confidence level. The following is an example give by Tsoi et al (1995) to illustrate the calculation performed using FFPA to derive the FP count:

TCF (FPA) = 0.65 + (0.01 * 48) = 1.13
TCF_{min}(Fuzzified FPA) = 0.65 + (0.01 * 44) = 1.09
TCF_{max}(Fuzzified FPA) = 0.65 + (0.01 * 51.5) = 1.165
Function Point Computed (FP) :
FP (FPA) = 211 * 1.13 = 238.43
FP_{min}(Fuzzified FPA) = 211 * 1.09 = 230
FP_{max}(Fuzzified FPA) = 211 * 1.165 = 245.815

As Tsoi et al (1995) explains, "it has been expected that the FPA result falls in the range of the FFPA result, from 230 to 245.815. There is around 8% difference of DI (Degree of Influence) between the two models."

Tsoi et al (1995) concludes that the estimates provided by this Fuzzified version of FPA "have been found more informative than the conventional FPA" and that "the range of estimates allows the project management to conduct contingency planning more effectively."

4) Estimation of fourth generation (4G) applications. Since FPA was invented prior to the existence of 4G languages, there may be possible areas of improvement to accommodate for 4G applications. An investigation carried out by Van Wonderen (1991) revealed that "improvements are necessary, particularly for the estimation of interactive 4th-generation language applications." This issue is particularly relevant to this research, as the applications to be automatically function point counted are developed using Oracle Forms and/or Oracle Reports which are considered to be 4G development tools. The issues relating to the usage of FPA to 4GLs and Oracle applications will be covered in a later chapter.

5) FPA is not readily adaptable to real-time, scientific environments. Jones (p.76), as cited by (Alford, 1991), explains that FPA "is not widely used for real time systems, military systems, or any other kind of software where algorithmic complexity is high and data complexity is low." Inspired by this, Jones proposed an adaptation of function points, known as *Feature Points*, to allow for the real-time environment. Feature Points uses the

five attributes proposed by Albrecht. The differences between the two metric concepts lie in the different weightings assigned to the internal file attribute, and also in the new attribute, *algorithmic complexity* (A), introduced in Feature Points. The new equation for the *Basic Feature Points* (BFEA) is

$$\mathbf{BFEA = 4EI + 5EO + \underline{7ILF} + 7EIF + 4EQ + \underline{3A}}$$

compared to $\mathbf{BFP = 4EI + 5EO + 10ILF + 7EIF + 4EQ}$ (as shown previously)

Note: 7 + 3 = 10 ie. Points of algorithmic complexity weighted same as ILFs.

FPA and Oracle

The literature research to date has not revealed any studies on the use of FPA models on Oracle applications. To achieve the objective of this project, it would be necessary to investigate how the definitions of the function point parameters, and the function point counting rules apply to Oracle applications. Once this has been established, it would be a natural progression to automate the function point computation for Oracle applications.

Chapter 3: METHODOLOGY OF RESEARCH

The research into a most effective way of automatically counting the number of function points in any given Oracle application followed this method:

1. Investigate how FP counting can be applied to Oracle applications, including whether this has been achieved elsewhere.
2. Analyse the structure of Oracle forms & reports to determine how the application components can be counted.
3. Determine the best automated method to do this counting.
4. Design an automatic method of analysing this structure to count inputs, outputs, inquiries, data files, and interfaces.
5. Set up the development environment.
6. Develop the software.

The remainder of the thesis follows this methodology.

Chapter 4: FP COUNTING IN RELATION TO ORACLE APPLICATIONS

An extensive literature search, with sources ranging from libraries to the Internet World-Wide Web, revealed a deficiency in previous researches, let alone the production of software, on the automatic counting of function points in Oracle applications. In fact, the only enlightening literature discovered that related to this subject was from an Oracle manual, titled *QMS Project Management*. This is a Quality Management Systems manual produced for project managers intending to develop quality systems. The manual contains a chapter on estimating projects, which contains a section on FPA. The automatic function point counting software resulting from this investigation was developed based mainly on the function point theory presented in this manual. This theory closely follows the IFPUG standard.

Steps to manually calculate Function Points in Oracle applications.

STEP 1. Generate a full text description of the Oracle Forms or Oracle Reports application - the *Module Documentation.*

FOR ORACLE FORMS APPLICATIONS:

a. Start up Oracle Forms Designer 4.5

b. *File* | *Open* then specify the name of the application eg. emp.fmb

c. *File* | *Administration* | *Form Doc*

FOR ORACLE REPORTS APPLICATIONS:

a. Start up Oracle Reports Designer 4.5

b. *File* | *Open* then specify the name of the application eg. dept.rpt

c. *File* | *Administration* | *Report Doc*

The text version, eg. emp.txt & dept.txt, should now be generated. This text file is then parsed for the application attributes (steps 2-5).

STEP 2. For each form or report module, count the number of base tables referenced.

Oracle (1994) defines a base table as one that is "associated with a specific database table or view." Base tables are associated with base table blocks within Oracle Forms.

STEP 3. For each form or report module, count the number of base table columns referenced.

Oracle (1994) explains that the base table columns "correspond directly to columns in the block's base table." They should correspond to a base table elected in the previous step.

STEP 4. Count the number of accumulated non-base tables referenced in the application.

Non-base tables are commonly referenced in *select* statements, such as those belonging to a record group (forms) or belonging to queries (reports). A SQL *select* statement may contain references to more than one table. Caution must be exercised in counting the number of tables referenced as, for example, *select* statements can be nested within each other.

STEP 5. Count the number of accumulated non-base columns referenced in the application.

This is similar to the count of the number of non-base tables referenced, as detected in the previous step. This involves a count of the number of associated columns referenced and can be an intricate process. Consideration must be made for such instances as nested calls to built-in functions. For example, *select nvl(round(max(salary), 2), 0) from...*

STEP 6. Determine the complexity rating for each component. (Hignite et al, 1993)

External Input: For each non-query-only form module, determine the complexity rating by applying the number of base tables and their columns, derived in Step 2 and Step 3, to the following table (Oracle, 1995)

# base table columns # base tables	1-4	5 - 15	16+
0 - 1	Low	Low	Average
2	Low	Average	High
3+	Average	High	High

Table 4

External Output: For each report module, determine the complexity rating by applying the number of base tables and their columns, derived in Step 2 and Step 3, to the following table (Oracle, 1995)

# base table columns # base tables	1-5	6 - 19	20+
0 - 1	Low	Low	Average
2 -3	Low	Average	High
4+	Average	High	High

Table 5

External Inquiry: For each query-only form module, determine the complexity rating by applying the number of base tables and their columns, derived in Step 2 and Step 3, to the following table (Oracle, 1995)

# base table columns # base tables	1-4	5 - 15	16+
0 - 1	Low	Low	Average
2	Low	Average	High
3+	Average	High	High

Table 6

Internal Logical Files: An internal entity/table is one that is maintained by the application through creation/deletion/update. Determine the complexity rating by applying the number of tables and their columns, derived in Step 4 and Step 5, to the following table (Oracle, 1995)

# columns referenced # tables referenced	1-19	20-50	51+
1	Low	Low	Average
2-5	Low	Average	High
6+	Average	High	High

Table 7

External Interface Files: An external entity/table is one that is used by the application through retrieval. Determine the complexity rating by applying the number of tables and their columns, derived in Step 4 and Step 5, to the following table (Oracle, 1995)

# columns referenced # tables referenced	1-19	20-50	51+
1	Low	Low	Average
2-5	Low	Average	High
6+	Average	High	High

Table 8

Step 7. Determine the total number of Unadjusted function points.

The complexity ratings derived from the above steps are then converted into function points by applying the ratings to this table. (IFPUG, 1990)

	Low	Average	High
External Input (EI)	3	4	6
External Output (EO)	4	5	7
External Inquiry (EQ)	3	4	6
Internal Logical File (ILF)	7	10	15
External Interface Files (EIF)	5	7	10

Table 9

Step 8. Calculate the Total Degree of Influence (TDI).

The TDI calculation is based on the summation of the fourteen general system characteristics, commonly elected by the system users. A rating (Oracle, 1995) of

0 Not present

1 Incidental influence

2 Moderate influence

3 Average influence

4 Significant influence

5 Strong influence throughout

is applied to each of the fourteen characteristics. The fourteen characteristics that relate to the general functionality of the application is as follows:

1. Data Communications
2. Distributed Data/Processing
3. Performance Objectives
4. Heavily-Used Configuration
5. Transaction Rate
6. On-Line Data Entry
7. End-Use Efficiency
8. On-Line Update

9. Complex Processing

10. Reusability

11. Conversion/ Installation Ease

12. Operational Ease

13. Multiple Site Use

14. Facilitate Change

Step 9. Calculate the Technical Complexity Factor (TCF).

Use the following formula (Hignite et al, 1993) to compute the TCF:

TCF = (TDI x 0.01) + 0.65

Step 10. Calculate the Total Function Points (TFP).

Finally, the total FP count can be derived by applying the following formula (Hignite et al):

TFP = TCF x Unadjusted function points

Once a method of manually counting the number of function points for any forms/reports-based Oracle application was identified, the next challenge was to automate this process. An evaluation of these methods are discussed in the next section.

Chapter 5: FP COUNTING IN RELATION TO ORACLE APPLICATIONS

- ISSUES

Since FPA was invented prior to the existence of 4G languages, there are a number of issues relating to the use of FPA to estimate Oracle applications that have been developed using Oracle Forms and/or Oracle Reports which are considered to be 4G development tools. These include

- The distinction between an input screen and an inquiry screen
- Categorisation of the List of Values (LOV) feature
- The determination of Internal Logical Files and External Interface Files.

The Input / Inquiry distinction

The IFPUG definition, as provided by Oracle (1995), states that an external input is one that "processes data or control information which enters the application's external boundary." When applied to 4GL applications, specifically to those developed using Oracle Forms, an external input could be referred to a screen developed using Oracle Forms, since a screen allows the input of data. One of the advantages of using Oracle Forms to develop screens for user inputs is that, by default, the data inquiry facilities are also provided by the input screen. This is where the complication of applying FPA to Oracle applications arise. How does one distinguish between an external input and an external inquiry in Oracle applications?

While FPA draws a distinction between external inputs and external inquiries, this is not necessary for Oracle Forms applications since both the input and inquiry features are typically

included in the same screen. To cater for these differences when using FPA to estimate Oracle applications, one can categorise an input/query screen as either:

- an external input only
- an external inquiry only or
- both an external input and an external inquiry

The preferred option to be used is entirely based on the individual estimator's preferences. The automatic calculation of function points in Oracle applications prototype software developed in conjunction with this documentation defines an input/query screen as an external input only. The explanation for this follows.

Although a screen developed using Oracle Forms allows both input and inquiry features by default, these features can also be toggled to be enabled or disabled. Thus, a screen can be either:

- an input only screen
- an inquiry only screen
- an input/inquiry screen or
- a non-input/non-inquiry screen.

To distinguish between an external input and an external inquiry, an Oracle Forms screen is only deemed to be an external inquiry if it is a query-only form. Based on this logical definition, the above selection of screens is categorised as follows:

Screen Type	Classification
Input only screen	External input
Inquiry only screen	External inquiry
Input / inquiry screen	External input
Non-input / non-inquiry screen	neither.

Table 10

When parsing the Oracle Forms text file, the automatic parser should search for the

Insert Allowed True/False
Query Allowed True/False

properties listed under the block(s) associated with the input/inquiry screen to determine the input/inquiry status. This will allow the classification of the screen as an external input or an external inquiry.

Categorisation of LOVs

A screen developed using Oracle Forms may contain one or more instances of a List of Values (LOV) to facilitate the ease of input. These are commonly known as "look-up tables." A LOV may be based on a record group which may query one or more database tables. An example of the use of an LOV is the entry of a postcode value belonging to an address section of a personal details screen. Rather than relying on the user to remember the postcode values for all suburbs, the postcode field may be implemented to use a LOV which queries the postcode database table to return a list of all of the suburbs and their associated postcodes. Once a suburb and its associated postcode is selected, the postcode field will be populated with the selected value.

If a LOV is based on a record group that queries one or more database tables, it should be classified as an external inquiry. This is a sensible assumption as a user is likely to perceive this LOV as a query.

When parsing the Module Documentation of the Oracle Forms/Reports, the automatic parser should search for the LOV property to ensure that an LOV is attached to a text item. This property would have a value of <null> if an LOV was not attached to it. An example illustrating an item with the *postcode_lov* attached is:

LOV	postcode_lov

Once it has been established that an LOV is attached to a text item, the next step would be to ensure that the LOV attached is based on a record group. To do this, the parser should search for the

LOV Type	Record Group
Record Group	Postcode_query

properties, under the LOV section. Once this has been established, the parser can search for the *Record Group Query* property under the record group section to obtain the query statement used for this record group. For example,

Record groups	
Name	Postcode_query
...	
Record Group Query	select distinct code, suburb from postcodes

These steps will allow the estimator to determine whether an LOV is based on a query of one or more database tables and if so, the database tables and columns that are used. This information will allow the estimator to determine the complexity rating for the external inquiry.

