A COMPARISON OF ALBRECHT'S FUNCTION POINT AND SYMONS' MARK II METRICS

Raimo Rask
University of Joensuu

Petteri Laamanen
University of Joensuu

Kalle Lyytinen
University of Jyvaskyla

A COMPARISON OF ALBRECHT'S FUNCTION POINT AND SYMONS' MARK II METRICS

Raimo Rask
Petteri Laamanen
Department of Computer Sciences
University of Joensuu

Kalle Lyytinen
Department of Computer Science and Information Systems
University of Jyväskylä

ABSTRACT

Software system size provides a basis for software cost estimation and management during software development. The most widely used product size metric for the specification level is Albrecht's Function Point Analysis (FPA). Symons has suggested an alternative for this metric called the Mark II metric. This metric is simpler, more easily scalable, and better takes into account the complexity of internal processing. Moreover, it suggests different size values in cases where the measured systems differ in terms of system interfaces. One problem in using these metrics has been that there are no tools that can be used to calculate them during the specification phase. To alleviate this we demonstrate how these metrics can be automatically calculated from Structured Analysis descriptions. Another problem has been that there are no reliable comparisons of these metrics based on sufficient statistical samples of system size measures. In this paper we address this problem by carrying out preliminary comparisons of these metrics. The analysis is based on a randomly generated statistical sample of dataflow diagrams. These diagrams are automatically analyzed using our prototype measurement system using both FPA and the Mark II metric. The statistical analysis of the results shows that Mark II correlates reasonably well with Function Points if some adjustments are done to the Mark II metric. In line with Symons's discussion our analysis points out that the size of correlation depends on the measured system type. Our results also show that we can derive useful size metrics for higher level specifications and that these metrics can be easily automated in CASE tools. Because the obtained results are based on simulation, in the future they must be corroborated with real life industrial data.

1. INTRODUCTION

In general, software metrics can be classified into process metrics and product metrics (Conte, Dunsmore and Shen 1986; Hunter 1990). *Process metrics* quantify attributes of the development process and the development environment. They are often concerned with development resources. *Product metrics* measure attributes of the software product. They focus on software requirements, design, or source code. If we have a generally accepted product (size) metric P we can estimate the process metric C in the form of *costs*, i.e., *work effort* (man-months) with the following linear or non-linear equations:

(1) $C = K + M * P$
(2) $C = K + M * P^E$

Here K and M denote environment dependent constants (DeMarco 1982; Conte, Dunsmore and Shen 1986; Londeix 1987). Normally, the product size metrics (P) in these equations have been based on lines of code (LOC) estimates. Though LOC measures are not universally accepted as a reliable basis to derive size, time and cost estimates, they are the most widely used and discussed (Albrecht and Gaffney 1983). The more fundamental problem is that usually LOC figures can be derived late in the development cycle, i.e., during the programming and testing phase, which makes them nearly useless from the managerial point of view.

It is widely regarded that it is essential to accurately estimate the cost of the software product in the early phases of development cycle (DeMarco 1982). Therefore, several researchers have proposed new size metrics that measure the functional complexity of the system. Accordingly, these metrics can be used when the functional specification of the system is available, i.e., during the analysis and specification phase. These include Albrecht's Function Points

(Albrecht 1979), Symons' Mark II (Symons 1991), and DeMarco's Function Bang (DeMarco 1982). During the life-cycle approach, it is then possible to dynamically adjust size estimations during each phase so that estimates gradually approach the actual result (DeMarco 1982; Banker, Kauffman and Kumar 1990).

One problem with these metrics so far has been that tools that could count these size metrics automatically from available functional specifications have not been available. Accordingly, their use has been difficult and costly and dependent on scarce expertise. Recently, Computer Assisted Software Engineering tools (CASE) have changed the situation. Using CASE, it is possible to capture functional representations of software systems in a computer readable form. However, there are currently only a few solutions, or algorithms, that can derive size metrics from specifications in CASE tools. In addition, there are few if any tests concerning the accuracy and reliability of such metrics. Therefore, a real research challenge is to develop appropriate tools for size estimation in CASE and to evaluate their reliability and accuracy.

The purpose of this paper is to make some preliminary steps in addressing these research challenges. The paper examines empirically through simulation the correlation between two product size metrics: Function Points (Albrecht 1979) and Mark II (Symons 1988; 1991). These metrics are derived automatically from Structured Analysis (SA) specifications using polynomial algorithms (Rask 1992). By doing so we develop procedures to estimate accurately software sizes in early phases of development that can be implemented in a CASE tool.

The paper is organized as follows. An overview of the examined metrics is presented in section 2. In sections 3 and 4 we discuss the data generation and measurement procedure used. In sections 5 and 6 we summarize our results from the experiments and discuss their implications for further research.

