# Identification of the structural weaknesses of Function Point metrics.

3rd Annual Oregon Workshop on Software Metrics

Portland, Oregon - March 18-19, 1991

ALAIN ABRAN

Montreal Trust
1800 McGill College Av.
Montreal, Québec, Canada, H3A 3K9
Tel: (514) 982-7175
Fax: (514) 982-7069

PIERRE N. ROBILLARD

École Polytechnique de Montréal
P.O. Box 6079, Succ. A
Montréal, Québec, Canada H3C 3A7
Tel: (514) 340-4238
Fax: (514) 340-3240

Revised: Feb. 18th, 1991

# Identification of the structural weaknesses of Function Point metrics.

## INTRODUCTION

Function Point metrics, developed by Allan Albrecht of IBM, were first published in 1979 and revised in 1984. Also in 1984, the International Function Point Users Group (IFPUG) was set up to clarify the rules, set standards and promote their use and evolution.

Function Point metrics provide a standardized method for measuring the various functions of a software application. Function Point metrics measure functionality from the user's point of view, that is, on the basis of what the user requests and receives in return. As a method for measuring the deliverables of the software development process, Function Point metrics have generated considerable interest in software engineering circles, where this measurement process is perceived to be a step in the right direction.

The *Function Point Counting Practices Manual, Release 3.0*, published by IFPUG in April 1990, is the latest official guide to the Function Point counting rules and our analysis is based on this release. The manual represents the IFPUG consensus of Function Point counting rules and is based on previous documents that constituted more or less a collection of many interpretations of the rules[1,2].

The motivation for this research was to assess whether Function Point metrics had the structural strength to become a lasting reference measurement system. Could it be used as a reference in combination with other measurement methods to analyze other aspects of our software products and processes?

The research object was the measurement method itself embedded within the Function Point metrics. The research purpose was to evaluate the measurement method and the perspective taken was that of the researcher: an analysis of the principles and mathematical foundations of this measurement methodology. The domain of the research was limited to the identification and use of scales in the method.

The primary focus is on the use of scales: how each type of scale (nominal, ordinal, interval and ratio) is used and transformed in the various steps of the method. In conclusion, one should be cautious in the interpretation of Function Point counts: as a result of mixed scales, it should be used as an indicator rather than a measure.

## FUNCTION POINT MEASUREMENT MODEL

The Function Point count is obtained through measurements of the application from two distinct perspectives (Fig. 1):

A) The functional size, calculated through the measurement of *each individual function*. We will refer to this as the measurement process.

B) The value adjustment factor, calculated through the measurement of the *application as a whole*. This factor measures the environment and processing complexity of the application. We will refer to this as the adjustment process.

The value adjustment factor in B) adjusts the functional size determined in A) by +/-35% to produce the Adjusted Function Points. We will refer to Fig. 1 as the FP measurement model.

### A) Functional Size

The measurement process is further decomposed into two processes, the data count and the transaction count (Fig. 2). These data and transaction counts are themselves complex processes. The output of the data measurement process produces both the data counts and the lists of files and data elements to be used as input to the transaction measurement process. The data points counted and the transaction points counted are then added to obtain the Unadjusted Function Points.

The *data measurement process* is further decomposed into six subprocesses (Fig. 3).

D1) The first subprocess is the selection of the data approach. As input to this subprocess, we have listed the three historical official approaches, Albrecht 79, Albrecht 84 and IFPUG 90, the last being the current standard. A fourth, data modeling, is currently being investigated, but is not yet officially recognized. In addition to these are proprietory approaches, such as Symons' and Jones'. Throughout this article we will only refer to the current offical standard, IFPUG 90.

D2) The second subprocess takes as input the application/project documentation and the data approach selected to identify the user identifiable groups of logically related data. It produces as output the list of logical files[1].

D3) This subprocess analyzes the border between the application/project being measured and either external applications or the user domain in order to classify the logical files into internal files or external files. The output of this process consists of sublists of Internal Logical Files

[1] Readers familiar with the Entity/Relations (E/R) model, should use the following equivalent: "Entity" for files, "Attributes" for data elements and "Relationships" for records.

and External Logical Files.

D4) The fourth subprocess counts the actual number of Data Element Types (DET[2]) and Record Element Types (RET[3]) within each logical File Type Referenced (FTR[4]). Here for the sake of clarity, we will use the term "data" to refer to the DET, and "record" to refer to the RET, as defined in IFPUG 90.

D5) The fifth subprocess applies the Function Point algorithm with the following five inputs: sublists of files, data counts, records counts, data functional complexity formula and data weights. The output of this process is a list of points for all logical files.

D6) The last data measurement subprocess is the addition of all points to produce the unadjusted data Function Points, or the data count referred to in Fig. 2.

The *transaction measurement process* is also decomposed into six subprocesses (Fig. 4).

T1) The first subprocess takes as input the data approach selected in subprocess D1) and the application/project documentation to identify the user's visible functions. This will produce the lists of External Inputs, External Outputs and External Inquiries. Here for the sake of clarity, we will refer to these as "inputs", "outputs" and "inquiries".

T2) The second subprocess takes these lists of functions and the model of Functions/Transactions to determine the list of countable transactions.

T3) In the third subprocess the logical files and the data identified in D2) and D4) are applied to the list of transactions to determine two sets of counts: the number of files and the number of data referenced for each transaction.

T4) The fourth subprocess applies the Function Point algorithm with the following five inputs: the list of transactions, files counts, data counts, functional complexity formula and transaction weights. The output of this process is a preliminary list of points for all transactions.