Internal Logical Files & External Interface Files

Oracle (1995) describes an internal logical file as an "entity which is maintained by the application, in other words: the CRUD matrix contains at least one C, U, or D for this entity", and an external interface file is defined as "an external entity .. with an R in the CRUD matrix." The CRUD matrix refers to the Create, Retrieve, Update, and Delete functionality. Based on these definitions, it may be worthwhile to note that the internal logical files (ILF) and the external interface files (EIF) are applied to the entire application. Therefore, the complexity rating should be applied to the accumulation of these entities for the entire application rather than for each separate module.

The automatic calculation of function points in Oracle applications prototype software developed in conjunction with this documentation defines the base tables as external entities. A base table is typically associated with a block within a screen.

As for the internal entities, the software parses the application text file for keywords: *create*, *update*, and *delete* to determine the existence of internal entities. The existence of such entities are rare in typical Oracle applications.

The accumulated count of internal and external entities are used to determine the complexity ratings for the ILF and the EIF.

In conclusion, it is important to note that although the customisations made by an estimator for the application of FPA to a 4GL application, such as Oracle, is crucial, it may not be as significant as the consistent usage of the same method for all of the applications to be estimated.

Chapter 6: ANALYSE THE STRUCTURE OF ORACLE FORMS & REPORTS

An Oracle application typically consists of a number of input screens, to allow user interaction with the data within database tables, and the facility to generate reports through the retrieval of data from the database tables. For example, an order entry system application may consist of order input/inquiry screens and order reporting facilities. Each of the components of an Oracle application are discussed in detail to provide a general overview of the concepts behind forms, tables and reports.

Forms

An input/inquiry screen within an Oracle application is typically designed using the Oracle Forms Designer development tool. When using Oracle Forms Designer to create an input screen, an inquiry facility is provided "free" to the same screen without additional effort required to update the form design.

A form is a logical collection of blocks, items, triggers and procedures. A block is a logical collection of fields in a form. It may correspond to, at the most, one table. A database table on which a block is based upon is known as a base table. Those items that are based upon these base tables are known as a base table items.

An item field is an area that is capable of accepting and displaying data. To facilitate the entry of data, an item field can appear in one of a number of different forms, including list items,

radio buttons, checkboxes, text items that allow data inputs, and display items that do not allow data inputs. The data displayed can correspond to a column in a database table.

A List of Values (LOV) is another way of assisting users to enter data in an item field. An LOV is a look-up table that consists of a query to a database table. An LOV may be associated with a text item. An example of the use of an LOV is for entering the customer code within an Invoice screen. Instead of relying on the user to memorise the customer codes for all existing customers, an input/inquiry screen may attach an LOV to the customer code item. This LOV may be based on a query that retrieves the customer codes and names for every customer in the database table.

Triggers and procedures within forms contain programming logic that may include read/write to database tables. The logic within these components can be written in PL/SQL. (Oracle, 1986)

Reports

Oracle Reports Designer is typically used for the reporting components of Oracle applications. To specify the data definitions within Oracle Reports Designer, a data model is created. A data model consists of the following data definition objects: queries, groups, columns, parameters and links.

Report queries consists of SQL SELECT statements written to fetch data from database table(s). An analysis of the report queries will reveal the tables and columns that have been referenced by a report. Once a query has been specified, groups and columns will be created

to reflect the query. Groups contain columns and are used primarily to create breaks in a report.

Links are used to specify parent-child relationships between one SELECT statement and one or more other SELECT statements.

Parameters are variables to which a user can assign values at runtime. The two types of report parameters are system parameters and user-defined parameters. It is the user-defined parameters that are relevant to the counting of function points since they may contain SQL SELECT statements to fetch data from database tables. (Oracle, 1988)

Chapter 7: LIST & EVALUATE POSSIBLE AUTOMATIC COUNTING METHODS

There is more than one way of implementing the automatic counting of the number of function points within an Oracle application. This section will highlight three of the more likely methods of achieving this and will provide a logical evaluation of these methods. These methods are:

- Using Designer/2000 (Oracle's CASE tool)
- Using SQL*Plus
- Parsing the Module Documentation.

METHOD 1: Using Designer/2000 (CASE)

Designer/2000, the most recent version of the Oracle CASE software, is capable of generating a number of reports, based on the parameters given. A group of these reports are based on Function Point Analyses. A sample of the printouts of these FPA reports is provided in Appendix A, and they are listed as follows:

MkI FPA Analysis Level	- CDFPA1A
MkI FPA Design Level	- CDFPA1D
FPA Analysis Level (DFDs & Event Models)	- CDFPAA2
FPA MkII (Design1)	- CDFPAD1
FPA MkII (Design2)	- CDFPAD2
Area Metric	- CDMETRIC

Advantages

These reports provide a detailed analysis of the application system requested by the user, for the purpose of function point analysis. The technique of counting the number of function points is controlled by Designer/2000 and is stored internally. This automatic computation of function points saves the user time which would have been expended on grasping the workings of Function Point Analysis and also on manually counting the function points for each individual module to be estimated. Since the same method is automatically applied by Designer/2000 to compute the number of function points in any given Oracle module, the results obtained are expected to be consistent.

Disadvantages

It appears that to take advantage of this utility, the Oracle application to be analysed must be designed and generated by Designer/2000, and stored in the database. This may pose an unnecessary obstacle if the applications were developed as a direct usage of Developer/2000, or more specifically, Oracle Forms Designer and Oracle Reports Designer. In this case, however, the Reverse Engineering utility provided by Designer/2000 may serve as a viable option.

The Reverse Engineering utility attempts to capture the data/functional design of an Oracle application in the CASE tool. When reverse engineering a forms module using *Module Data Diagrammer*, the *blocks* within the forms are translated to entities, the *items* to attributes, and the *relationships* (defined through the presence of the foreign key constraints) are translated to the relationships between entities. This process will result in a data diagram displaying the entities, their attributes, and the relationships between entities. In addition to this, the

properties of each of the elements within this data diagram provide further information that are relevant to the element highlighted. For example, the properties of an attribute contain information including whether it is updateable, and whether it is queryable. The data diagram produced will assist in function point counting.

The main obstacle with the use of Designer/2000 to generate FPA reports is the requirement of the Designer/2000 software, which in turn, demands an increase in the hardware requirements. It cannot be done easily, or definitively.

Conclusion: viable but difficult.

*METHOD 2: Using SQL*Plus*

Prior to the development of an Oracle application, the usual practice asserts the creation and population of tables in the database first. This is normally achieved through the execution of a Data Definition Language SQL script. By parsing this SQL script, or querying the database after the creation of the tables, one would be able to retrieve such information as the number of tables and columns that exist in the database. For example, the script

```
select table_name from user_tables;
```

would list all of the tables that exist in this database.

Advantages

This appears to be a simple method of collecting information, such as a count of the number of tables and columns, to assist the performance of a FPA. The simplicity is partly due to the

ease of data collection using SQL, and also partly due to the sole requirement of the standard SQL*Plus product which is a common product for Oracle developers.

Disadvantages

However, upon further examination, one should be convinced that this method provides insufficient data.

Firstly, the mere creation or existence of a table in the database does not guarantee its usage by the application to be function point analysed. A table in the database may not be referenced by the application to be analysed at all. There appears to be a lack of an easy solution to differentiating between those tables that are, and those that are not, relevant to the function point analysis of an application.

Furthermore, one cannot distinguish whether a table in the database that is referenced by an application is referenced as a base table for a block or referenced by a radio group in a *select* statement.

Conclusion: not viable.

METHOD 3: Parsing the text files of Oracle Forms & Oracle Reports

The conversion of an Oracle Forms or Oracle Reports binary file to its text equivalent, Module Documentation, is a simple process that can be achieved by following the instructions provided in the first step of the previous chapter.

Advantages

The Module Documentation (MD) covers detailed information regarding the forms/reports module. Relating to FPA, the MD incorporates all of the necessary information to perform a function point calculation. This information includes the base tables and their columns that are referenced by the module, trigger texts, and SQL code for record groups and for other components. By parsing this text file, a function point count can be achieved automatically.

Disadvantages

The issues relating to the method are:

- Software requirement
- Large text file size
- Truncation of the MD

A detailed discussion is provided in a later chapter.

Conclusion: Viable and do-able.

CHOICE of Method

It appears that the third method, parsing of the Modular Documentation, is the most suitable method to use for automatically computing the number of function points in Oracle applications.

Determination of a Suitable Language for Parsing

An optimal parser for these Oracle-generated text files should be able to deliver the following characteristics:

- Backtracking: The parser should be able to scan in both directions, that is, forwards and backwards. An example of the use of backtracking is to get the name of a base table column. To do this, the parser must firstly search for an item with the "base table item" property set to true. Once this is found, the parser will be required to reverse its direction to resolve the name of the item by searching for the "name" property.
- Data Structure: One of the more significant data structures that the parser should possess is the array structure. The parser should be able to keep track of the accumulated number of base tables and base table columns detected for each and every form and report text file parsed, and also keep track of the complexity ratings for each of the form/report module. Reporting facility: At the end of the parsing phase, the parser should be able to produce a report that presents the results in a clear, logical, and presentable form to the analyst.
- Availability: Ideally, the parser end-product should be an automatic estimator that is easily attainable by analysts. An Oracle analyst should be able to access the automatic estimator and execute the parser straight away, reducing the unnecessary wasted time on installations, compatibility checks and other pre-installation procedures.

For this research, Microsoft Word Basic has been chosen as the prototype language to implement the parser for the automatic calculation of function points in Oracle applications. The main objective of the prototype is to provide a "proof of concept" for the ideas presented in this document. The use of Microsoft Word Basic achieves this and also meets the above requirements for an effective parser for this research.

CHAPTER 8: DESIGN AN AUTOMATIC COUNTING METHOD

- ISSUES WITH IMPLEMENTATION

By using the third method discussed in the previous section, the automatic function point counting software could be implemented successfully. However, there are a number of issues that should be considered. A comprehensive discussion of these issues is given in this section.

Software Requirement

A basic requirement of the implementation of the method under discussion is the following software: Oracle Forms Designer, Oracle Reports Designer, and Microsoft Word. The first two application tools are required for the automatic generation of the module text documents, and Microsoft Word is required to view the generated text documents, to parse the text files, and to generate a report of the results of the automatic function point analysis.

These three pieces software are all within reasonable expectations. If an application has been developed using the Oracle Forms Designer and Oracle Reports Designer development tools then these tools may still be available at the time of function point analysis. As for Microsoft Word, this software was deliberately chosen to perform the analysis work, due to its popularity with personal computer users.

Large File Size

An important consideration when generating a text document of a binary Oracle Forms or Reports file is that the generated document can be relatively large. For example, generating a simple binary forms file of size 32KBytes can result in a text document of ten times its original size, in this case 231KBytes, which equates to approximately 78pages when viewed using Microsoft Word size 10 font.

The automatic function point computation software developed appears to parse the large text files within a reasonable amount of time. For example, on a 486DX2-66MHz laptop with eight megabytes of RAM, a very large forms text file opened in Microsoft Word size 10 font, spanning 263 pages, consisting of 14 425 lines and 43, 213 words, was parsed by the software in approximately two-and-a-half minutes.

Since an Oracle application will consist of many form and report modules to be parsed separately, the total time taken to automatically count the number of function points may become quite significant. This potential problem was conceived at the design phase of the software development and a method was incorporated into the software in an attempt to alleviate this symptom. The software prompts the user for the names of all of the application module text files, stores these file names in an array, and then parses all of the modules together. This way, the user is not required to be present to continually feed the next module into the parser.

The large text files resulting from the document generation facility in Oracle Designer appears to be unavoidable. Consequently, the time taken to parse these text files will inevitably be

lengthened. To alleviate this problem, one can only alleviate the symptoms. By incorporating the method mentioned, the time requirement on the user's behalf is reduced.

Truncation of Generated Document

Another disadvantage to be highlighted is the occasional truncation of those lines of code, mainly those within trigger texts, within a module that exceed their limitation. This may result when using the document generation facility provided by Oracle Forms Designer and Oracle Reports Designer.

One way of avoiding the truncation of trigger texts would be to generate a *.fmt* extension of the text file instead of generating a *.txt* extension. This facility is also provided in the development tools, however, this format of the text file does not include the other information, such as that relating to base tables, which is required to perform a function point count.

The prototype function point counting software is developed based on the assumption that the occurrence of the right-truncation of lines, for any given Oracle application, will not be frequent enough to produce a significant variation to the final function point count achieved.

Chapter 9: DESIGN AN AUTOMATIC COUNTING METHOD

- THE COMPLEXITY DETERMINATION PROCESS FOR ORACLE APPLICATIONS

The derivation of the Unadjusted Function Point count depends on the complexity rating of the five components: External Interface File, Internal Logical File, External Input, External Output, External Inquiry. A discussion of the accommodation of the IFPUG definitions and method for an Oracle application is provided in this section. This is to assist the implementation of the automatic counting software for Oracle applications.

External Inputs (EI)

The IFPUG definition of an External Input is defined as one that "processes data or control information which enters the application's external boundary." When tailored to Oracle applications, an EI becomes "a .. module of which the CRUD (Create, Retrieve, Update, Delete) usage contains a C, U, D." (Oracle, 1995) An obvious example of an EI is an Oracle Form module that is not query-only.

The IFPUG complexity rating of an EI is dependent on the number of File Types Referenced (FTR) and the number of Data Element Types (DET).