2. DEFINITION OF METRICS

2.1 Function Point Analysis

A widely applied measurement method is Albrecht's *Function Point Analysis* (FPA). A *function point* is defined as one end-user business function. The functions can be organized into the following five groups (Albrecht and Gaffney 1983; Dreger 1989): external inputs, external outputs, external inquiries, logical internal files, and external interface files. To derive the FPA function, we first identify all functions in the design specification and classify them into the five groups. Then we weight each function by its level of complexity into simple, average, or complex.[1] The sum of these weights is called a *total of unadjusted function points* (TUFP). Table 1 shows the possible weight values for each function type.

Table 1. FPA Function Types and the Possible Weights (Based on Dreger 1989)

	Complexity		
Function Type	**Simple**	**Average**	**Complex**
External Input	3	4	6
External Output	4	5	7
Logical Internal File	7	10	15
External Interface File	5	7	10
External Inquiry (Input)	3	4	6
External Inquiry (Output)	4	5	7

Table 2. Classification Factors for User Functions

Function Type	Files Referenced	Data Items Referenced	Logical Record Format/Relationships
External Input	X	X	
External Output	X	X	
Logical Internal File		X	X
External Interface File		X	X
External Inquiry (Input)	X	X	
External Inquiry (Output)	X	X	

The classification and weighting of inputs, outputs, and inquiries depends on the number of files referenced and on the number of data items referenced. Correspondingly, the classification and weighting of logical internal files and external interface files depends on the data items referenced and on the logical record formats of the files or logical relationships between the files. The classification factors are illustrated in Table 2. The exact limits of classification factors can be found in Dreger (1989).

The implementation of a logical system may vary depending on goals and the target hardware/software environment. Because of this, Albrecht has defined fourteen environmental *adjusting factors* to count the *final function point value* (FFPV). This is derived from the following formula:

(3) $\text{FFPV} = \text{TUFP} * (0.65 + 0.01 * \text{TDI})$

where TDI depicts total degree of influence determined by the sum of the adjusting factors of the hardware/software platform.

2.2 Mark II

Symons (1988) gives a new alternative, *Mark II*, for specifying the system size (in his recent book, Symons [1991] uses the name "Mk II Function Point Analysis Method"). This metric contains the following assumptions:

- a system consists of logical input/process/output combinations,
- interfaces on the logical level are treated as any other input or output,
- inquiries are viewed just as any other input/process/output combination, and
- a logical file concept is interpreted at the logical transaction level as an entity, i.e., anything in the real world about which the system provides information.

The task then is to find properties of input, process, and output components of each logical transaction type. To calculate the complexity of the process component, we apply the number of entity types referenced (created, updated, read, or deleted) by the transaction type. For the input and output components, the number of data element types forms the size of the component. The Mark II formula for product size expressed in *unadjusted function points* (UFP) can be written (Symons 1988; 1991):

(4) $\text{UFP} = N_I W_I + N_E W_E + N_O W_O$

where N_I = number of input data element types, W_I = weight of an input data element type, N_E = number of entity-type references, W_E = weight of an entity-type reference, N_O = number of output data element types, and W_O = weight of an output data element type. Based on Albrecht's Function Point Analysis and industrial calibration material, Symons (1988; 1991) defines the following *weights* for Mark II formula:

(5) $\text{UFP1} = 0.44N_I + 1.67N_E + 0.38N_O$ (Symons 1988),

or

(6) $\text{UFP2} = 0.58N_I + 1.66N_E + 0.26N_O$ (Symons 1991).

To count the final size of the software, Symons (1991) proposes to use the technical complexity adjustment (TCA) by adding five general application characteristics to those of Albrecht's environmental adjusting factors:

(7) $\text{TCA} = 0.65 + C * \text{TDI}$

where TDI depicts total degree of influence determined by the sum of the (extended) adjusting factors and the current industry-average value of "C" is 0.005. The final system size, *function point index* (FPI), can be obtained:

(8) $\text{FPI} = \text{UFP} * \text{TCA}$.

2.3 Comparison of the Two Metrics

Clearly, Symons' Mark II metric (UFP) is easier to understand and calculate when compared with Albrecht's metric (TUFP). Mark II contains fewer function types and the functions need not to be classified and weighted. Moreover, the principles to calculate the final function point value (FFPV) and the function point index (FPI) do not differ significantly.

Symons (1991) investigates the conversion from Albrecht's FPA to Mark II by remarking that it deals with the problem of the product size as measured by unadjusted function points. The input and output parts of logical transactions according to the Mark II view are roughly equivalent to the input and output, and the input/output parts of inquiries of the Albrecht view. The Mark II handles the processing complexity within transactions by counting each data entity once every time it is referenced in a logical transaction. In Albrecht's FPA we count each internal logical file and external interface file once, irrespective of how often they are used in transactions.