T5) The fifth subprocess selects for each inquiry function the greater of the transaction input or output counts, and eliminates the lesser counts to produce the final transaction counts.

[2] Data Element Type (DET): A unique occurrence of data, also referred to as a data element, variable or field (IFPUG 1990, Glossary)

[3] Record Element Type (RET): A unique record format within an Internal Logical File or External Interface (IFPUG 1990, Glossary).

[4] File Types Referenced (FTR): The number of Internal Logical Files or External Interfaces read, created or updated by a function type (IFPUG 1990, Glossary).

T6) The last transaction measurement subprocess is the addition of all points to produce the unadjusted transaction Function Points, or the transaction count referred to in Fig. 2.

The addition of the data points D6) to the transactions points T6) produces the total Unadjusted Function Points and give the Functional Size (Fig. 1) of the application.

### B- Value Adjustment Factor

To determine the environment and processing complexity of an application, the adjustment measurement process is decomposed into five subprocesses (Fig. 5).

V1) The first step takes as input the application/project documentation and the criteria definitions for each of the General System Characteristics (GSC). Depending on the specifics of each GSC, this subprocess produces one or multiple criteria.

V2) The second subprocess assigns a score to the GSC based on the GSC score/criteria table.

V3) In the third subprocess, a degree of influence is assigned to each score, based on a score/degree of influence table. This is currently a one-to-one relationship, and it is exactly the same for each characteristic.

V4) The fourth subprocess is the addition of all the degrees of influence for each of the 14 characteristics.

V5) The last factor measurement process is the application of the factor formula (0.65 + 0.01*N)) to obtain the Value Adjustment Factor.

## Survey of the Literature

Most publications on Function Points have addressed such issues as:

1- Comparisons with other software metrics based on lines of code (Halstead and McCabe)[3];
2- Accuracy of estimates[4];
3- Inter-rater reliability[5,6,7,8];
4- Productivity measurements[4,9-12].

Only a few authors have reviewed Function Point methodology and they have identified some of its weaknesses[13-16]. On the other hand, the work of the IFPUG group has focused on clarifying the rules and guidelines for all the *Identify* and *Classify* subprocesses in our FP measurement model. They have also considerably improved the definitions for the assignment of scores for the General System Characteristics. Similarly IFPUG has recognized that the data modeling approach[13], pending further analysis and scheduled for future release, would provide added rigor. In our opinion, however, the data modeling approach will address only the *Identify* and *Classify* subprocesses, and not the measurement foundation of the process.

For the purposes of this analysis, we make the assumption that, with current rules and guidelines, the *Identify* and *Classify* subprocesses produce clearly identified data functions, transaction functions and GSC scores. We, therefore, set aside all issues that could arise between the mapping of an application/project and the selected data approach for the Function Point (FP) Model. We are now free to look exclusively at the *Count* and *Algorithmic* subprocesses of the FP measurement model.

**Overview of the measurement scales.**

This analysis focuses on two key aspects of any measurement process: the identification and the analysis of the use of the different scales within the various steps of the FP measurement model with reference to the mathematical operations allowed for each type of scale. The mathematical operations permitted for each scale are summarized in Table 1.

Scales: Admissible Tansformations[17]

| Scale type | Represents | Admissible transformations | Description of operations | Examples |
|---|---|---|---|---|
| Nominal | (R,=) | f unique | name, distinguish | colours, shapes |
| Ordinal | (R,> =) | f strictly increasing monotonic function | rank, order | first, second, third |
| Interval | (R,> =,+) | f(x)= ax + b, a>0 | add | age, distance |
| Ratio | (R,> =,+) | f(x)= ax, a>0 | multiply, divide | price |
| Absolute | (R,> =,+) | f(x)= x | multiply, divide | absolute temperature: Kelvin degrees |

Table 1

1- Nominal scale: this scale is used to name objects or events; this scale is used for identification purposes only, and has no quantitative implications associated with it.

2- Ordinal scale: this scale is used to order or rank items, based on a criterion that can be subjective or, preferably, objective.

3- Interval scale: this scale (also called cardinal scale) is used to determine the difference between the ranks; it is continuous between two end-points, neither of which is necessarily fixed.

4- Ratio scale: with this scale, the items can be distinguished and ranked, and the differences between ranks measured.

5- Absolute scale: in addition to the properties of the ratio scale, the absolute scale has a unique origin from which to begin the measurement.

## Data Measurement Process Model

We examine the uses of the measurement scales within the *data measurement process* of Fig. 3). In the FP methodology, the algorithm for the calculation of points is described as a three-step calculation: a file is first classified as of *low*, *average* or *high* complexity, then a number of points is assigned depending on its level of complexity and then, the points are added.

For example, an application has the following three Internal Logical Files (ILF):

Logical File 1: has 1 record and 5 data elements,
Logical File 2: has 3 records and 21 data elements,
Logical File 3: has 6 records and 26 data elements,

In the FP methodology file 1 would be classified as *low* with a weight of 7, file 2 as *average* with a weight of 10 and file 3 as *high* with a weight of 15. The addition of their weights will give a total number of 32 points for these three Internal Logical Files.

This is however an oversimplification. In order to identify the types of scales and analyze their uses in this measurement process it must be broken down further as follows:

1- In subprocesses D2) & D4) of Fig. 3, three different object types are counted: data (DET), records (RET) and files (FTR).

It should be noted that these three types of objects are not independent, but are organized in a hierarichal way (Fig. 6). A record (RET) is composed of one or many data (DET), and a file (FTR) is composed of one or many records (RET). They represent three different levels of abstraction: the RET represents a structure of DETs, and the FTR, a structure of RETs. These three object types also have different semantics and properties.