An IFPUG version of a FTR is "counted for each Internal Logical File maintained or referenced and each External Interface file referenced during the processing of the External Input." This can be tailored to Oracle applications to be defined as one that "is counted for

each ..table ..used by the .. module." and the DET is the number of associated attributes. (Oracle, 1995).

# base table columns # base tables	1-4	5 - 15	16+
0 - 1	Low	Low	Average
2	Low	Average	High
3+	Average	High	High

Table 11

External Outputs (EO)

The IFPUG definition of an External Output is defined as one that "processes data or control information that exits the application's external boundary." When tailored to Oracle applications, an EO becomes "a .. module of which the CRUD (Create, Retrieve, Update, Delete) usage contains only R's." (Oracle, 1995) An obvious example of an EO is an Oracle Report.

The complexity rating of an EO is dependent on the number of File Type Referenced (FTR) and the number of Data Element Types (DET), both of which are explained in the *External Inputs* section.

# base table columns # base tables	1-5	6 - 19	20+
0 - 1	Low	Low	Average
2 -3	Low	Average	High
4+	Average	High	High

Table 12

External Inquiry (EQ)

Oracle (1995) defines an External Inquiry as one that "requires input parameters, usually a unique identifier, and produces output with a fixed volume, usually with a fixed volume, usually one record." Following the concept presented by Oracle (1995), the implementation of the automatic function point counting software classifies query-only forms as external inquiries, instead of external outputs, since it "is usually not explicitly specified by the user." The complexity rating of an EQ is dependent on the number of File Type Referenced (FTR) and the number of Data Element Types (DET), both of which are explained in the *External Inputs* section.

# base table columns # base tables	1-4	5 - 15	16+
0 - 1	Low	Low	Average
2	Low	Average	High
3+	Average	High	High

Table 13

Internal Logical Files (ILF)

A formal definition of the Internal Logical Files, provided by Engineering Information, Inc (1996), is "a user identifiable group of logically related data or control information maintained within the boundary of the application being counted." In addition to this definition, the counting rules in relation to ILF are also provided. These rules specify that the following rules "must apply for the group of data/control information to be counted as an ILF:

It is a logical, or user identifiable, group of data that fulfils specific user requirements.

It is maintained within the application boundary.

It is modified, or maintained, through an elementary process of the application.

It has not been counted as an EIF for this application." (Engineering Information, 1996)

When related to an Oracle applications, an internal entity, or table, is one that"is maintained by the application, in other words: the CRUD matrix contains at least one C, U or D for this entity (or table)."

The complexity rating of an ILF is dependent on the number of Record Types (RET) and the number of Data Element Types (DET). Oracle (1995) explains that "an entity or a table can have only one record definition: RET=1" and that the "DET is the number of attributes."

# columns referenced # tables referenced	1-19	20-50	51+
1	Low	Low	Average
2-5	Low	Average	High
6+	Average	High	High

Table 14

External Interface Files (EIF)

A formal definition of the External Interface Files, provided by Engineering Information, Inc (1996), is "a user identifiable group of logically related data or control information referenced by the application being counted, but maintained within the boundary of another application." In addition to this definition, the counting rules in relation to EIF are also provided. These rules specify that the following rules "must apply for the group of data/control information to be counted as an EIF:

It is a logical, or user identifiable, group of data that fulfils specific user requirements.

It is referenced by, and external to, the application being counted.

It is not maintained by the application being counted." (Engineering Information, 1996)

When related to an Oracle applications, an external entity, or table, is one that only appears with a Retrieved in the CRUD matrix.

The complexity rating of an EIF is dependent on the number of Record Types (RET) and the number of Data Element Types (DET), both of which are explained in the *Internal Logical File* section.

# columns referenced # tables referenced	1-19	20-50	51+
1	Low	Low	Average
2-5	Low	Average	High
6+	Average	High	High

Table 15

- THE PARSING PROCESS

The principal component of the implementation of the automatic function point counting software is the parsing of the module text files. The module text files are parsed to collect such information as the number of base tables referenced, the number of base table columns referenced, the number on non-base tables referenced and the number of non-base table columns referenced which are required to determine the complexity rating for the components stated in the previous section, and consequently, obtain the Unadjusted Function Point count.

Base Tables Referenced

When parsing a module text file to search for the base tables referenced, the automatic function point counting software searches for the keywords *Base Table*. Once these two words are found, the parser examines the subsequent word to check whether it is a base table name. If it is, the name is stored, otherwise, the parser ignores the subsequent word and continues its search. An example of a base table appears as follows:

Base Table EMPLOYMENTS

The following two lines, however, would be ignored by the parser:

Base Table <Null>

Base Table Item False

Base Table Columns Referenced

When parsing a module text file to search for the base table columns referenced, the automatic function point counting software searches for the keywords *Base Table Item True.* Once these keywords have been found, the parser reverses its search direction to seek the name of the base table column. An example of a base table item appearing in the module text file is:

Name	**CEASE_DATE**
Class	<Null>
Item Type	Text Item
Canvas	CG$PAGE_1
Displayed	True
X Position	84
Y Position	22
.	
:	
Navigable	True
Next Navigation Item	<Null>
Previous Navigation Item	<Null>
Base Table Item	**True**
Primary Key	False
Insert Allowed	False
Query Allowed	False
Query Length	12
Case Insensitive Query	False

Non- Base Tables and Columns Referenced

The non - base tables and columns referenced can appear in triggers, record groups, report queries, and many other tables. The automatic function point counting software searches for those tables and columns that appear in *create, select, update,* and *delete* statements. Of these statements, the *select* statement seems to be the most common. For this reason, a detailed discussion of the parsing of the *select* statement is accommodated here.

The *SQL Language Quick Reference* (1992) defines a *select* statement as one that "queries one or more tables or views. (The *select* statement) returns rows and columns of data. (The *select*

statement) may be used as statement or as a subquery in another statement." The syntax for the *select* statement is provided in the Appendix.

One of the more significant challenges of the *select* statement is the flexibility provided by the SQL language. There are many variations to a *select* statement. The parsing of a *select* statement includes considerations such as nested *select* statements, those statements with references to functions consisting of a variable number of parameters, the possible spreading of the statement over an unpredictable number of lines, and the combination of any or all of these.

For a nested *select* statement, consider the following example:

```
SELECT roster_dec_hrs_fn
    INTO  temp_number
    FROM  rosters a
    WHERE a.id_number = :control.person_id_number
    AND   a.pers_pos_no = :control.pers_pos_no
    AND   a.rec_status != 'x'
    AND   a.commence_date = (SELECT max(commence_date)
                FROM rosters
                WHERE a.id_number = :control.person_id_number
                AND   a.pers_pos_no = :control.pers_pos_no
                AND   a.rec_status != 'x'
                AND   nvl(a.delete_flag, 'n') != 'y');
```

The parser scans such nested statements separately to determine the table and column names. In this example, the parser identifies *rosters* as the only table referenced, and *roster_dec_hrs_fn, commence_date* as the columns referenced.

The above example also illustrates references to functions. Function references can also be nested, and may contain any number of parameters. Consider the following:

```
SELECT substr(ltrim(rtrim(nvl(region_inst, main_inst))), 1, 6) FROM institutions;
```

This seemingly simple statement contains references to the functions:

substr consisting of *three* parameters,
ltrim consisting of *one* parameters,
rtrim consisting of *one* parameters,
nvl consisting of *one* parameters.

With such statements, the automatic function point counting parser examines the open- and close- brackets to distinguish the columns referenced from the functions. In this example, the parser correctly identifies *region_inst, main_inst* as the columns referenced and *institutions* as the table referenced.

To illustrate the spreading of a *select* statement over a variable number of lines, consider the simple SQL statement:

```
SELECT id_number, name, age FROM employments;
```

This same statement can also be legally coded in the following format:

1)
```
SELECT id_number, name, age
        FROM employments;
```

2)
```
SELECT
id_number,
name,
age FROM employments;
```

3)
```
  SELECT
    id_number
  ,    name    ,
          age FROM     employments;
```

The above illustrates only a sample of a seemingly infinite number of variations to the same statement! All of these statements are identified by the parser as consisting of the table *employments*, and the columns *id_number, name, age.*

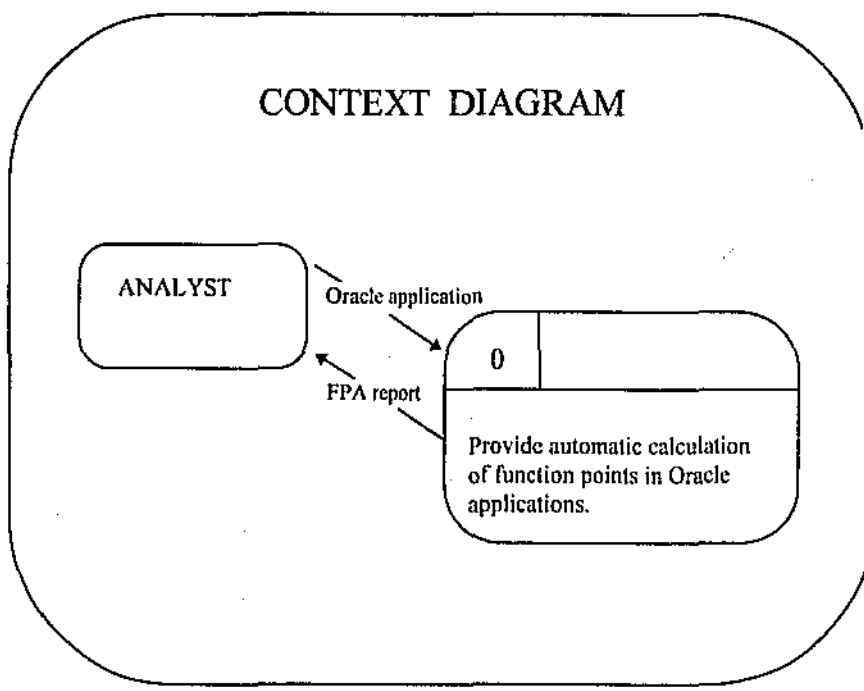
Enhancements

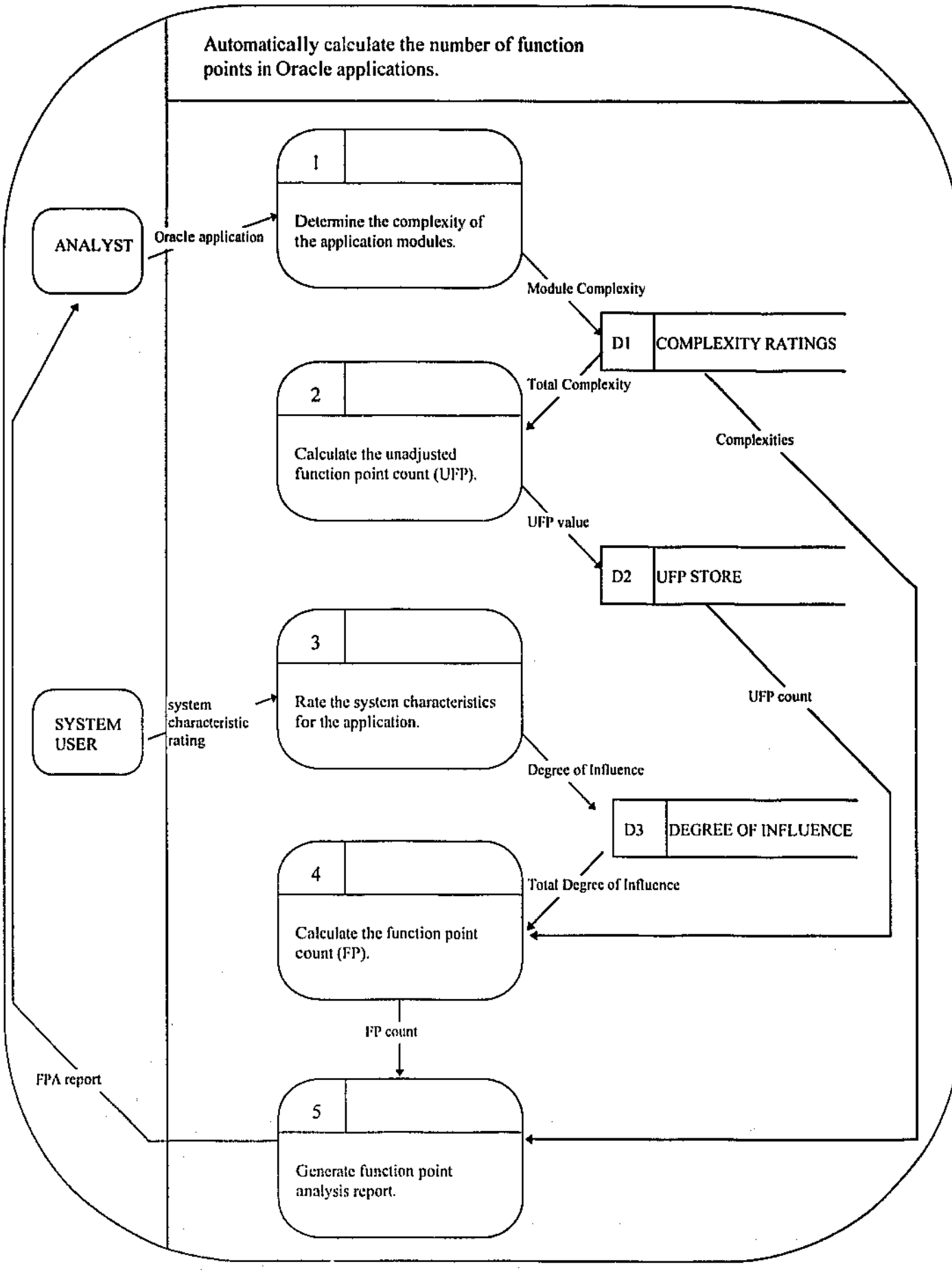
To extend the automatic function point counting prototype to a more comprehensive software, possible enhancements may be incorporated. These areas include reducing the parsing time and fine-tuning the parser.