According to the Symons' own comparisons (1988; 1991), Albrecht's metric and Mark II do not always correlate very well, especially in situations where the internal complexity of the system varies. Therefore, he argues that it is not possible to forecast accurately the Mark II size value from Albrecht's FPA size because the scatter diagram is nearly random. Moreover, the Mark II method seems to give systematically higher UFP scores as a function of system size compared with Albrecht's method.

To corroborate some of Symons' claims, we automated the calculation of both metrics from Yourdon's (1989) SA descriptions. To generate statistically significant test material, we produced randomized SA descriptions. We have also used the same environment to compare Albrecht's Function Points with DeMarco's (1982) Function Bang metric (Rask, Laamanen and Lyytinen 1992).

3. AUTOMATION OF METRICS CALCULATION

3.1 Architecture of the Prototype System

We have developed a prototype system, JoUCASE (Joensuu University CASE), to analyze SA descriptions and calculate product size metrics (Laamanen and Rask 1991). This has been implemented using a Turbo Pascal for Windows development environment and currently it runs under MS Windows 3.0.

The tools of the system cover editors for data dictionary, dataflow diagrams (DFD), and entity-relationship diagrams (ERD). It offers the notations suggested by Yourdon with three notable exceptions. It uses distinct symbols to denote an external application, a primitive process, and an inquiry. This makes it possible for count software sizes more accurately. The new symbols to dataflow diagrams are shown in Figure 1.

To check the correctness of SA specifications the environment applies the elementary validity rules of DeMarco (1979), the balancing rules of Yourdon, and consistency and completeness checks of Cowan (1990). In all, we have implemented 35 rules to guarantee the validity of the SA specification for software size estimation purposes (Laamanen and Rask 1991). Moreover, the environment offers three size estimation tools including Function Point Analyzer, Function Bang Analyzer and Mark II Analyzer.

3.2 Function Point Analyzer

Figures 2, 3 and 4 and Table 3 exhibit a structured specification produced by the editors of our prototype system. We use this specification to illustrate how Function Point Analyzer calculates the total unadjusted function point (TUFP) value.

Figure 2 depicts a context diagram of our sample specification. Outside the system boundary, we have four terminators T1,...,T4 and an external data store F1. Terminator T1 has one external input *a*. Terminator T2 has one external input *b*. External application T4 has one external interface input *f*. An external output *c* is produced to terminator T2. Between terminator T3 and the system, we have one inquiry with an input part *d* and an output part *e*. Finally, data store F1 acts as an external interface file to the external application T4.

In Figure 3 the context diagram is further decomposed. We divide the system into six subprocesses P1, P2, P3, P4, P5, P6, and three data stores F2, F3, F4. Dataflow *a* in the context diagram is decomposed into two subdataflows *x* and *u*. Process P5 transfers the external input *f* as a transaction file from the external application T4. Process P6 sends interface file (external data store in Figure 2) F1 to the external application. Dataflow *g* describes a shared file connection between the system and the external application T4.

The ERD for our example system is shown in Figure 4. Each entity in an ERD must correspond to some data store in a DFD. Detailed specifications of the dataflows and data stores are stored in the data dictionary (Table 3) following the notation given in Yourdon.

For a logical file, we use a notation <F,K>, where F denotes a data store and K denotes its key. A new logical record format for a logical file is found, if its referencing flow has the same key as the logical file, and the name of the referencing flow differs from the names of the other flows referencing to it. The unnamed dataflows are interpreted to have the same contents as the file they are referencing.

Each logical file can contain one or more logical record format. The logical record formats are inferred from the dataflows referencing to the data stores. If two dataflows refer to the same data store, and they have the same key, then they refer to the same logical file. But if the names of these two dataflows differ, then we have two different logical record formats. This is identified by a common key of the referencing dataflows. For any logical file <F,K>, the number of logical relationships is the same as the number of logical relationships of the data store F. Finally, the number of referenced data items in a logical file is counted as the cardinality of the union of the sets of elementary data items in all of the logical record formats in that file.

Our example has four *external inputs* at the lowest level DFD. Input flow *x* references one logical file <F4,x1>, flow *u* two logical files <F2,z1> and <F3,y1>, flow *b* one logical file <F4,x1>, and finally flow *f* one logical file <F3,y1>. The sizes of input dataflows *x, u, b* and *f* can be counted using Table 3 to be 5, 9, 2 and 3 elementary data items, respectively. Using sheets in Dreger, the unadjusted function points for these input dataflows will be 3, 4, 3 and 3, correspondingly (3 = simple, 4 = average).