Subprocesses D2) & D4) count the above as three different objects types and the results of each addition can be used on an interval scale. For example, we can say that a report with 21 data elements has 3 times as many data elements as a report with 7, similarly a report which accesses 3 files has 3 times as many files in access as a report with only one.

2) Let us now examine Subprocess D5) in Fig.3. We refer to it as the *data algorithm.*

2a) The first step (Table 2) of this data algorithm consists in positioning the results of the previous additions of data and records into ranges.

Ranges

| Objects | Range 1 | Range 2 | Range 3 |
|---|---|---|---|
| Data | 1-19 | 20-50 | 51+ |
| Record | 1 | 2-5 | 6+ |

Table 2

It must be noted that the results obtained in the mapping on these ranges cannot be added, but only ranked. Consequently they do not qualify as an interval scale: they are merely a ranking process, and therefore constitute strictly an ordinal scale. From this point on, it can be said only that Rank 2 is bigger than Rank 1, and not that it is twice as big, or has twice as many elements. It must be observed that we have lost mathematical flexibility in changing scales.

Let us label the different ordinal ranges of the two different object types, data and records, with the following rank identifiers (Table 4).

Rank Identifiers

| Objects | Rank 1: | Rank 2: | Rank 3: |
|---|---|---|---|
| Data | D1 | D2 | D3 |
| Record | R1 | R2 | R3 |

Table 3

2b) The next step (Table 4) is much more complex from a measurement perspective: the ranks of two distinct types of objects (data and records) are taken as parameters of a function using two different ordinal scales. This can be represented in mathematical notations as a function with two arguments fn(Di,Rj). While this is a normal process when building indices based on a variety of factors, this is not a standard measurement process.

Nominal identifiers

| DATA/ RECORDS | Rank D1 | Rank D2 | Rank D3 |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Rank R1 | D1,R1 | D2,R1 | D3,R1 |
| Rank R2 | D1,R2 | D2,R2 | D3,R2 |
| Rank R3 | D1,R3 | D2,R3 | D3,R3 |

Table 4

This transformation does not produce results of an ordinal nature: while it could be argued that cell (D1,R1) is smaller than cell (D3,R3), we cannot conclude that cell (D1,R2) is either equal to or greater than cell (D3,R1). This is a nominal scale, with a loss of measurement information from the previous individual ordinal scales of the parameters of the algorithm.

2c) The next step (Table 5) assigns a rank (Low, Average and High) depending on the position on the matrix:

- above the diagonal: Low
- on the diagonal : Average
- under the diagonal: High

Functional Complexity Matrix

| DATA/ RECORDS | Rank D1 | Rank D2 | Rank D3 |
|---|---|---|---|
| Rank R1 | Low | Low | Average |
| Rank R2 | Low | Average | High |
| Rank R3 | Average | High | High |

Table 5

This is called *functional complexity assignment* in FP methodology. In our opinion, the terms used (Low, Average and High) are misleading and hide some of the deficiencies of the method. It leads to the belief that an ordinal scale is being used at this stage of the measurement process. Furthermore, it does not explain what is understood by "complexity", and we are not aware of impact studies conducted to verify either this classification or the classification process itself.

It should be noted that the diagonal does not correspond to usual mathematical notation. It does not express symmetry between the two axes, but it does express a ranking of simultaneous increases on both axes (on an ordinal scale, with nonregular and nonsimilar ranges). Its purpose seems rather to express an equivalence of the rankings of the two axes when they are combined! For example, the smallest ranking on one axis combined with the highest ranking on the other axis is ranked similarly to the inverse (For example, (D1,R3) = (D3,R1).

The expected result in FP methdolology is however treated as an ordinal number. For experienced programmers, this way of defining and quantifying the complexity of files seems to be intuitively valid; however, *this is not a strict measurement process equivalence*!

2d) The next step (Table 6) consists in assigning a weight to the rank, based on whether it is an internal logical file or an external logical file. Since both the nominal and the ordinal scales were useless in a quantitative measurement process, a set of weights was needed for measuring (and/or adding) the different types of components within the nominal scale. This required a transformation into a ratio scale by assigning different weights according to (in our opinion) the "perceived different user values" of the function types.

Weights

| **File types** | **Rank 1: Low** | **Rank 2: Average** | **Rank 3: High** |
|---|---|---|---|
| Internal | 7 | 10 | 15 |
| External | 5 | 7 | 10 |

Table 6

These weights by themselves represent relative absolute numbers from 3 to 15, depending on the function type, and the end number obtained in this step would appear superficially to be a valid number on a ratio scale. Their true purpose is to assign numbers to subsets of ranks (not intervals, because they are not equal) that will permit their use in a ratio process: for example, an external file whose position on the matrix is under the diagonal will have a weight of 10, twice as big as another external file which is positioned above the diagonal which is assigned a weight of 5. Within each of the three sets, a different weight is assigned.

Finally, provided that the results of the previous steps were valid, the results of the addition in the previous data subprocess would provide results on a ratio scale. This is an artifice to transform an ordinal scale result into a ratio result. Even though this process again appears to be intuitively valid because of the weights assigned by expert developers, this is not a sound mathematical transformation.

Even if we had made the hypothesis that the weights were valid (we will comment on this in the paragraphs that follow), the mathematical operations are not. An ordinal number (under the hypothesis that the scale transpositions from the previous step were valid) cannot mathematically be transformed either into an interval or a ratio scale.