The parsing time may be improved by upgrading the hardware or improving the parsing method. The current parser scans the Module Documentation more than once to count the number of base tables & columns, and non-base tables & columns. The parsing time may be reduced by limiting the parser to scan the Module Documentation once only.

This automatic Oracle applications estimator prototype can be fine-tuned since it provides a list of the names of the tables, their associated columns, the base tables and their associated columns. The names of the objects that have been detected by the software can be compared with the object names manually detected to determine and tune the accuracy of the parser. For example, occasionally, the software may display an object name that is, in fact, not an object, but a reserved word. This can be turned so that the reserved word will be ignored.

Data Flow Modelling is used as part of the design of the automatic calculation of function points in Oracle applications software. As stated by Oracle (1992b), upon which this model is based, the objective of this data flow model is "to ensure that functions are supplied with the necessary data in order to provide the information intended and also to identify the sources of the requisite data and the destinations for the information produced.




Automatically calculate the number of function points in Oracle applications.
1
Determine the complexity of the application modules.
ANALYST
Oracle application
Module Complexity
D1
COMPLEXITY RATINGS
2
Calculate the unadjusted function point count (UFP).
Total Complexity
Complexities
UFP value
D2
UFP STORE
3
Rate the system characteristics for the application.
SYSTEM USER
system characteristic rating
UFP count
Degree of Influence
D3
DEGREE OF INFLUENCE
4
Calculate the function point count (FP).
Total Degree of Influence
FP count
FPA report
5
Generate function point analysis report.

Chapter 12: RESULTS & CONCLUSIONS

The investigation part of this research has revealed very little previous work on the application of Function Point Analysis to Oracle applications. The most useful literature on this topic appears to be the QMS Project Management manual from Oracle Corporation. To date, no literature appears to be available on the *automatic* calculation of function points in Oracle applications.

This investigation also revealed that automatic function point analysis can be performed in a number of ways. Upon evaluation, it was concluded that the parsing of the module text documents generated by Oracle Developer/2000 would be the most suitable for an a posteriori evaluation of Oracle applications.

Software was developed that incorporated a number of the features discussed, to enable it to perform a comprehensive analysis, and automatic count of the number of function points for any Oracle application developed using Oracle Forms Designer, and Oracle Reports Designer. The successful implementation of this software appears to be the first of its kind. For this reason, it represents a worthwhile proof of concept for automatically counting function points for Oracle applications.

The automatic function point counting software produces a detailed report on the results of the automatic function point analysis at the end of its execution. This report presents the results in a very logical manner, following the format presented by Hignite et al (1993), to show the

calculations leading to the derivation of the final function point count. Since the report is produced as a Microsoft Word document, it can be easily printed at the user's discretion.
A complex form with 5 base tables, 55 base table columns and referencing 16 other non-base tables and 20 non-base table columns was parsed in approximately five minutes. Its Module Documentation, with a size of 761KBytes, spanned 14 425 lines over 263 pages.

The overall success rate of the automatic estimator reveals a proof of concept that provides the grounds for the possible launch of further researches in this area. This research and the development of the associated software is a worthwhile source of the proof of the concept. It is the first version produced, and for this reason, several possible areas of improvements may be incorporated in future researches to enhance the software.

Oracle is the second largest software company world-wide and there are many Oracle applications and Oracle users in the IT industry. For this reason, it is expected that the automatic function point counting of Oracle applications software should be beneficial in many project estimation exercises.

REFERENCES

1. Alford, Mark. (1991). Re: What the heck's a function point? *http://www.qucis.queensu.ca/Software-Engineering/archive/funcpoints.*
2. Behrens, Charles A. (1993). Measuring the Productivity of Computer Systems Development Activities With Function Points. *IEEE Transactions on Software Engineering.* Vol: SE-9 No: 6 p. 648-652.
3. Betteridge, R (1992). Successful experience of using function points to estimate project costs early in the life-cycle. *Information and Software Technology.* Vol:34 Iss:10 p.655-8. UK.
4. Dreger, J. (1989). Function Point Analysis. New Jersey : Prentice Hall.
5. Ferens, D; Gurner, R. (1992). An evaluation of three function point models for estimation of software effort. *Proceedings of the IEEE 1992 National Aerospace and Electronics Conference, NAECON 1992.* Vol:2 p.635-42. USA: IEEE.
6. Heemstra, F.J; Kusters, R.J. (1991). Function point analysis: evaluation of a software cost estimation model. *European Journal of Information Systems.* Vol: 1 Iss: 4 p.229-37. UK.
7. Hignite M, Johnson R, Foster K. (1993). The use of function point analysis to assess end user computing systems. *Journal of Computer Information Systems.* Vol:33 pp:46-50.
8. IBM Corporation (1975). DP Services Size and Complexity Factor Estimator, DP Services Technical Council
9. IFPUG (1990). International Function Point User's Group Counting Practices Manual, Release 3.0. *IFPUG.* Westerville, Ohio.
10. Information Engineering, Inc. (1996). About Function Point Analysis. *http//www.bannister.com/ifpug/home/docs/abfpa.html*

11. Internet: Funcnet. http://www.spr.com/library/funcnet.html

12. Jones, C. (1991). Applied Software Measurement. New York: McGraw Hill.

13. Kansala, K; Kitchenham, B. (1993). Inter-item correlations among function points. *Proceedings First International Software Metrics Symposium.*. pp. 11-14. USA.

14. Kremer, Chris F. (1987). An empirical validation of software cost estimation models. *Communications of the ACM.* Vol:30 No:5 pp:416-429.

15. Low, G.C; Jeffery, D.R. (1990). Function points in the estimation and evaluation of the software process. *IEEE Transactions on Software Engineering.* Vol:16 Iss:1 p.64-71. USA

16. Oracle (1986). SQL*Forms Designer's Reference 3.0 *Oracle Corporation*

17. Oracle (1988). Building Reports with Oracle Reports 2.0 *Oracle Corporation*

18. Oracle (1992). SQL Language Quick Reference *Oracle Corporation*

19. Oracle (1992b). Oracle Education Services Course Notes: Analysis Techniques *Oracle Corporation*

20. Oracle (1994). Oracle Forms 4.5 Developer's Guide *Oracle Corporation*

21. Oracle (1995). QMS Project Management *Oracle Corporation*

22. Rask R., Laamanen P., and Lyytinen K. (1993). Simulation and comparison of Albrecht's function point and DeMarco's function bang metric in a CASE environment. *IEEE Transactions on software engineering.* Vol:19 Iss:7 pp:661-71.

23. Symons, Charles R. (1988). Function Point Analysis: Difficulties and improvements. *IEEE Transactions on Software Engineering.* Vol:14 No:1 pp.2-11.

24. Tsoi, R; Yau,C. (1995). Assessing the fuzziness of general system characteristics in estimating software size. *Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems.* p.189-93. New York: IEEE.

25. van Wonderen, J. (1991). Another look at function point analysis. *Informatie.* Vol:34 Iss:6 p.334-43. Netherlands.

26. Weaver, K.R. (1989). Function points - a productivity measure benefits APL. *APL Quote Quad.* Vol: 19 Iss: 4 p. 377-80. USA.

27. Yau, C; Gan L. (1995). Comparing the top-down and bottom-up approaches of function analysis: a case study. *Software Quality Journal.* Vol:4 Iss:3 P.175-87. UK

Appendix A: FPA reports generated by Designer/2000

The FPA reports can be obtained from Designer/2000 by following these simple steps:

1. Run the Designer/2000 product: Repository Object Navigator (RON).
2. Within RON, select from the menu: Tools | Repository Reports (RR)
3. Within RR, expand in the object navigator: Reports | Function Point Analysis

A copy of each of the following FPA reports are included in this section.

MkI FPA Analysis Level	- CDFPA1A
FPA Analysis Level (DFDs & Event Models)	- CDFPAA2
FPA MkII (Design1)	- CDFPAD1
FPA MkII (Design2)	- CDFPAD2
Area Metric	- CDMETRIC

A printout sample copy of the *MkI FPA Design Level* report has been intentionally excluded as its layout is an exact replica of the *MkI FPA Analysis Level* layout.

Designer/2000

Report : FPA (IFPUG) - Analysis Level

Filename

Run by OWNER1

Report Date 16-MAY-96 03:06pm

Total Pages : 6

Parameter Values

Application System : TEST
Application Version : 1
Function Label : TEST
Help Inquiries : 1

For Application System : TEST
Version : 1
Starting at Function : TEST

maintain a person and their jobs

Unadjusted Function Point Count

Type	Description	Functional Complexity		Total
		Low	Average	
ILF	Internal Logical Files No. of Entities created, updated or deleted by one	1 * 7 = 7 With < 51	0 * 10 = 0 With > 50	7
EIF	External Interface Files No. of Entities read but not created, updated or	1 * 5 = 5 With < 51	0 * 7 0 With > 50	5
EI	External Inputs No. of Entity Create, Update and Delete Usages	3 * 3 = 9 With < 15	0 * 4 0 With > 14	9
EO	External Outputs No. of Entity Read Usages by a Leaf Function where there is no Create, Update or Delete usage of any	0 * 4 = 0 With < 20	0 * 5 0 With > 19	0

Type	Description	Functional Complexity		Total
		Low	Average	
EQ	External Inquiries			
	No. of Entity Function Usages counting 1 for any combination of Create, Read, Update and Delete involving 1 Entity and 1 Function (excluding read	2 * 3 = 6 With < 15	0 * 4 = 0 With > 14	6
	No. of low complexity External Inquiries for Help (e.g 1 for full screen help throughout the	1 * 3 = 3		3
FC	Function Count	Total Unadjusted Function Points		30

General System Characteristics

ID	Characteristic	Degree of Influence*
C1	Data Communications	
C2	Distributed Functions	
C3	Performance	
	Frequency: 0 PER — No. Of Functions: 1	
C4	Heavily Used Configuration	
C5	Transaction Rate	
C6	Online Data Entry	
C7	End User Efficiency	
C8	Online Update	
C9	Complex Processing	
C10	Reusability	
C11	Installation Ease	
C12	Operational Ease	

ID	Characteristic	Degree of Influence*
C13	Multiple Sites No. of Business Units to 0	
C14	Facilitate Change	
TDI	Total Degree of Influence	

* Degree of Influence Values:

- Not present or no influence = 0
- Insignificant influence = 1
- Moderate influence = 2
- Average influence = 3
- Significant influence = 4
- Strong influence, throughout = 5

Value Adjustment Factor (VAF) = (TDI * 0.01) + 0.65 = _____

Function Point Count (FP) = FC * VAF = _____

Designer/2000

FPA (IFPUG) - Analysis Level

End of Report

Designer/2000

Report : MKII FPA Information

Filename

Run by OWNER1

Report Date 16-MAY-96

Total Pages : 5

Parameter Values

Application System : TEST
Version : 1

Based at the Analysis Level (where DFD's and Event Models have been used)

For Application System : TEST
Version : 1

Information Processing Logic Size

Input

No. of dataflow contents on each dataflow which exists between an external entity and a function included as a logical transaction (No. of attribute types input)		0 *

Processing

No. of functions triggered by an event of type time (Logical transactions triggered by reaching a specific point in time)	0	
No. of functions where exists a dataflow from an external entity (Logical transactions triggered by external)	0	
Logical Transactions	0	
No. of distinct entities included in dataflows between datastores and functions included as logical transactions (No. of entity types referenced)		0 * 1.66

Output

No. of dataflow contents on each dataflow which exists between a function included as a logical transaction and an external entity (No. of attribute types output)	0 * 2.66
Information Processing Logic Size in Unadjusted Function Points	0

Technical Complexity Characteristics

1. Data Communications

2. Distributed Functions

3. Performance

Frequency	No. Of Functions

4. Heavily Used Configuration

5. Transaction Rate

6. Online Data Entry

7. End User Efficiency

8. Online Update

9. Complex Processing

10. Reusability

11. Installation Ease

12. Operational Ease

13. Multiple Sites (No. Of Business Units to Functions included as Logical Transaction Usages) 0

14. Facilitate Change

15. Interface Requirement Of Other Applications (Number of Functions included as Logical Transactions which are Master Functions in other Applications) 0

16. Security, Privacy, Audit

17. User Training Needs

18. Third Party Use

19. Documentation

20. Site Specific

Total Degree of Influence

Technical Complexity Adjustment
(0.65 + C * Total Degree of Influence
where C may take value of 0.005)

Size of System in MKII Function Points
(Information Processing Logic Size * Technical Complexity Adjustment)

Designer/2000

MKII FPA Information

End of Report

Designer/2000

Report : MKII FPA - Design Level 1

Filename

Run by OWNER1

Report Date 16-MAY-96

Total Pages : 6

Parameter Values

Application System	: TEST
Version	: 1
Include Shared Modules	: True
Module Type	: %
Language	: %