The *external output* dataflow *c* references one logical file <F4,x1>. From the data dictionary, we can calculate it to consist of four elementary data items. Thus, the unadjusted function points for that output dataflow will be 4 (simple).

In the context diagram (Figure 2), we have further one *external inquiry* between the system and terminator T3. The inquiry contains an input part *d* and an output part *e*. From Figure 3 and Table 3 we can infer that the inquiry references to logical files <F2, z1>, <F3,y1>, and <F4,x2>. The data dictionary of Table 3 tells the size of the input part *d* to be one and the size of the output part *e* to be six elementary data items. The complexity of the input part is thus four (average) and the complexity of the output part is five (average). The final value of unadjusted function points for the inquiry is five (the maximum of the input and output part complexities).

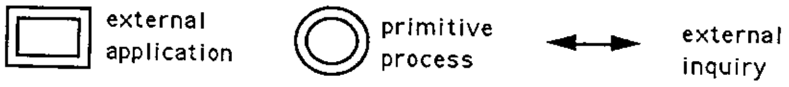

Figure 1. New Symbols for Dataflow Diagrams

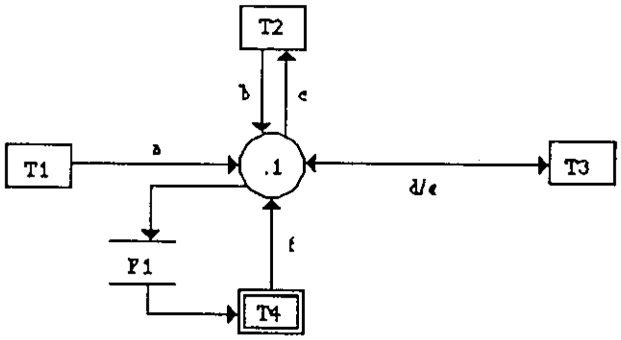

Figure 2. Context Diagram of the Sample Specification

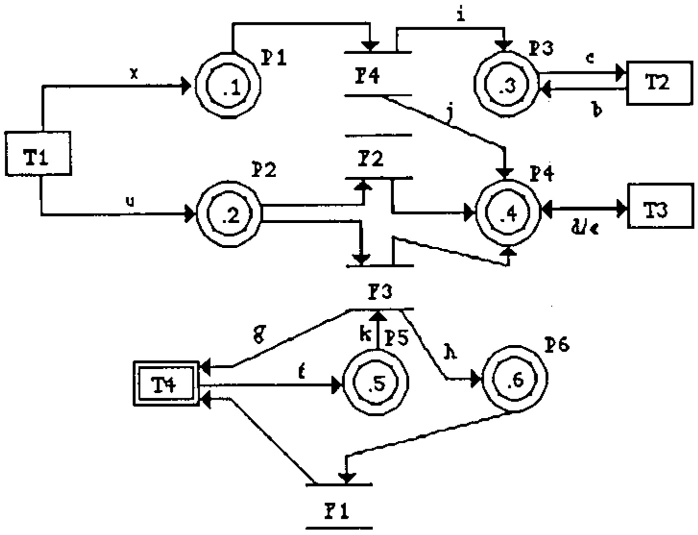

Figure 3. Decomposition of the Context Diagram

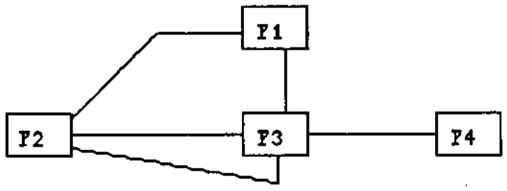

Figure 4. ERD of the Sample Specification

Table 3. The Data Dictionary of the Sample Specification

a	=	[x|u];				f	=	t1 + t2 + t3;
u	=	y + z;				k	=	@y1 + y4 + y5 + y6;
F4	=	x;				g	=	@y1 + y2 + y3 + y4 + y5 + y6;
F3	=	y;				x1-range	=	r1 + 42;
F2	=	z;				i	=	@x1 + x2 + x5;
F1	=	h;				j	=	@x2 + x3 + x4;
h	=	@y1 + y2 + y6;				b	=	x1-range;
x	=	@x1 + x2 + x3 + x4 + x5;				c	=	x1 + x2 + x5 + x-sum;
y	=	@y1 + y2 + y3 + y4 + y5 + y6;				d	=	x2;
z	=	@z1 + z2 + z3;				e	=	x2 + x3 + x4 + y3 + y4 + y6;
x-sum	=	**;						
x1	=	**;	x2	=	**;	x3	=	**;
x4	=	**;	x5	=	**;			
y1	=	**;	y2	=	**;	y3	=	**;
y4	=	**;	y5	=	**;	y6	=	**;
z1	=	**;	z2	=	**;	z3	=	**;
t1	=	**;	t2	=	**;	t3	=	**;
r1	=	**;	r2	=	**;			