Symons and others had already noted the subjectiveness of the weights determined by debate and trial and had questioned their transportability to environments other than the one from which they had been derived (Information Systems at IBM in 1979 and 1984). The true significance of these weights is hard to determine. Although they seem to measure the complexity of the function (based of the terminology used: Low, Average and High) based on a function with two parameters, the word complexity itself is not defined. It seems rather an attempt to measure the "relative values" of the function types. The criteria used to determine these weights are not known. Were they based on users' criteria (who participated, users? if so, which types of users? or information systems experts in lieu of users?), or, more likely, on an assessment by information systems experts.

Another hypothesis is that it establishes an equivalence of the functionality with the amount of work effort required to deliver these types of functionalities! What this does is the following: it transforms the desired measurement of *size*, based initially on the number of data and records, into an assessment of "*values*", within value scales. This might qualify as a semantic transposition to arrive at a ratio value.

3) In the last Subprocess D6) in Fig. 3, all the points obtained per function type are added per function type. Under the hypothesis that the numbers obtained were valid ratio numbers, the resulting number would also be a valid ratio number.

### Transaction and Adjustment Measurement Processes

All of the above comments on the uses of the scales apply to the *transaction measurement process* described in Fig. 4 and to the *adjustement measurement process* described in Fig. 5[18].

4) As a last step of the FP methodology, all the points are then added together, whether they come from Internal Logical Files, External Logical Files, Inputs, Outputs or Inquiries. It must be observed that this has been made possible only by the assignment of the weights to transform five different types of objects into one single object of a different type and of an unspecified nature, a *Function Point*. Again this is done only through semantic transformations, and not through a strict measurement process with the proper use of the different types of scales.

The end results (Unadjusted and Adjusted Function Points) become, therefore, very difficult to interpret: there are so many dimensions involved and so many uses of different types of scales that the end measure, which might look rather simple and reasonable, is, in fact, a pot-pourri that might not have correct mathematical meaning.

It is doubtful that such a comparison, and ratio, would be valid because of the mixing of apples and oranges that has occurred within the Function Point methodology. Therefore, it would be more appropriate to call the end count of function points an "index", rather than a "size" measure.

## Conclusions and Recommendations

The results should then be taken for what they are: indices that can be helpful in indicating trends only. The Function Point methodology does not derive from a well-defined and proven theory: it is entirely empirically based on expert opinion. The Function Point count is derived from a set of steps, rules and formulas: it is an algorithmic metric, and therefore has the problems inherent in any algorithmic (or synthetic) metrics system:

- Algorithmic metrics are difficult to interpret: the reasons for the assignments of specific values (weights) are not clear;

- The value of the output of the formula is useful only if the formula is based on a solid theory, such as physics. But this is not the case with these software metrics.

- To control the measurement process, it should be a requirement to store the basic values produced by the measurement process of the metric algorithm, in order to conduct further studies. Strictly as a measurement process, these additions of the basic components (data and records) would be a much more mathematically meaningful size measure than the results of inappropriate mathematical transformations and the semantic transpositions of scale components.

Two other areas of the measurement process should also be investigated further: the different levels of granularity of the basic components (data, records and files), as well as the manipulation of the five different function types (internal files, external files, inputs, outputs and inquiries). Only with an adequate measurement process will statistical analysis on Function Point counts be meaningful.

**************************************************

This research was carried out jointly by Montreal Trust, École Polytechnique de Montréal and the Organizational Research Division of the Canadian Workplace Automation Research Centre/ Communications Canada. This joint project was made possible under the terms of the National Scientific Staff Exchange Program. The opinions expressed in this report are solely those of the authors.

**************************************************

## References

1- IBM CIS & a Guidelines 313, "AD/M Productivity Measurement and Estimate Validation", Nov. 1984.

2- International Function Point Users Group (IFPUG), "Function Point Counting Practices Manual, Release 3.0", IFPUG, Westerville, Ohio, April 1990.

3- Albrecht, Allan J., Gaffney John E., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Trans. on Soft. Eng., Vol. SE-9, no. 6, Nov. 1983.

4- Kemerer, Chris F., "An Empirical Validation of Software Cost Estimation Models", Communications of the ACM, Vol. 30, no. 5, May 1987.

5- Kemerer, Chris F. "Function Point Measurement Reliability: A Field Experiment", IFPUG Fall Conference, San Antonio, Texas, October 1990.

6- Low, Graham C., Jeffery D. Ross, "Function Points in the Estimation and Evaluation of the Software Process", IEEE Trans. on Soft. Eng., Vol. 16, no. 1, Jan. 1990.

7- Rudoph, "Productivity in Computer Application Development", Dep. Management Studies, Univ. Auckland, Australia, 1983.

8- Rudolph, E.E., "Precision of Function Point Counts", IFPUG Spring Conference, SanDiego, CA, April 1989.
Abran, Alain, "Function Points: Research Report 91-02, Montreal trust, Montreal, 1991.

9- Behrens, Charles A., "Measuring the Productivity of Compsuter Systems Development Activities with Function Points", IEEE Trans. on Soft. Eng., Vol. SE-9, no. 6, NOv. 1989.

10- Emrick, Ronald D. "Software Development Productivity - Second Industry Survey", IFPUG Spring Conference, Dallas, May, 1988.

11- Emrick, Ronald D. "Further Analysis - Software Development Productivity - Second Industry Survey", IFPUG Fall Conference, Montréal, Sept. 1988.

12- Gaffney, John E., "The Impact on Software Development Costs of Using HOL's", IEEE Trans. on Soft. Eng., Vol, SE-12, no. 3, March, 1986.