Based at the Design Level

For Application System : TEST
Version : 1

Information Processing Logic Size

Module Type SCREEN Language : Oracle Forms

INPUT - Number of attribute types input

No. of select column usages	=	48	
No. of select detailed column usages	=	45	
No. of Input parameters	=	0	
No. of modified parameters	=	0	
No. of other parameters	=	0	

PROCESSIN

No. of Modules	=	4	%TOTAL 66.67
No. of tables/Views - No. of entity types referenced	=	6	
No. of look-up links between table usages	=	0	
No. of base links between table usages	=	6	

OUTPUT - Number of attribute types output

No. of column usages in create/update/nullify	=	08
No. of detailed column usages in create/update/nullify	=	08
No. of output parameters	=	0
No. of modified parameters	=	0
No. of other parameters	=	0

Information Processing Logic Size

Module Type PACKAGE Language : PL/SQL

INPUT - Number of attribute types input

No. of select column usages	=	0
No. of select detailed column usages	=	0
No. of Input parameters	=	0
No. of modified parameters	=	0
No. of other parameters	=	0

PROCESSIN

No. of Modules	=	1	%TOTAL 16.67
No. of tables/Views - No. of entity types referenced	=	0	
No. of look-up links between table usages	=	0	
No. of base links between table usages	=	0	

OUTPUT - Number of attribute types output

No. of column usages in create/update/nullify	=	0
No. of detailed column usages in create/update/nullify	=	0
No. of output parameters	=	0
No. of modified parameters	=	0
No. of other parameters	=	0

Information Processing Logic Size

Module Type PROCEDURE Language : PL/SQL

INPUT - Number of attribute types input

No. of select column usages	= 0	
No. of select detailed column usages	= 0	
No. of Input parameters	= 0	
No. of modified parameters	= 0	
No. of other parameters	= 0	

PROCESSIN

No. of Modules	= 1	%TOTAL 16.67
No. of tables/Views - No. of entity types referenced	= 0	
No. of look-up links between table usages	= 0	
No. of base links between table usages	= 0	

OUTPUT - Number of attribute types output

No. of column usages in create/update/nullify	= 0
No. of detailed column usages in create/update/nullify	= 0
No. of output parameters	= 0
No. of modified parameters	= 0
No. of other parameters	= 0

TOTAL INPUTS

Sum of select column usages	=	48
Sum of select detailed column usages	=	45
Sum of input parameters	=	0
Sum of modified parameter	=	0
Sum of other parameters	=	0

TOTAL PROCESSING

Sum of modules	=	6
Sum of tables/view	=	6
Sum of look_up links between table usages	=	0
Sum of base links between table usages	=	6

TOTAL OUTPUTS

Sum of column usages in create/update/null	=	48
Sum of detailed column usages in create/update/nullify	=	48
Sum of output parameters	=	0
Sum of modified parameter	=	0
Sum of other parameters	=	0

Designer/2000

MKII FPA - Design Level 1

Designer/2000

Report : MKII FPA - Design Level 2

Filename

Run by OWNER1

Report Date 16-MAY-96

Total Pages : 4

Parameter Values

Application System	: TEST
Version	: 1
Include Shared Modules	: True
Module Type	:

Module Type

Technical Complexity Characteristics

1. Data Communications

2. Distributed Functions
   No. of nodes (Total in application system) = 0
   No. of node to module usages = 0
   Distinct No. of databases to table/view usages.

Module Type	Distinct No. of databases

   No. of snapshots (Total in application system = 0

3. Performance

4. Heavily used configuration

5. Transaction Rate

6. Online Data Entry

7. End User Efficiency

8. Online Update

9. Complex Processing

	Module Complexity			
	EASY	AVERAGE	DIFFICULT	OTHER
TOTAL				
%TOTAL				

10. Reusability
    No of modules owned by this Application system/version which are shared with others = 0

No of Business unit to module usages

11. Installation Ease

12. Operational Ease

13. Multiple Sites

	=	0
No of access group to module usages	=	0

14. Facilitate Change

15. Interface Requirement of other applications

No of modules owned by other application system/versions which are shared with this one	=	0

16. Security, privacy and audit

17. User training needs

Average No. of help text lines across tables which have help text (Total in application system)	=	
Average no of lines of help text across all the tables that could have help text (Total in application system)	=	0.00

18. Third Party Use

19. Documentation

20. Site Specific

Designer/2000

MKII FPA - Design Level 2

End of Report

Designer/2000

Report : Application System Metrics

Filename :

Run by : OWNER1

Report Date : 16-MAY-96 03:28pm

Total Pages : 3

Parameter Values

Application System : TEST
Version : 1
Shared? : False

1. Area Metric Based on Entity Model

$$N=(A*E)+(R*A)+(R*E)$$

Where A=No. of Attributes,
E=No. of Entities,
R=No. of Relationships

= 20

2. Area Metric Based on Schema Design

$$D=(T*C)+(T*F)+(F*C$$

Where T=No. of Tables,
C=No. of Columns which are not Foreign Keys,
F=No. of Foreign Keys

= 458

3. Area Metric Based on Comparison of Entity Model and Schema Design

$$M=D/N$$

= 22.9

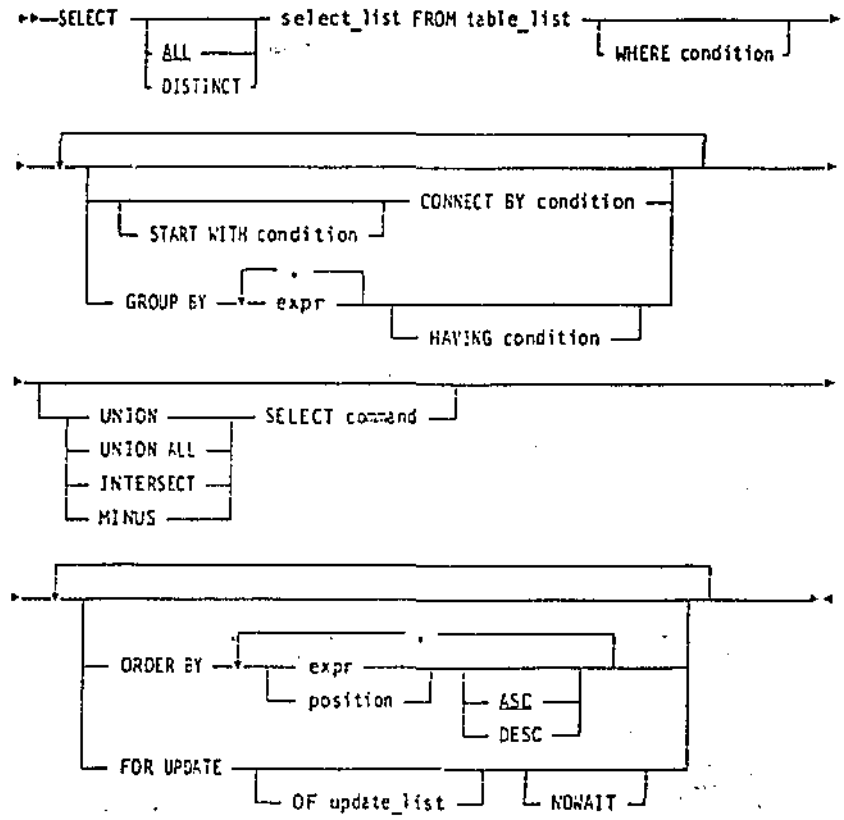
Designer/2000

Application System Metrics

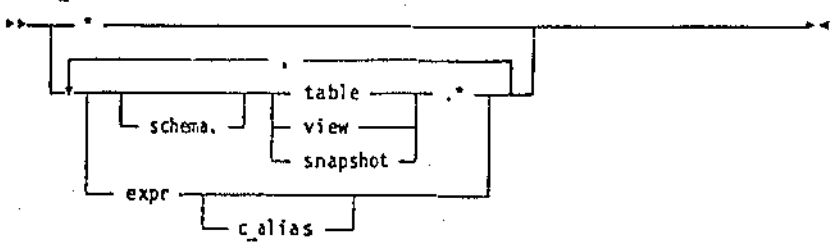
End of Report

Appendix B : The SQL SELECT statement

Oracle (1992) states the syntax of the SQL select statement as follows:

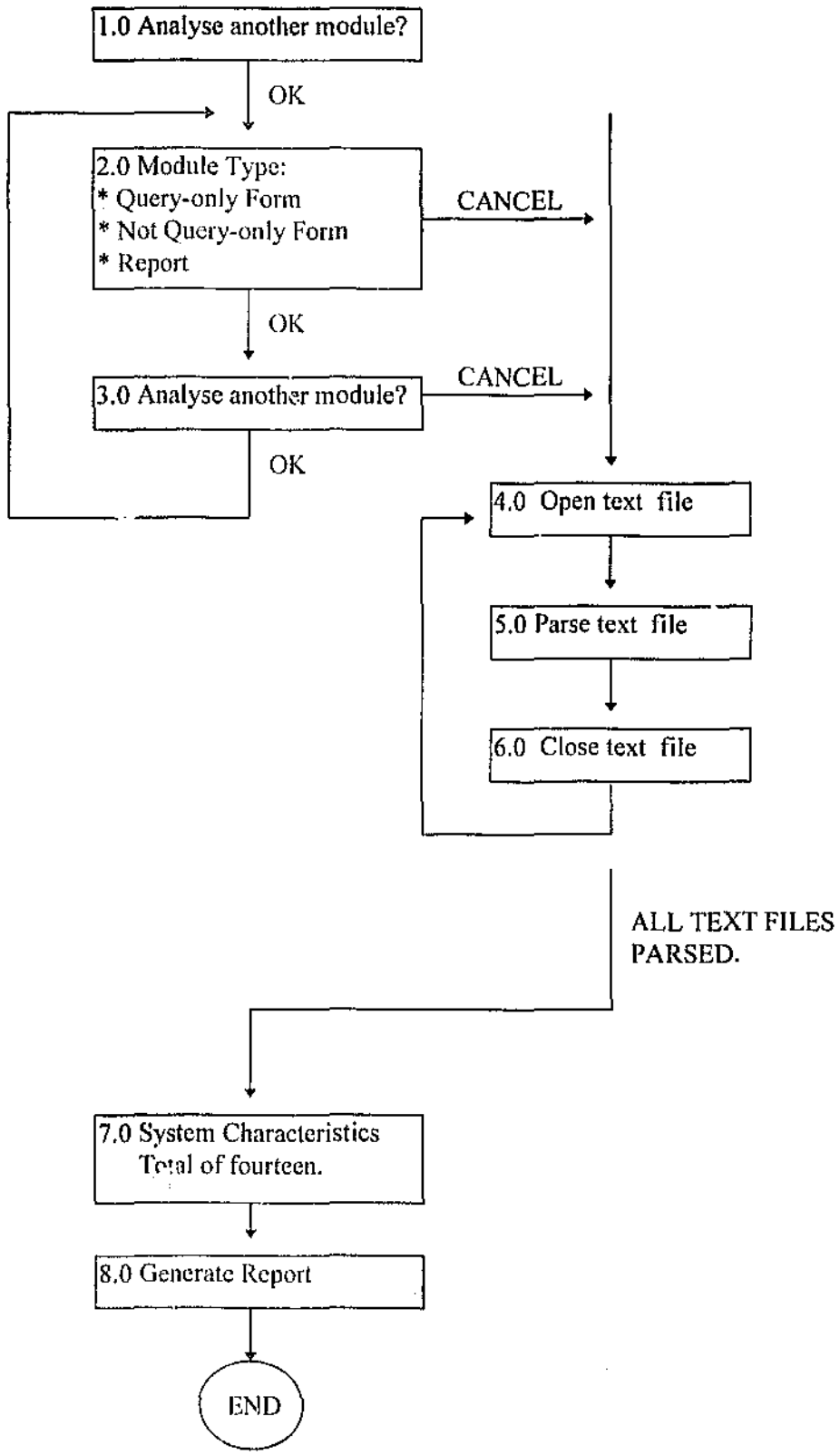


select_list ::=



Appendix C: Design of Function Point Counting Software

A layout of the General flow of the function point counting software is illustrated in the following diagram.