External application T4 has three *external interfaces.* Data store F3 acts as a shared interface file, data store F1 forms an external (passed) interface file, and data flow *f* produces an input to the system (a transaction file). The DFDs of Figures 2 and 3 show an interface between the *shared data store* F3 and the external application T4. This shared interface is presented by a dataflow labeled *g*. Dreger (1989) suggests that each logical group of user data shared with two applications should be considered as an external interface. Based on this principle, the size of the considered shared interface is six, determined by the number of data items in the dataflow *g*. Clearly, the only record format of the shared interface is *g*. The number of file relationships for the shared interface is interpreted the same as the number of file relationships for the shared data store F3 (Rask 1992). From the ERD of Figure 4, we can count the number of relationships for data store F3 to be four. The size (six data items), and the maximum of the number of logical record formats and the number of file relationships (four), state that the complexity class for the shared interface *g* is simple, and thus it will be credited five unadjusted function points.

External interface file F1 has only one logical record format (it is referenced only by one key, the primary key of the file) but, from the ERD of Figure 4, we can count that the number of relationships is two. Because the maximum of these weighting values is two and the number of referenced data items in file F1 is three, the number of unadjusted function points for this interface will be five (simple).

The last interface *f* is sent to the system in the form of a *transaction file*. Interface points for a transaction file are counted only if it is converted (Dreger 1989). A *conversion* is made if the input flow of the transaction process does not balance with the output flows of the transaction process, i.e., the process changes the contents of the input flow. No conversion is made if the input flow is, say, A and the label A is defined as A = [B|C|D] or A = B+C+D in the data dictionary and the output flows from the process are labeled with a combination of B, C and D. Should any other labels, including blank labels, be used for the output flows, then our prototype system interprets that the transaction file has been converted.

The transaction process P5 in Figure 3 converts the transaction input flow *f* into flow *k*. That is why we need to count the interface points for this input flow. From the data dictionary definition (f = t1 + t2 + t3) we can recognize that flow *f* (transaction file) has only one logical record format and the number of elementary data items in flow *f* is three. Therefore, the amount of function points for this interface will be five (simple). In any case no file relationships can be found for transaction files.

Table 4. Logical Files of the Sample Specification

Logical File	Logical Record Formats
<F1,y1>	F1
<F2,z1>	F2
<F3,y1>	F3, k, h, q
<F4,x2>	j
<F4,x1>	F4, i

Finally, we have to count functions for the *logical internal files*. By examining the DFD and the data dictionary, we can derive the logical files in the system. First we must find those data stores that need to be taken into account when we examine internal logical files. Data stores F2 and F4 are clearly internal because only the system uses them. Data store F3 is a shared one because it is referenced by an external application. It should also be counted as an internal one because it is also referenced by the system. Data store F1 is external. Since it is sent as an output to an external application, it should also be counted as an internal file (Dreger 1989). The logical files, their keys, and logical record formats are shown in Table 4.

In our example, the external (interface) data store F1 has only one unnamed referencing data flow. Accordingly, the data store F1 has only one logical file <F1,y1> and it has just one logical record format (F1). Because the maximum of the number of the relationships (two) and the number of logical record formats (one) is two, and the size of the logical file is three (F1 = @y1 + y2 + y6), the unadjusted function points for logical file <F1,y1> is seven (simple).

In the same manner we can calculate the function points for the logical file <F2,z1> to be seven (F2 has three relationships, the logical file has just one logical record format, and the size of the logical file is three). Data store F3 as well has only one logical file <F3,y1>. However, we now have a total of four logical record formats in the logical file. Because the number of relationships of F3 is four, and the size of <F3,y1> based on its logical record formats is six (the number of elementary data items in the union of the elementary components of the record formats F3, *k*, *h* and *g*), we can conclude that the number of function points here is seven (simple). Data store F4 has two logical files <F4,x1> and <F4,x2>. The number of relationships for these files is one. Logical file <F4,x1> has two (F4 and i) and <F4,x2> has one logical record format (j). After counting the sizes of the files, we can conclude that the number of function points for both files is seven (simple). Table 5 summarizes the results of the function point counting process. The total value of unadjusted function points in our example is 71.