13- Desharnais, Jean-Marc, "Analyse statistique de la productivité des projets de développement en informatique à partir de la technique des points de fonction", maîtrise informatique de gestion, Université du Québec à Montréal, Déc. 1988.

14- Desharnais, Jean-Marc, "Adjustment Model for Function Points Scope Factors - A Statistical Study", IFPUG Spring conf., Florida, April 1990.

15- Jones, Capers, "Feature Points (Function Point Logic for Real Time and System Software)", IFPUG Fall 1988 Conference, Montreal, Québec, Oct. 1988.

16- Symons, Charles R., "Function Points Analysis: Difficulties and Improvements, IEEE Trans. on Soft. Eng., Vol. SE-14, no. 1, January 1988.

17- Zuse, Horst, "Software Metrics - Methods to Investigate and Evaluate Software Complexity Measures", 2nd Annual Oregon Workshop on Software Metrics, Portland, Oregon, March 19, 1990.

18- Abran, Alain, "Function Points: Research Report 91-02, Montreal Trust, Montreal, 1991.

## FUNCTION POINT MEASUREMENT MODEL

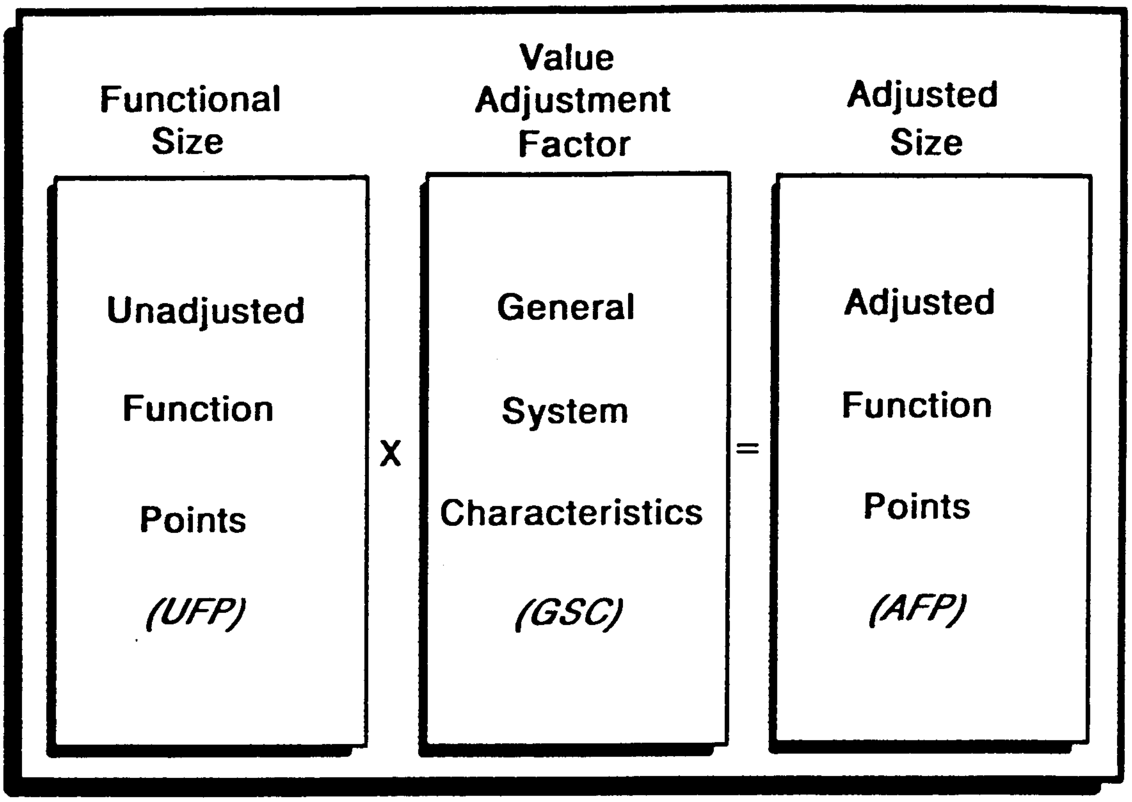

Functional Size
Value Adjustment Factor
Adjusted Size
Unadjusted Function Points (UFP)
X
General System Characteristics (GSC)
=
Adjusted Function Points (AFP)