Appendix D: A sample FPA report produced

FUNCTION POINT CALCULATION REPORT

TOTAL UNADJUSTED FUNCTION POINT CALCULATION

	COMPONENT	LOW	AVERAGE	HIGH	TOTAL
EI	External Input	1 x 3 = 3	0 x 4 = 0	0 x 6 = 0	3
EO	External Output	0 x 4 = 0	0 x 5 = 0	0 x 7 = 0	0
EQ	External Inquiry	0 x 3 = 0	0 x 4 = 0	1 x 6 = 6	6
ILF	Internal Logical File	0 x 7 = 0	0 x 10 = 0	0 x 15 = 0	0
EIF	External Interface File	0 x 5 = 0	0 x 7 = 0	1 x 10 = 10	10
				Total Unadjusted FP	19

TECHNICAL COMPLEXITY CALCULATION

GENERAL SYSTEMS CHARACTERISTIC	DI	GENERAL SYSTEMS CHARACTERISTIC	DI
Data communications	3	On-Line update	1
Distributed Processing	1	Complex Processing	5
Performance	4	Reusability	0
Heavily used configuration	2	Installation ease	2
Transaction rates	4	Operational ease	0
On-Line data entry	3	Multiple sites	4
End-User efficiency	5	Facilitate change	2
		Total Degree of Influence (TDI)	**36**
		Technical Complexity Factor (TCF) = (TDI * .01) + 0.65	**1.01**