Table 5. Summary of the FPA

Business Function	Number	Complexity		Factor		TOTALS Line	TOTALS Group
OUTPUTS	1	SIMPLE	*	4	=	4	
	0	AVERAGE	*	5	=	0	
	0	COMPLEX	*	7	=	0	
						4	4
INPUTS	3	SIMPLE	*	3	=	9	
	1	AVERAGE	*	4	=	4	
	0	COMPLEX	*	6	=	0	
						13	13
INQUIRIES	0	COMPLEX	*	6	=	0	
	1	SIMPLE	*	4	=	4	
	0	AVERAGE	*	5	=	0	
	0	COMPLEX	*	7	=	0	
						4	4
FILES	5	SIMPLE	*	7	=	35	
	0	AVERAGE	*	10	=	0	
	0	COMPLEX	*	15	=	0	
						35	35
INTERFACES	3	SIMPLE	*	5	=	15	
	0	AVERAGE	*	7	=	0	
	0	COMPLEX	*	10	=	0	
						15	15
TOTAL UNADJUSTED FUNCTION POINTS (TUFP) =							71

3.3 Mark II Analyzer

To illustrate the counting process of the Mark II metric, we will use the dataflow diagram in Figure 4 and the data dictionary in Table 3. To find the logical file references for inputs and outputs for the Mark II metric in a dataflow diagram, we use the definition of a transaction given by Shoval (1988).

To calculate the number of input and output data elements, we sum the corresponding data elements in the data dictionary. Each input and output is processed exactly once. Each inquiry contains one separate input and one separate output. The interfaces are interpreted as inputs or outputs depending on their direction. For example, the shared file connection *g* and the external file F1 in Figure 4 are interpreted as outputs and the transaction file connection *f* as an input. These interpretations correspond to the transformation rules of Symons (1991). The results are shown in Tables 6 and 7.

From the equation (4) above we can count the total unadjusted value for the Mark II metric without the weights to be: UFP = 20 + 14 + 19 = 53.

4. DATA GENERATION AND MEASUREMENT

We used our prototype system to calculate Albrecht's and Mark II unadjusted function point metrics from DFDs and analyzed the relationships between these metrics. Our goal was to see whether there is a reliable correlation between the software sizes calculated using these metrics. Because we had no possibility to obtain real life data from the software industry to do the comparison, we decided to simulate real data by generating test material using randomized SA descriptions. By randomizing we mean here generating a random data dictionary for each DFD. Figure 5 shows the phases of the data generation and measurement process.

Table 6. Summary of Mark II: Inputs and File References

Inputs		Entities Referenced	
Name	Data Elements	Name	Total
x	5	<F4,x1>	1
u	9	<F2,z1>, <F3,y1>	2
b	2	<F4,x1>	1
d	1	<F2,z1>, <F3,y1>, <F4,x1>	3
f	3	<F3,y1>	1
Total	20		8

Table 7. Summary of Mark II: Outputs and File References

Outputs		Entities Referenced	
Name	Data Elements	Name	Total
c	4	<F4,x1>	1
e	6	<F2,z1>, <F3,y1>, <F4,x1>	3
g	6	<F3,y1>	1
F1	3	<F3,y1>	1
Total	19		6