**FIGURE 1**

## FUNCTION POINT MEASUREMENT PROCESS

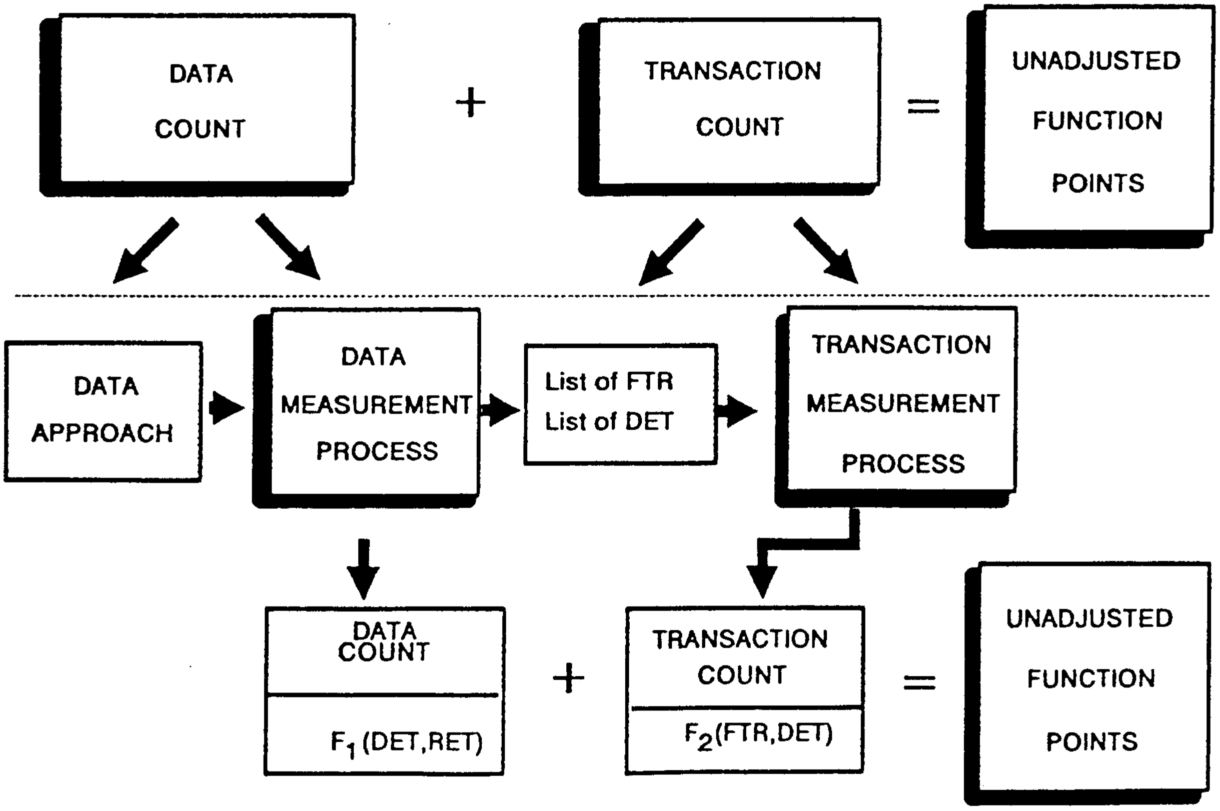


**FIGURE 2**

# DATA MEASUREMENT PROCESS