Total Function Points = TCF * TUFP
= 1.01* 19
= 19.19

**** A LIST OF POSSIBLE BASE TABLES ****
employments
COUNT = 1

**** A LIST OF POSSIBLE TABLES ****
pay_offers
COUNT = 1

**** A LIST OF POSSIBLE BASE TABLE COLUMNS ****
commence_date
COUNT = 1

**** A LIST OF POSSIBLE COLUMNS ****
pay_offer_desc
COUNT = 1

APPENDIX E: Source Code for the Function Point Counting Software

The following is a listing of the macro code that has been created to perform the automatic function point computation. It is based on the model illustrated in Appendix B.

```
Dim Shared ModType$(50)          'eg. Reports,Forms (Q-only, not Q-only) R,QO,NQO
Dim Shared ModName$(50)          'name of the module text file
Dim Shared ModCtr                'the number of module text files to be parsed
Dim Shared Analyse
Dim Shared NumT                  'number of external tables referenced by application.
Dim Shared NumCol                'number of ext table columns referenced by application.
Dim Shared NumIT                 'number of internal tables referenced by application.
Dim Shared NumICol               'number of internal table columns referenced by application.
Dim Shared NumB                  'number of base tables.
Dim Shared NumBCol               'number of base table columns referenced.
Dim Shared EO(3)'external output counter for Low, Av, Hi complexities
Dim Shared EI(3)'external input counter for Low, Av, Hi complexities
Dim Shared EQ(3)'external inquiry counter for Low, Av, Hi complexities
Dim Shared ILF(3)'internal logical counter for Low, Av, Hi complexities
Dim Shared EIF(3)'external interface file counter for Low, Av, Hi complexities
'System characteristic variables.
Dim Shared scDC
Dim Shared scDP
Dim Shared scP
Dim Shared scHUC
Dim Shared scTR
Dim Shared scODE
Dim Shared scEE
Dim Shared scOU
Dim Shared scCP
Dim Shared scR
Dim Shared scIE
Dim Shared scOE
Dim Shared scMS
Dim Shared scFC

'*************************************************************
'*                          MAIN
'*************************************************************
Sub MAIN
'Initialisations.
        ModCtr = 0       'Number of modules to be analysed.
        Analyse = 1
        NumT = 0
        NumCol = 0
        For IndexCount = 1 To 3
```

```
                EO(IndexCount) = 0
                EI(IndexCount) = 0
                EQ(IndexCount) = 0
                ILF(IndexCount) = 0
                EIF(IndexCount) = 0
        Next IndexCount

        CtnsAnalysePrompt       'Analyse another module?
        ParseTextFiles          'Parse each module text file.
        ILFComplexity           'Determine complexity for ILF.
        EIFComplexity           'Determine complexity for EIF.
        If ModCtr > 1 Then
                GetSysCharsDlg  'System characteristic ratings.
                FileOpen "C:\TEMP\REPORT.DOC"
        End If
        GenerateReport          'Generate a report on the FPA.
End Sub

'*********************************************************************
'*                       CtnsAnalysePrompt
'*********************************************************************
Sub     CtnsAnalysePrompt
        While Analyse = 1
                AnalyseModuleDlg
        Wend
End Sub

'*********************************************************************
'*                       ParseTextFiles
'*********************************************************************
Sub ParseTextFiles
        For Counter = 1 To (ModCtr - 1)
                NumB = 0
                NumBCol = 0
                ParseThisFile(ModName$(Counter))
                DetermineComplexity(ModType$(Counter))
        Next Counter
End Sub'OpenModNames

'*********************************************************************
'*                       EIFComplexity
'*********************************************************************
Sub EIFComplexity
        Select Case NumT
                Case 0
                Case 1
                        Select Case NumCol
                                Case 0
                                Case 1 To 50
                                        EIF(1) = EIF(1) + 1      'External Output = LO
                                Case Else
                                        EIF(2) = EIF(2) + 1      'External Output = AV
                        End Select
                Case 2 To 5
                        Select Case NumCol
```

```
                    Case 0
                    Case 1 To 19
                        EIF(1) = EIF(1) + 1        'External Output = LO
                    Case 20 To 50
                        EIF(2) = EIF(2) + 1        'External Output = AV
                    Case Else
                        EIF(3) = EIF(3) + 1        'External Output = HI
                End Select
            Case Else
                Select Case NumCol
                    Case 0
                    Case 1 To 19
                        EIF(2) = EIF(2) + 1        'External Output = AV
                    Case Else
                    EIF(3) = EIF(3) + 1        'External Output = HI
                End Select
        End Select
End Sub'EIFComplexity

'*********************************************************************
'*                    ILFComplexity
'*********************************************************************
Sub ILFComplexity
        Select Case NumIT
            Case 0
            Case 1
                Select Case NumICol
                    Case 0
                    Case 1 To 50
                        ILF(1) = ILF(1) + 1        'External Output = LO
                    Case Else
                        ILF(2) = ILF(2) + 1        'External Output = AV
                End Select
            Case 2 To 5
                Select Case NumICol
                    Case 0
                    Case 1 To 19
                        ILF(1) = ILF(1) + 1        'External Output = LO
                    Case 20 To 50
                        ILF(2) = ILF(2) + 1        'External Output = AV
                    Case Else
                        ILF(3) = ILF(3) + 1        'External Output = HI
                End Select
            Case Else
                Select Case NumICol
                    Case 0
                    Case 1 To 19
                        ILF(2) = ILF(2) + 1        'External Output = AV
                    Case Else
                        ILF(3) = ILF(3) + 1        'External Output = HI
                End Select
        End Select
End Sub'ILFComplexity

'*********************************************************************
```

```
'*                          TotalUFP
'*************************************************************************
Function TotalUFP
        TotalUFP = TotalEI + TotalEO + TotalEQ + TotalILF + TotalEIF
End Function'TotalUFP

'*************************************************************************
'*                          TotalEI
'*************************************************************************
Function TotalEI
        TotalEI = (EI(1) * 3) + (EI(2) * 4) + (EI(3) * 6)
End Function'TotalEI

'*************************************************************************
'*                          TotalEO
'*************************************************************************
Function TotalEO
        TotalEO = (EO(1) * 4) + (EO(2) * 5) + (EO(3) * 7)
End Function'TotalEO

'*************************************************************************
'*                          TotalEQ
'*************************************************************************
Function TotalEQ
        TotalEQ = (EQ(1) * 3) + (EQ(2) * 4) + (EQ(3) * 6)
End Function'TotalEQ

'*************************************************************************
'*                          TotalEIF
'*************************************************************************
Function TotalEIF
        TotalEIF = (EIF(1) * 5) + (EIF(2) * 7) + (EIF(3) * 10)
End Function'TotalEQ

'*************************************************************************
'*                          TotalILF
'*************************************************************************
Function TotalILF
        TotalILF = (ILF(1) * 7) + (ILF(2) * 10) + (ILF(3) * 15)
End Function'TotalILF

'*************************************************************************
'*                          TDI
'*************************************************************************
Function TDI
'Calcutes the total degree of influence by summing each of the system characteristics.
        TDI = scDC + scOU + scDP + scCP + scP + scR + scHUC + scIE + scTR + scOE + scODE +
scMS + scEE + scFC
End Function'TDI

'*************************************************************************
'*                          TCF
'*************************************************************************
Function TCF
'Total Complexity Factor
```

```
        TCF = (TDI * 0.01) + 0.65
End Function'TotalILF

'*********************************************************************
'*                          GetSysCharsDlg
'*********************************************************************
Sub GetSysCharsDlg
        Dim DIScale$(5)
        DIScale$(1) = "Not present"
        DIScale$(2) = "Incidental Influence"
        DIScale$(3) = "Moderate Influence"
        DIScale$(4) = "Average Influence"
        DIScale$(5) = "Significant Influence"
        Dim Dlg As UserDialog
        Begin Dialog UserDialog 768, 314, "System Characteristics"
                DropListBox 24, 37, 329, 76, DIScale$(), .DataCommList
                DropListBox 24, 70, 329, 76, DIScale$(), .DistributedList
                DropListBox 24, 103, 329, 76, DIScale$(), .PerformanceList
                DropListBox 24, 136, 329, 76, DIScale$(), .HeavyList
                DropListBox 24, 169, 329, 76, DIScale$(), .TransactionList
                DropListBox 24, 202, 329, 76, DIScale$(), .EntryList
                DropListBox 24, 235, 329, 76, DIScale$(), .EfficiencyList

                DropListBox 426, 37, 329, 76, DIScale$(), .UpdateList
                DropListBox 426, 70, 329, 76, DIScale$(), .ComplexList
                DropListBox 426, 103, 329, 76, DIScale$(), .ReuseList
                DropListBox 426, 136, 329, 76, DIScale$(), .InstallList
                DropListBox 426, 169, 329, 76, DIScale$(), .EaseList
                DropListBox 426, 202, 329, 76, DIScale$(), .MultipleList
                DropListBox 426, 235, 329, 76, DIScale$(), .ChangeList


                OKButton 297, 277, 88, 21
                CancelButton 415, 277, 88, 21
                Text 24, 24, 164, 13, "Data Communications", .Text1
                Text 24, 57, 212, 13, "Distributed Data Processing", .Text2
                Text 24, 90, 96, 13, "Performance", .Text14
                Text 24, 123, 207, 13, "Heavily Used Configuration", .Text3
                Text 24, 156, 132, 13, "Transaction Rate", .Text4
                Text 24, 189, 145, 13, "On-Line Data Entry", .Text5
                Text 24, 222, 151, 13, "End User Efficiency", .Text6
                Text 426, 24, 120, 13, "On-Line Update", .Text7
                Text 426, 57, 152, 13, "Complex Processing", .Text8
                Text 426, 90, 84, 13, "Reusability", .Text9
                Text 426, 123, 127, 13, "Installation Ease", .Text10
                Text 426, 156, 131, 13, "Operational Ease", .Text11
                Text 426, 189, 103, 13, "Multiple Sites", .Text12
                Text 426, 222, 133, 13, "Facilitate Change", .Text13
        End Dialog

        If Dialog(dlg) Then
                scDC = dlg.DataCommList
                scDP = dlg.DistributedList
                scP  = dlg.PerformanceList
                scHUC = dlg.HeavyList
```

```
                scTR = dlg.TransactionList
                scODE = dlg.EntryList
                scEE = dlg.EfficiencyList
                scOU = dlg.UpdateList
                scCP = dlg.ComplexList
                scR = dlg.ReuseList
                scIE = dlg.InstallList
                scOE = dlg.EaseList
                scMS = dlg.MultipleList
                scFC = dlg.ChangeList
        End If
End Sub

'*********************************************************************
'*                          GenerateReport
'*********************************************************************
Sub GenerateReport
'Generates a report on the results of the function point
'analysis process, based on the parsing of the individual
'files.
        Close #1
        Open "C:\TEMP\AAA.TXT" For Output As #2
        Print #2, "FPA REPORT" + Str$(Counter) + Chr$(13) + Chr$(13) + Chr$(13)
        GenerateTUFP
        GenerateSysChars
        CleanReport
'       Close #2
End Sub'GenerateReport

'*********************************************************************
'*                          GenerateTUFP
'*********************************************************************
Sub GenerateTUFP
        Bold
        Insert "FUNCTION POINT CALCULATION REPORT" + Chr$(13) + Chr$(13) + Chr$(13) +
Chr$(13) + "TOTAL UNADJUSTED FUNCTION POINT CALCULATION" + Chr$(13) + Chr$(13)
        TableInsertTable .ConvertFrom = "", .NumColumns = "6", .NumRows = "7", .InitialColWidth
= "2.7 cm", .Format = "0", .Apply = "167"
TableColumnWidth .RulerStyle = "1", .ColumnWidth = "1.44 cm"
        NextCell
        Bold
        Insert "COMPONENT"
        TableColumnWidth .RulerStyle = "1", .ColumnWidth = "4.25 cm"
        NextCell
        Bold
        Insert "LOW"
        NextCell
        Bold
        Insert "AVERAGE"
        NextCell
        Bold
        Insert "HIGH"
        NextCell
        Bold
        RightPara
```

```
Insert "TOTAL"
NextCell
Insert "EI"
NextCell
Insert "External Input"
NextCell
Insert Str$(EI(1)) + " x 3 = " + Str$(EI(1) * 3)
NextCell
Insert Str$(EI(2)) + " x 4 = " + Str$(EI(2) * 4)
NextCell
Insert Str$(EI(3)) + " x 6 = " + Str$(EI(3) * 6)
NextCell
Insert Str$(TotalEI)
RightPara
NextCell
Insert "EO"
NextCell
Insert "External Output"
NextCell
Insert Str$(EO(1)) + " x 4 = " + Str$(EO(1) * 4)
NextCell
Insert Str$(EO(2)) + " x 5 = " + Str$(EO(2) * 5)
NextCell
Insert Str$(EO(3)) + " x 7 = " + Str$(EO(3) * 7)
NextCell
Insert Str$(TotalEO)
RightPara
NextCell
Insert "EQ"
NextCell
Insert "External Inquiry"
NextCell
Insert Str$(EQ(1)) + " x 3 = " + Str$(EQ(1) * 3)
NextCell
Insert Str$(EQ(2)) + " x 4 = " + Str$(EQ(2) * 4)
NextCell
Insert Str$(EQ(3)) + " x 6 = " + Str$(EQ(3) * 6)
NextCell
Insert Str$(TotalEQ)
RightPara
NextCell
Insert "ILF"
NextCell
Insert "Internal Logical File"
NextCell
Insert Str$(ILF(1)) + " x 7 = " + Str$(ILF(1) * 7)
NextCell
Insert Str$(ILF(2)) + " x 10 = " + Str$(ILF(2) * 10)
NextCell
Insert Str$(ILF(3)) + " x 15 = " + Str$(ILF(3) * 15)
NextCell
Insert Str$(TotalILF)
RightPara
NextCell
Insert "EIF"
```

```
        NextCell
        Insert "External Interface File"
        NextCell
        Insert Str$(EIF(1)) + " x 5 = " + Str$(EIF(1) * 5)
        NextCell
        Insert Str$(EIF(2)) + " x 7 = " + Str$(EIF(2) * 7)
        NextCell
        Insert Str$(EIF(3)) + " x 10 = " + Str$(EIF(3) * 10)
        NextCell
        Insert Str$(TotalEIF)
        RightPara
        NextCell
        NextCell
        NextCell
        NextCell
        NextCell
        Bold
        Insert "Total Unadjusted FP"
        NextCell
        Bold
        InsertPara
        Insert Str$(TotalUFP)
        RightPara
        LineDown 1
End Sub'GenerateTUFP

'*********************************************************************
'*                          GenerateSysChars
'*********************************************************************
Sub GenerateSysChars
        Insert Chr$(13) + Chr$(13) + Chr$(13)
        Bold
        Insert "TECHNICAL COMPLEXITY CALCULATION" + Chr$(13) + Chr$(13)
        TableInsertTable .ConvertFrom = "", .NumColumns = "4", .NumRows = "10",
.InitialColWidth = "Auto", .Format = "0", .Apply = "167"
        Bold
        Insert "GENERAL SYSTEMS CHARACTERISTIC"
        NextCell
        Bold
        RightPara
        Insert "DI"
        TableColumnWidth .RulerStyle = "1", .ColumnWidth = "1.2 cm"
        NextCell
        Bold
        Insert "GENERAL SYSTEMS CHARACTERISTIC"
        NextCell
        Bold
        RightPara
        Insert "DI"
        TableColumnWidth .RulerStyle = "1", .ColumnWidth = "1.2 cm"
        NextCell
        Insert "Data communications"
        NextCell
        RightPara
        Insert Str$(scDC)
```

```
NextCell
Insert "On-Line update"
NextCell
RightPara
Insert Str$(scOU)
NextCell
Insert "Distributed Processing"
NextCell
RightPara
Insert Str$(scDP)
NextCell
Insert "Complex Processing"
NextCell
RightPara
Insert Str$(scCP)
NextCell
Insert "Performance"
NextCell
RightPara
Insert Str$(scP)
NextCell
Insert "Reusability"
NextCell
RightPara
Insert Str$(scR)
NextCell
Insert "Heavily used configuration"
NextCell
RightPara
Insert Str$(scHUC)
NextCell
Insert "Installation ease"
NextCell
RightPara
Insert Str$(scIE)
NextCell
Insert "Transaction rates"
NextCell
RightPara
Insert Str$(scTR)
NextCell
Insert "Operational ease"
NextCell
RightPara
Insert Str$(scOE)
NextCell
Insert "On-Line data entry"
NextCell
RightPara
Insert Str$(scODE)
NextCell
Insert "Multiple sites "
NextCell
RightPara
Insert Str$(scMS)
```

```
        NextCell
        Insert "End-User efficiency"
        NextCell
        RightPara
        Insert Str$(scEE)
        NextCell
        Insert "Facilitate change"
        NextCell
        RightPara
        Insert Str$(scFC)
        NextCell
        NextCell
        NextCell
        Insert Chr$(13)
        Bold
        Insert "Total Degree of Influence (TDI)"
        NextCell
        Bold
        RightPara
        InsertPara
        Insert Str$(TDI)
        LineDown 1
        CharLeft 1
        Bold
        Insert "Technical Complexity Factor (TCF)"
        InsertPara
        Insert "= (TDI * .01) + 0.65"
        NextCell
        RightPara
        InsertPara
        Bold
        Insert Str$(TCF)
        LineDown 1
        Insert Chr$(13) + Chr$(13)
        Bold
        Insert "Total Function Points" + Chr$(9) + "= TCF * TUFP" + Chr$(13)
        Insert Chr$(9) + Chr$(9) + Chr$(9) + "= " + Str$(TCF) + "*" + Str$(TotalUFP) + Chr$(13)
        Insert Chr$(9) + Chr$(9) + Chr$(9) + "= " + Str$(TCF * TotalUFP) + Chr$(13) + Chr$(13) +
Chr$(13)
End Sub'GenerateSysChars

'*********************************************************************
'*                          CleanReport
'*********************************************************************
Sub CleanReport
        EditSelectAll
        Font "Arial"
        FontSize 8
        LineDown 1
End Sub

'*********************************************************************
'*                          AnalyseModuleDlg
'*********************************************************************
Sub AnalyseModuleDlg
```

```
        Begin Dialog UserDialog 355, 85, "Function Point Analysis"
                Text 24, 8, 300, 15, "Analyse another (or the first) module for", .AnalyseIt
                Text 23, 26, 125, 13, "this application?", .Text1
                OKButton 82, 53, 88, 21
                CancelButton 184, 53, 88, 21
        End Dialog

        Dim dlg As UserDialog
        ModCtr = ModCtr + 1
        Button = Dialog(dlg)        'display the dialog
        If Button = -1 Then 'ok button
                SelectModTypeDlg
        ElseIf Button = 0 Then'cancel button
                Analyse = 0
        End If
End Sub

'*********************************************************************
'*                        SelectModTypeDlg
'*********************************************************************
Sub SelectModTypeDlg
'Prompt for the module type.
        Begin Dialog UserDialog 400, 118, "Function Point Analysis"
                OKButton 99, 89, 88, 21
                CancelButton 215, 89, 88, 21
                GroupBox 18, 7, 369, 77, "Select the File Module Type"
                OptionGroup .ModType
                OptionButton 48, 23, 250, 16, "Oracle Forms : &Query-only"
                OptionButton 48, 41, 250, 16, "Oracle Forms : &Not query-only"
                OptionButton 48, 59, 250, 16, "Oracle &Reports "
        End Dialog
        Dim dlg As UserDialog
        If Dialog(dlg) Then
                Select Case dlg.ModType
                        Case 0
                                ModType$(ModCtr) = "QO"
                                GetFileName
                                AnalyseModuleDlg
                        Case 1
                                ModType$(ModCtr)  = "NQO"
                                GetFileName
                                AnalyseModuleDlg
                        Case 2
                                ModType$(ModCtr) = "R"
                                GetFileName
                                AnalyseModuleDlg
                        Case Else
                                MsgBox "Not a list style"
                End Select
        Else 'Cancel button
                ModCtr = ModCtr - 1
        End If
End Sub 'SelectModTypeDlg

'*********************************************************************
```

```
'*                              GetFileName
'*********************************************************************
Sub GetFileName
'Fills an array with the names of all files in the current directory. 'The instructions first count the files to
determine the size of the 'array. Then they define the array, fill it
'with the filenames, and sort the elements. The array is then presented
'in a user-defined dialog box.
        temp$ = Files$("*.*")
        Counter = - 1
        While temp$ <> ""
                Counter = Counter + 1
                temp$ = Files$()
        Wend
        If Counter > - 1 Then
                Dim list$(Counter)
                list$(0) = Files$("*.*")
                For i = 1 To Counter
                        list$(i) = Files$()
                Next i
                SortArray list$()
        Else
                MsgBox "No files in current directory."
        End If

        FileNameDlg(List$())
End Sub

'*********************************************************************
'*                              FileNameDlg(FileList$())
'*********************************************************************
Sub FileNameDlg(FileList$())
On Error Resume Next