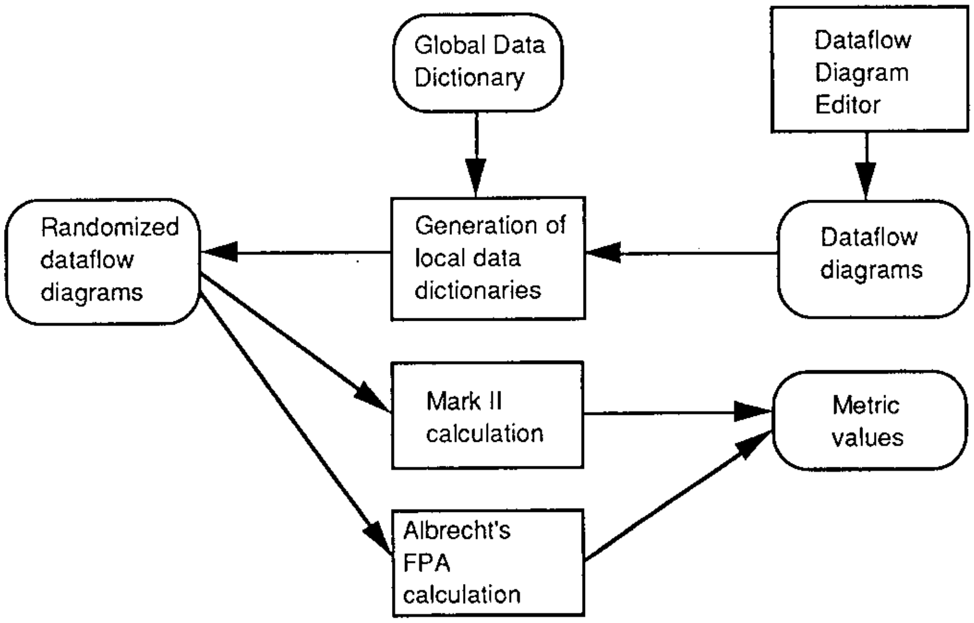

Figure 5. Data Generation and Measurement Process

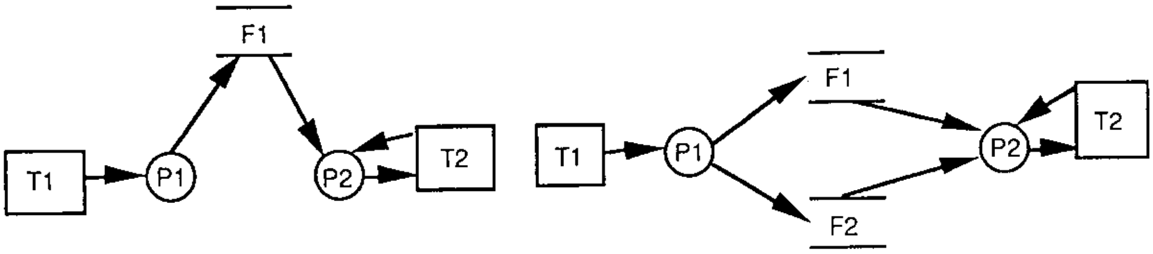

Figure 6. Two Examples of Test Dataflow Diagrams

The global data dictionary is generated taking into account the distribution of referenced data items for files in Albrecht's FPA. The upper limit of the size of each data specification is 69 primitive data items. The global dictionary contains these 69 different primitive data item names and the different combinations of these.

The *basic* dataflow diagrams in our study contain inputs, outputs, internal files, and processes (Rask and Laamanen 1991). The number of processes is from one to seven and the number of files does not exceed the number of processes. The upper limit for the number of processes is based on the distribution of file relationships in Albrecht's FPA. Each data store is updated by one or more input dataflows and each data store is used by one or more input/output combinations. The number of dataflows around data stores and around the processes is minimized.

For one process, we can draw only one basic type DFD. Figure 6 shows the possible diagrams when we have two processes. When we have three processes and one data store, it is possible that either two processes update the data store and one process uses it, or one process updates the data store and two processes use it.

When we form all possible combinations based on these principles, the total number of basic dataflow diagrams grows to 113. The range of the diagram sizes in unadjusted Function Points varies from 14 to 242. In this respect, the size of our material is comparable with the published materials used in the empirical studies (Table 8). In fact, the number of systems (diagrams) in our current study is greater than normal in empirical studies and our test environment enables even larger material.

Table 8. A Comparison of FP Sizes in Empirical Studies

Reference	TUFP Range
Albrecht and Gaffney (1983)	199 - 1902
Behrens (1983)	27 - 599
	22 - 435
Kemerer (1987)	100 - 2307
Low and Jeffery (1990)	23 - 76

Each data store in a randomized dataflow diagram contains from one to sixty-nine primitive data items. One of them is always a primary key. Each output dataflow contains from one to twenty-four primitive data items and each input dataflow contains from one to nineteen primitive data items. These upper limits are based on the distributions of data items referenced in FPA. The number of file relationships are produced randomly. The upper limit here is the number of data stores minus one.

In addition to applying basic DFDs in the calculation, we wanted to examine the effect of inquiries, interfaces, and naming of file references on size measures. To measure the *effect of inquiries*, we replace each input/output combination of the basic diagrams by an external inquiry symbol. Based on the principles of FPA, an external inquiry can not update a data store. To explain the *effect of interfaces*, we add one instance of each interface type (transaction file, shared file, and external file) into the basic diagram. *Naming of file references* affects the number of internal logical files in FPA. To analyze its effect, we use limits 0% (all file references unnamed) and 100% (all file references named).

5. DATA ANALYSIS OF TWO METRICS

We carried out statistical analysis to the results of our experiment by counting correlation coefficients and by regression analysis. These are also recommended by Conte, Dunsmore and Shen (1986) and Bourque and Ct (1991) for metric comparison. We tested the effect of each system type separately. In randomizing local data dictionaries, we used both a uniform and a normal distribution. In our first experiment, we produced random numbers that are uniformly distributed into range [0,1]. In our second experiment we used a transformation

$$(9) \quad Z = \sqrt{-2 \ln U_1} \cos(2\pi U_2)$$

to derive normally N(0,1) distributed random numbers from uniformly distributed random numbers. In equation (9), U_1 and U_2 are uniformly distributed into range [0,1]. The normally N(0,1) distributed random numbers we then transform into the range [0,1] using a mean 0.5 and standard deviations 0.15 and 0.25. By normalization we tried to reduce the effect of extreme values. For example, we wanted to avoid situations where a data store contains only one data item. Normalization thus gives more realistic data for statistical analysis. By using different standard deviations, we can further affect the shape of the normal distribution. By a standard deviation 0.15, we have tried to enforce all the random numbers into range [0,1]. By the standard deviation 0.25, we allowed part of the random numbers to fall outside the range [0,1]. This increases the number of extreme values to approach that of the uniform distribution.