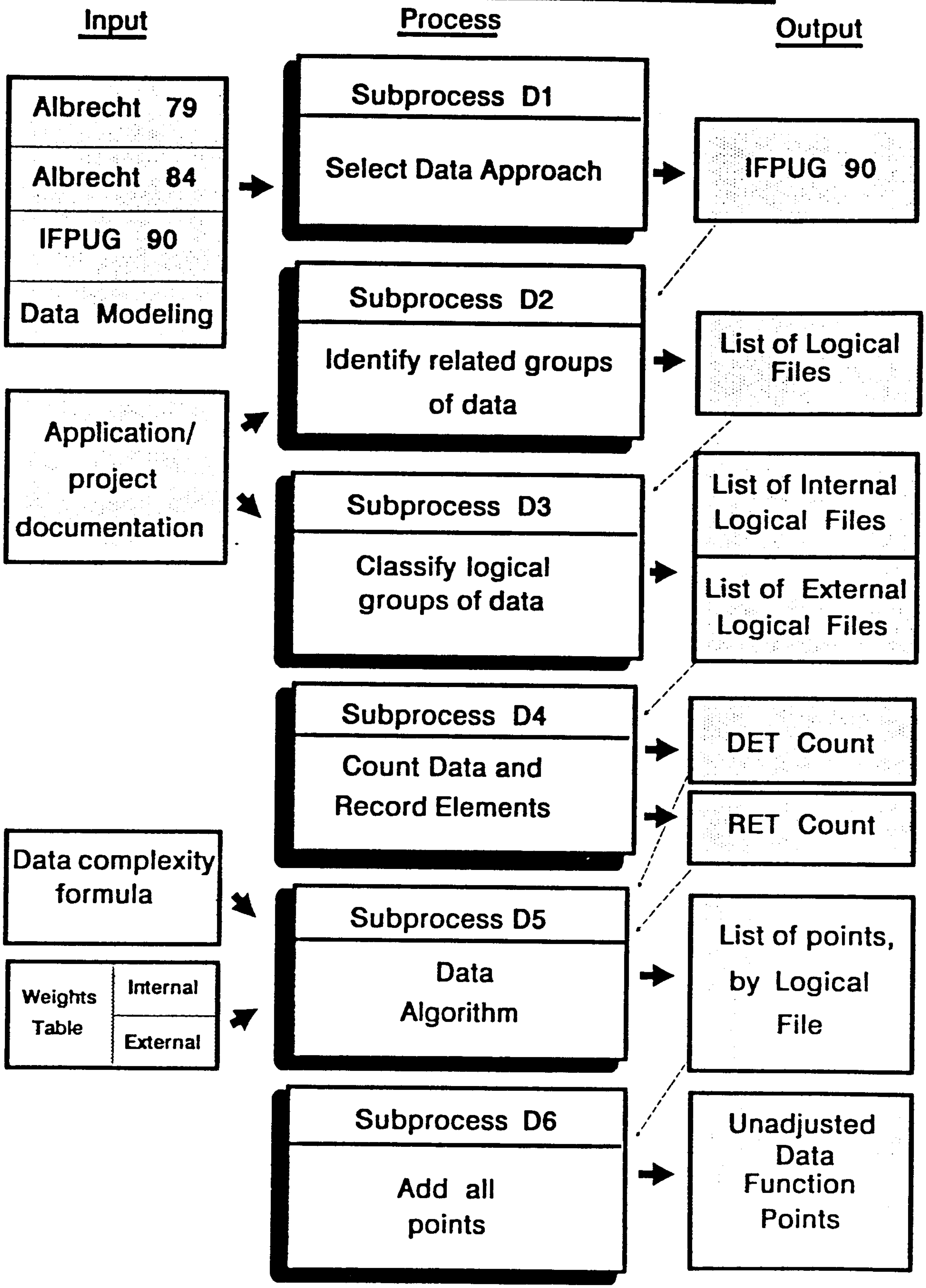


FIGURE 3

# TRANSACTION MEASUREMENT PROCESS

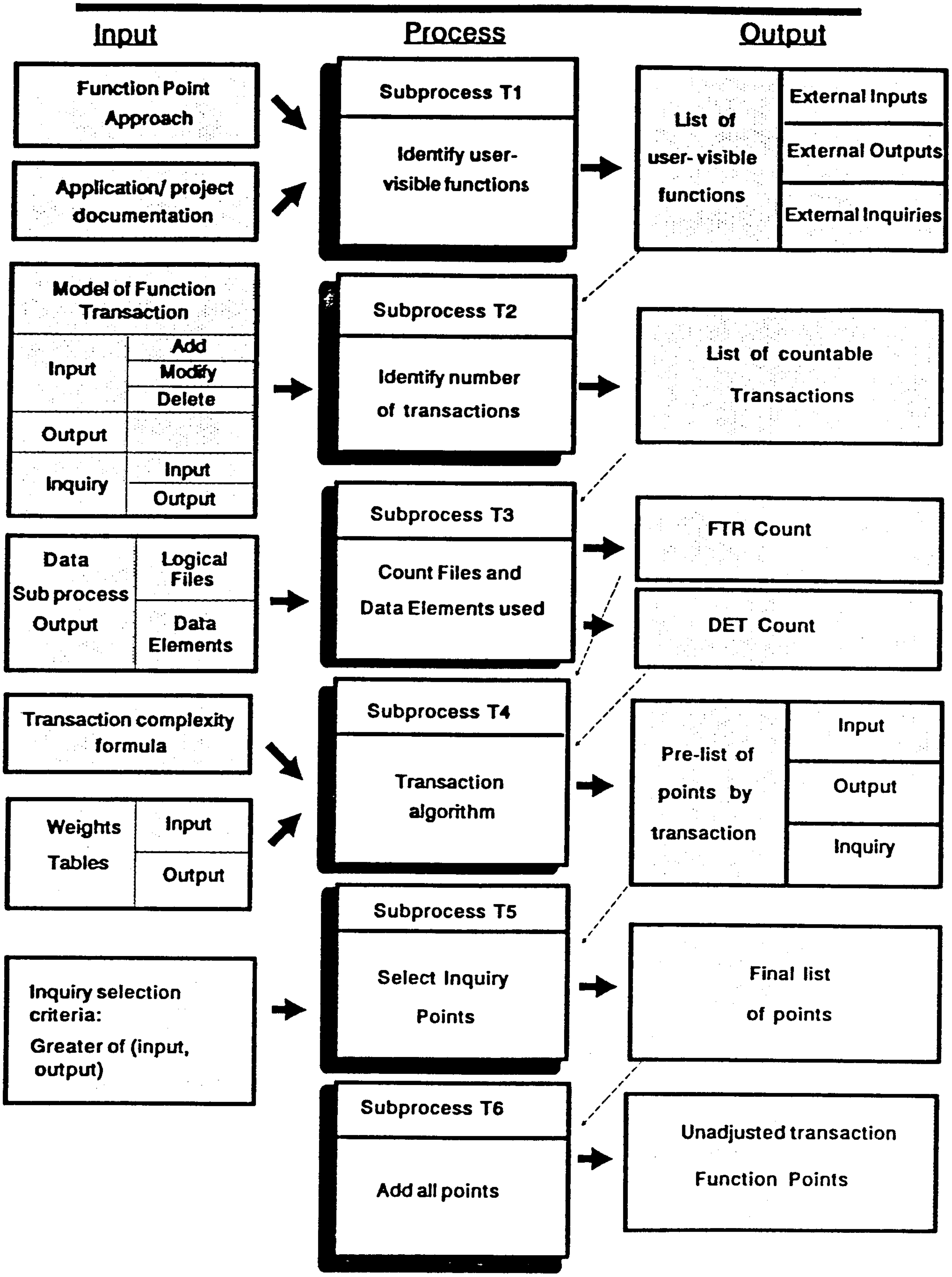
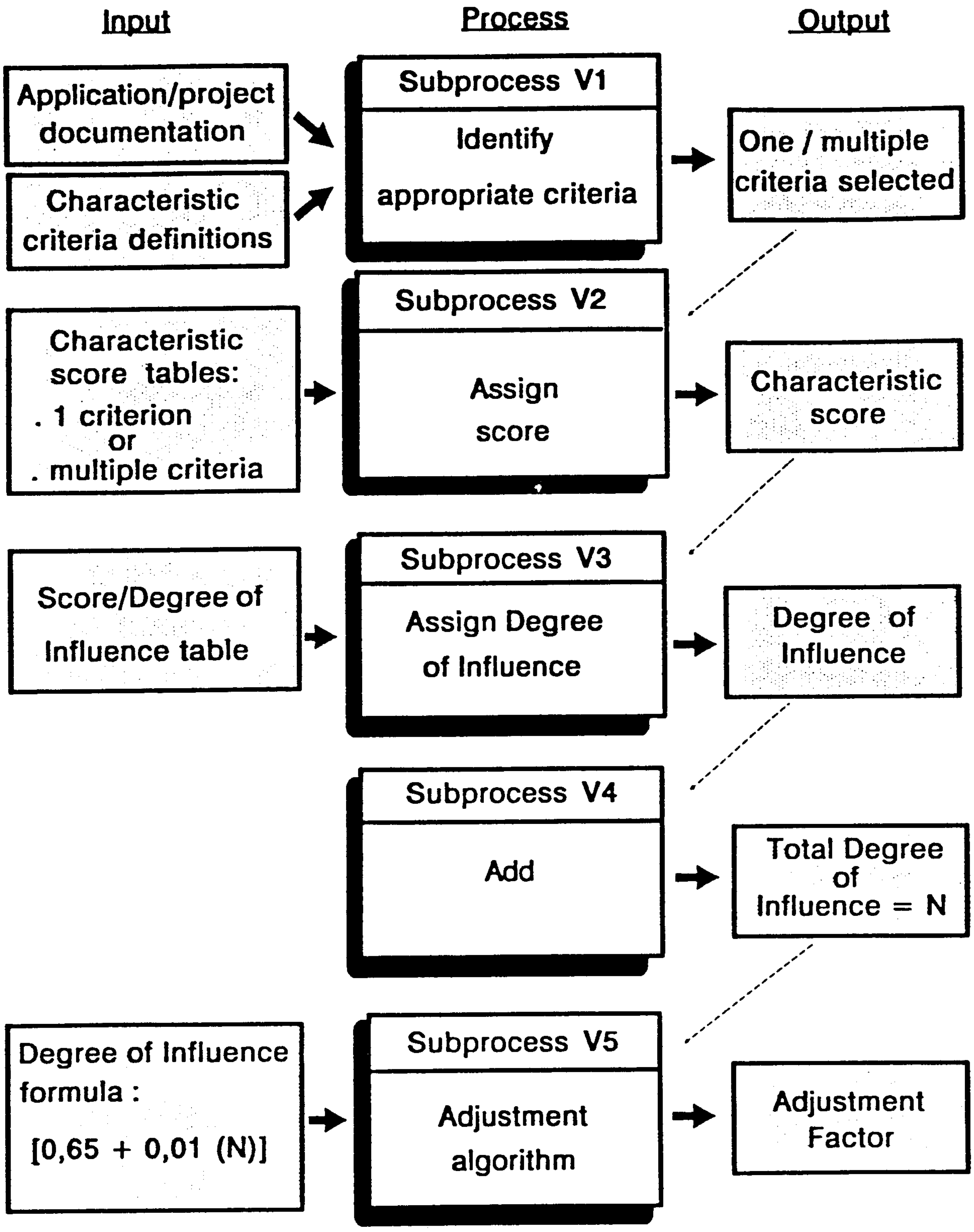