Begin Dialog UserDialog 440, 160, "Function Point Analysis"
        Text 29, 8, 261, 13, "Select the text file to be analysed:"
        ComboBox 29, 25, 380, 84, FileList$(), .FileList
        OKButton 134, 123, 88, 21
        CancelButton 248, 123, 88, 21
End Dialog
Dim dlg As UserDialog
If Dialog(dlg) Then
        ModName$(ModCtr) = dlg.FileList
Else
        ModCtr = ModCtr - 1
End If
End Sub

'*********************************************************************
'*                              ParseThisFile(ThisFile$)
'*********************************************************************
Sub ParseThisFile(ThisFile$)
If Files$(ThisFile$) <> "" Then
        FileOpen ThisFile$
        SearchTheTables
        FileClose(2)
Else
```

```
        MsgBox "File " + ThisFile$ + " not found."
End If
End Sub

'*********************************************************************
'*                          SearchTheTables
'*********************************************************************
Sub SearchTheTables
'DETERMINE THE NUMBER OF POSSIBLE BASE TABLES.
        Open "C:\TEMP\REPORT.DOC" For Output As #1
        Print #1, "**** A LIST OF POSSIBLE BASE TABLES ****"
        SearchIt("B", "Table", 1, "base tables")
        Close #1
'DETERMINE THE NUMBER OF POSSIBLE TABLES REFERENCED.
        Open "C:\TEMP\REPORT.DOC" For Append As #1
        Print #1, "**** A LIST OF POSSIBLE TABLES ****"
        SearchIt("T", "From", 0, "tables referenced")
        Close #1
'DETERMINE THE NUMBER OF BASE TABLE COLUMNS.
        Open "C:\TEMP\REPORT.DOC" For Append As #1
        Print #1, "**** A LIST OF POSSIBLE BASE TABLE COLUMNS ****"
        SearchIt("BCOL", "Base Table Item                True", 0, "base table columns")
        Close #1
'DETERMINE THE NUMBER OF COLUMNS REFERENCED.
        Open "C:\TEMP\REPORT.DOC" For Append As #1
        Print #1, "**** A LIST OF POSSIBLE COLUMNS ****"
        SearchIt("TCOL", "Select", 0, "columns referenced")
        Close #1
End Sub

'*********************************************************************
'*                          SkipSpacesRight
'*********************************************************************
Sub SkipSpacesRight
        'Skip spaces & CR
        While (Asc(Selection$()) = 13 Or Asc(Selection$()) = 9 Or Asc(Selection$()) = 32)
                CharRight 1, 0
        Wend
End Sub

'*********************************************************************
'*                          SkipSpacesLeft
'*********************************************************************
Sub SkipSpacesLeft
        'Skip spaces & CR
        CharLeft 1, 0
        While (Asc(Selection$()) = 13 Or Asc(Selection$()) = 9 Or Asc(Selection$()) = 32)
                CharLeft 1, 0
        Wend
        CharRight 1, 0
End Sub

'*********************************************************************
'*                          NameWith_Symbol$
'*********************************************************************
```

```
Function NameWith_Symbol$
        'Assumes that the word is already selected.
        TempWord$ = Selection$()
        CharRight 1, 0
        While Selection$() = "_"
                TempWord$ = TempWord$ + Selection$()
                CharRight 1, 0
                SelectCurWord
                TempWord$ = TempWord$ + Selection$()
                CharRight 1, 0
        Wend
        WordLeft 1. 1
        NameWith_Symbol$ = TempWord$
End Function

'*********************************************************************
'*                        ADuplicate$(PossTabName$)
'*********************************************************************
Function ADuplicate$(PossTabName$)
Dim NameLength
        ADup$ = "Y"
        NameLength = Len(LTrim$(RTrim$(PossTabName$)))
        Open "C:\TEMP\REPORT.DOC" For Input As #1
        Input #1, name$
        If name$ = "" Then
                ADup$ = "N"
                Goto Finish
        End If
                While LCase$(Left$(Name$, NameLength)) <>
LCase$(LTrim$(RTrim$(PossTabName$)))
                        If Eof(1) Then
                                ADup$ = "N"
                                Goto Finish
                        End If
                        Input #1, Name$
                Wend
        ADup$ = "Y"
Finish:
        ADuplicate$ = ADup$
        Close #1
End Function'ADuplicate$

'*********************************************************************
'*                        NotDuplicate(TempWord$)
'*********************************************************************
Function NotDuplicate(TempWord$)
        NotDup = 0
        Close #1
        If (TempWord$ <> Chr$(32) And ADuplicate$(TempWord$) = "N") Then'not blank nor
duplicate
                Open "C:\TEMP\REPORT.DOC" For Append As #1
                Print #1, LCase$(TempWord$)
                NotDup = 1
                Close #1
        End If'Print table
```

```
        NotDuplicate = NotDup
End Function

'*********************************************************************
'*                          IsLastColInStatement
'*********************************************************************
Function IsLastColInStatement
'Assumes that the word is highlighted.
        IsLast = 0
        WordRight 1, 0
        SkipSpacesRight
        SelectCurWord
        If (UCase$(Selection$()) = "FROM" Or UCase$(Selection$()) = "INTO") Then
                IsLast = 1
        End If'is FROM
        WordLeft 1, 0
        SkipSpacesLeft
        WordLeft 1, 1
        IsLastColInStatement = IsLast
End Function

'*********************************************************************
'*                          UsesFunction
'*********************************************************************
Function UsesFunction
'Assumes that word is highlighted
        UsesFn = 0
        CharRight 1, 0
        SkipSpacesRight
        If Selection$() = "(" Then
                UsesFn = 1
        End If
        SkipSpacesLeft
        WordLeft 1, 1
        UsesFunction = UsesFn
End Function

'*********************************************************************
'*                          ReferenceColumns
'*********************************************************************
Function ReferenceColumns
'Assumes that the word to the right of "select" is highlighted.
NotColumn$ = "COUNT"
leaveloop = 0
TempCols = 0
If InStr(NotColumn$, UCase$(Selection$())) <> 0 Then
'eg. neglect : select count(*) from emp;
        LeaveLoop = 1
End If
While Leaveloop = 0
        CharRight 1, 0
        SkipSpacesRight
        If Selection$() = "," Or Selection$() = "_" Then
                'eg. select id_number, name from emp;
                SkipSpacesLeft
```

```
                WordLeft 1, 1
                TempWord$ = NameWith_Symbol$
                Close #1
                If UsesFunction = 1 Then
                'eg. select to_date(to_char(...))..
                        CharRight 1, 0
                        SkipSpacesRight
                        Goto UsesFunction
                Else
                        LeaveLoop = IsLastColInStatement
                        If NotDuplicate(TempWord$) = 1 Then
                                TempCols = TempCols + 1
                        End If
                        CharRight 1, 0
                        SkipSpacesRight
                        WordRight 1
                        SelectCurWord
                End If'of UsesFunction
        ElseIf Selection$() = "." Then
                ' eg. select e.idnumber from emp e;
                CharRight 1, 0
                WordRight 1, 1
                TempWord$ = NameWith_Symbol$
                Close #1
                LeaveLoop = IsLastColInStatement
                If NotDuplicate(TempWord$) = 1 Then
                        TempCols = TempCols + 1
                End If
                CharRight 1, 0
                SkipSpacesRight
                WordRight 1
                SelectCurWord
        ElseIf Selection$() = "(" Then
UsesFunction:
                ' eg. select ltrim(rtrim(name)) from emp;
                OpenBracketCounter = 1
                CharRight 1, 0
                While Selection$() <> ")"
                        If Selection$() = "(" Then
                                OpenBracketCounter = OpenBracketCounter + 1
                        End If
                        CharRight 1, 0
                Wend
                EditFind .Find = "(", .Direction = 1, .MatchCase = 0, .WholeWord = 1, .PatternMatch
= 0, .SoundsLike = 0, .Format = 0, .Wrap = 0
                If EditFindFound() <> 0 Then
                        CharRight 1, 0
                        SkipSpacesRight
                'Ensure it is not a form field eg :id_number
                        If Selection$() = ":" Then
                                CharRight 1, 0
                        End If
                'Check for alias eg. emp.id_number
                        WordRight 1, 0
                        If Selection$() = "." Then
```

```
                        CharRight 1, 0
                        SkipSpacesRight
                    Else
                        WordLeft 1, 0
                    End If
                    SelectCurWord
                    TempWord$ = NameWith_Symbol$
                    Close #1
                    If NotDuplicate(TempWord$) = 1 Then
                        TempCols = TempCols + 1
                    End If

                    'Check if this is the last column in statement
                    CharRight 1, 0
                    For CloseBracketCounter = 1 To OpenBracketCounter
                        EditFind .Find = ")", .Direction = 0, .MatchCase = 0, .WholeWord =
1, .PatternMatch = 0, .SoundsLike = 0, .Format = 0, .Wrap = 0
                        If EditFindFound() <> 0 Then
                            CharRight 1, 0
                        Else
                            LeaveLoop = 1
                            MsgBox "Query statement contains a syntax error."
                        End If
                    Next CloseBracketCounter

                    SkipSpacesRight
                    If (UCase$(Selection$()) = "F" Or UCase$(Selection$()) = "I") Then
                        SelectCurWord
                        If (UCase$(Selection$()) = "FROM" Or UCase$(Selection$()) =
"INTO") Then
                            LeaveLoop = 1
                        End If
                        CharLeft 1, 0
                    End If'is FROM
                    WordRight 1, 0
                    SkipSpacesRight
                    WordRight 1, 0
                    SkipSpacesRight
                    CharLeft 1, 0
                Else
                    MsgBox "Error searching for open bracket."
                    LeaveLoop = 1
                End If
            ElseIf (UCase$(Selection$()) = "F" Or UCase$(Selection$()) = "I") Then
                'Ensure that it is the word "FROM" or "INTO"
                SelectCurWord
                If (UCase$(Selection$()) = "FROM" Or UCase$(Selection$()) = "INTO") Then
                    CharLeft 2, 0
                    SkipSpacesLeft
                    SelectCurWord
                    TempWord$ = NameWith_Symbol$
                    Close #1
                    If NotDuplicate(TempWord$) = 1 Then
                        TempCols = TempCols + 1
                    End If
```

```
                    LeaveLoop = 1
                End If
        Else
        'most likely not a proper select statement.
                Leaveloop = 1
        End If
Wend
ReferenceColumns = TempCols
End Function

'*************************************************************************
'*                    DetermineComplexity(ThisModType$)
'*************************************************************************
Sub DetermineComplexity(ThisModType$)
'called from SearchIt.
'This proc determines the complexity rating of Low, Average, or High
'for a component.
        Select Case ThisModType$
                Case "R"        'External Outputs
                        Select Case NumB
                                Case 0 To 1
                                        Select Case NumBCol
                                                Case 0
                                                Case 1 To 19
                                                        EO(1) = EO(1) + 1       'External
Output = LO
                                                Case Else
                                                        EO(2) = EO(2) + 1       'External
Output = AV
                                        End Select
                                Case 2 To 3
                                        Select Case NumBCol
                                                Case 0
                                                Case 1 To 5
                                                        EO(1) = EO(1) + 1       'External
Output = LO
                                                Case 6 To 19
                                                        EO(2) = EO(2) + 1       'External
Output = AV
                                                Case Else
                                                        EO(3) = EO(3) + 1       'External
Output = HI
                                        End Select
                                Case Else
                                        Select Case NumBCol
                                                Case 0
                                                Case 1 To 5
                                                        EO(2) = EO(2) + 1       'External
Output = AV
                                                Case Else
                                                        EO(3) = EO(3) + 1       'External
Output = HI
                                        End Select
                        End Select
                Case "NQO"      'External Inputs
```

```
                Select Case NumB
                        Case 0 To 1
                                Select Case NumBCol
                                        Case 0
                                        Case 1 To 15
                                                EI(1) = EI(1) + 1 'External Output = LO
                                        Case Else
                                                EI(2) = EI(2) + 1 'External Output = AV
                                End Select
                        Case 2
                                Select Case NumBCol
                                        Case 0
                                        Case 1 To 4
                                                EI(1) = EI(1) + 1 'External Output = LO
                                        Case 5 To 15
                                                EI(2) = EI(2) + 1 'External Output = AV
                                        Case Else
                                                EI(3) = EI(3) + 1 'External Output = HI
                                End Select
                        Case Else
                                Select Case NumBCol
                                        Case 0
                                        Case 1 To 4
                                                EI(2) = EI(2) + 1 'External Output = AV
                                        Case Else
                                                EI(3) = EI(3) + 1 'External Output = HI
                                End Select
                End Select
        Case "QO"       'External Inquiries
                Select Case NumB
                        Case 0 To 1
                                Select Case NumBCol
                                        Case 0
                                        Case 1 To 15
                                                EQ(1) = EQ(1) + 1       'External
Inquiry = LO
                                        Case Else
                                                EQ(2) = EQ(2) + 1       'External
Inquiry = AV
                                End Select
                        Case 2
                                Select Case NumBCol
                                        Case 0
                                        Case 1 To 4
                                                EQ(1) = EQ(1) + 1       'External
Inquiry = LO
                                        Case 5 To 15
                                                EQ(2) = EQ(2) + 1       'External
Inquiry = AV
                                        Case Else
                                                EQ(3) = EQ(3) + 1       'External
Inquiry = HI
                                End Select
                        Case Else
                                Select Case NumBCol
```

```
                                        Case 0
                                        Case 1 To 4
                                                EQ(2) = EQ(2) + 1          'External
Inquiry = AV
                                        Case Else
                                                EQ(3) = EQ(3) + 1          'External
Inquiry = HI
                                End Select
                        End Select
        Case Else
                EO(1) = - 2
        End Select
End Sub

'*********************************************************************
'*              SearchIt(ItemType$, WordToSearch$, MatchTheCase, SearchItem$)
'*********************************************************************
Sub SearchIt(ItemType$, WordToSearch$, MatchTheCase, SearchItem$)
        LeaveSearchloop = 0
        StoredLineNumber = - 3
        CommentedLine$ = "--, /*"
        SpecialChar$ = Chr$(9) + Chr$(11) + Chr$(32) + Chr$(34) + Chr$(40) + Chr$(41) + Chr$(58)
+ Chr$(59) + Chr$(60) + Chr$(62) + Chr$(160)
        NotTable$ = "FROM, ITEM, NULL, DUAL, THE, THIS, AND, NAME);" + SpecialChar$
        EndOfDocument
        Insert Chr$(13) + WordToSearch$ + " ENDOFDOCUMENTSYMBOL "
        StartOfDocument
        While LeaveSearchloop = 0
                EditFind .Find = WordToSearch$, .Direction = 0, .MatchCase = MatchTheCase,
                        .WholeWord = 1, .PatternMatch = 0, .SoundsLike = 0, .Format =
        0, .Wrap = 0
                If EditFindFound() <> 0 Then
                        If SelInfo(10) <> StoredLineNumber Then
                        'ie. if there are >2 WordToSearch$ words in one line
                                StartOfLine
                                CharRight 1, 1
                                        While LTrim$(Selection$()) = ""
                                                CharRight 1, 1
                                Wend
                                CharRight 1, 1
                                StoredLineNumber = SelInfo(10)
                                If InStr(CommentedLine$, LTrim$(Selection$())) <> 0 Then
                                        LineDown
                                        Goto EndOfLoopLabel
                                Else
                                        StartOfLine
                                        EditFind .Find = WordToSearch$, .Direction = 0,
.MatchCase = MatchTheCase, .WholeWord = 1, .PatternMatch = 0, .SoundsLike = 0, .Format = 0,
.Wrap = 0
                                End If'Commented Line
                        End If'SelInfo

                        CharRight 1, 0
                        SkipSpacesRight
                        SelectCurWord
```

```
                    If Selection$() = "ENDOFDOCUMENTSYMBOL" Then
                        EditReplace .Find = WordToSearch$ + "
ENDOFDOCUMENTSYMBOL", .Replace = "", .Direction = 1, .MatchCase = 1, .WholeWord = 1,
.PatternMatch =  0, .SoundsLike = 0, .ReplaceOne, .Format = 0, .Wrap = 1
                        EditClear - 2
                        LeaveSearchloop = 1
                    Else
                        If ItemType$ = "BCOL" Then
                            CharRight 1
            EditFind .Find = "Name                    ", .Direction = 1, .MatchCase = 1,
.WholeWord = 1, .PatternMatch = 0, .SoundsLike = 0, .Format = 0, .Wrap = 0
                            WordRight 1, 0
                            SelectCurWord
                        End If

If ItemType$ = "TCOL" Then
    Counter = ReferenceColumns + Counter

    CharRight 1, 0
    SkipSpacesRight
Else
                        If InStr(NotTable$, UCase$(Selection$())) = 0 Then
                            If Selection$() = Chr$(13) Then'Carriage return
                                CharRight 1, 0
                                SelectCurWord
                            End If
                            TempWord$ = Selection$()
                            CharRight 1, 0
                            While Selection$() = "_"
                                TempWord$ = TempWord$ + "_"
                                CharRight 1, 0
                                SelectCurWord
                                TempWord$ = TempWord$ + Selection$()
                                CharRight 1, 0
                            Wend
                             Close #1
                            If NotDuplicate(TempWord$) = 1 Then
                                Counter = Counter + 1
                            End If
                        Else
                            CharRight 1, 0
                        End IfNot table
End IfTCOL
                    End IfEndOfDocSymbol
EndOfLoopLabel:
                End IfEditFindFound
    If ItemType$ = "BCOL" Then
                CharRight 1
EditFind .Find = WordToSearch$, .Direction = 0, .MatchCase = MatchTheCase, .WholeWord = 1,
.PatternMatch = 0, .SoundsLike = 0, .Format = 0, .Wrap = 0
    End If

    Wend
```

```
        Close #1
        Select Case ItemType$
                Case "T"            'Tables referenced in select statements.
                        NumT = NumT + Counter
                Case "TCOL"'Columns referenced in select statements.
                        NumCol = NumCol + Counter
                Case "B"
                        NumB = NumB + Counter
                Case "BCOL"
                        NumBCol = NumBCol + Counter
        End Select

        Open "C:\TEMP\REPORT.DOC" For Append As #1
        Print #1, "COUNT =  " + Str$(Counter) + Chr$(13) + Chr$(13) + Chr$(13)
        Close #1
End Sub'SearchIt
```