FIGURE 4

# ADJUSTMENT MEASUREMENT PROCESS



Input

Process

Output

Application/project documentation

Characteristic criteria definitions

Subprocess V1

Identify appropriate criteria

One / multiple criteria selected

Characteristic score tables:
. 1 criterion
or
. multiple criteria

Subprocess V2

Assign score

Characteristic score

Score/Degree of Influence table

Subprocess V3

Assign Degree of Influence

Degree of Influence

Subprocess V4

Add

Total Degree of Influence = N

Degree of Influence formula :

[0,65 + 0,01 (N)]

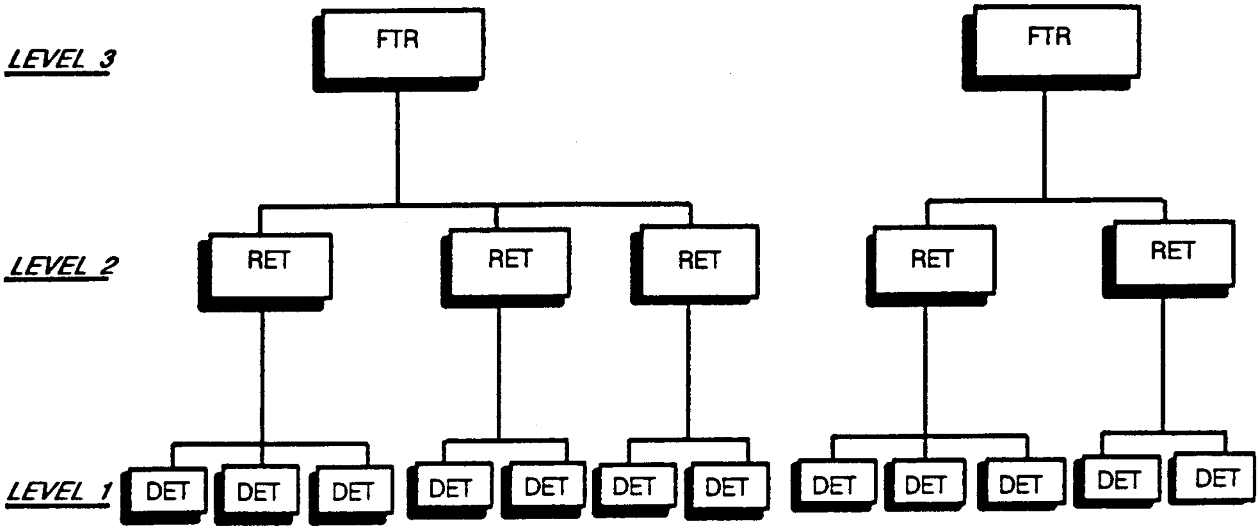
Subprocess V5

Adjustment algorithm

Adjustment Factor

Figure 5

# DATA LEVELS



DET : Data Element Type
RET : Record Element Type
FTR : File Type Referenced

**FIGURE 6**