We have summarized the results of our experiments in the following six correlation tables and two regression tables (Tables 9 through 16). We repeated our experiments ten times and counted the correlation r_i for each repetition i per each test group. After repetitions, we derived for each test group the mean correlation $\overline{r}$ (Tables 9, 12, and 14) and the summary of correlations based on the whole test material (Tables 10, 13, and 16). Regression lines were formed for N(0.5,0.0225) distribution when the named file reference percent is 100% (Tables 11 and 15). We made these restrictions after we observed that this distribution and the naming percent used gave the best correlation.

Table 9 represents mean correlations per $\overline{r}$ diagram type for each distribution. These have been calculated without taking into account weights of the Mark II metric. Correlations in Table 9 indicate the difference between the methods. We can see that the number of input and output elements exercises a great effect on the values of Mark II. The variation in the number of input and output elements easily causes an inverse deviation when compared with the corresponding Albrecht's TUFP value. The correlations in the three distributions are near one another. The best correlation can be achieved with the normal N(0.5,0.0225) distribution and the worst correlation with the uniform distribution. This is explained by the nature of the distributions. The effect of the extreme values is smallest with the normal N(0.5,0.0225) distribution.

The correlation using external inquiries is much lower than the correlations for basic diagrams. This is because with Mark II an inquiry includes one input and one output. Instead, with Albrecht's TUFP we have either one input or one output. Hence, when compared with basic diagrams, the inquiry diagrams do not affect the calculation of the Mark II metric values so much as with Albrecht's TUFP values. In particular, the interfaces increase the value of Mark II because they are interpreted as inputs and outputs. Thus, adding interfaces reduces correlation in the external interfaces class.

The naming of file references increases correlations in all groups. The naming of file references always increases Albrecht's TUFP values because it increases the number of logical internal files. The naming, however, has now effect on Mark II values.

Table 10 shows the total correlation without the weights of the Mark II metric when the named file reference percentage is 100%, the total correlation when the named file reference percentage is 0% (all file references are unnamed), and the total correlation counted from all the test material used. As we can see, the correlations are neither considerably dependent on the diagram types nor the naming percentage.

Table 9. Mean Correlations per Diagram Type without Weights of Mark II

Diagram Type	Named File Reference %	Mean Correlation $\bar{r}$ U[0, 1]	N(0.5, 0.0225)	N(0.5, 0.0625)
Basic diagrams	100%	0.66	0.68	0.66
	0%	0.60	0.64	0.61
External inquiries	100%	0.56	0.61	0.59
	0%	0.47	0.52	0.50
External interfaces	100%	0.40	0.53	0.47
	0%	0.38	0.52	0.44

Table 10. Total Correlations per Distribution without Weights of Mark II

Named File Reference %	U[0, 1]	N(0.5, 0.0225)	N(0.5, 0.0625)
100%	0.67	0.73	0.72
0%	0.67	0.74	0.71
all together	0.60	0.65	0.64

Table 11. Results from Regression Analysis without Weights of Mark II

Diagram Type	n	r	Regression Equation	σ_e
Basic diagrams	1130	0.68	UFP = 41.68 + 0.68 * TUFP	23.90
External inquiries	1120[1]	0.61	UFP = 51.41 + 0.65 * TUFP	24.68
External interfaces	1130	0.53	UFP = 132.83 + 0.70 * TUFP	37.92

[1]For inquiries, we cannot draw a DFD with one process only.

The diagram type independence can also be seen from the regression equations of Table 11. It lists the regression equations for N(0.5,0.0225) distribution when the named file reference percentage is 100%. As can be seen, the regression coefficients are close to each other for all equations. From standard deviations σ_e for the errors e_i in Mark II values (UFP), we can see that the deviations from the regression line are greatest with external interfaces. Instead, the standard deviations for basic diagrams and external inquiries are smaller and of equal size. The effect of the input and output sizes can also be recognized from the intercept of the regression lines on the Mark II axis. External interfaces have the greatest intercept point and diverging flows have the smallest.

When we use the weights for Mark II suggested by Symons (1988; 1991), using formulas (5) and (6) we obtain the two correlation Tables 12 and 13. Results in Tables 12 and 13 show that we can improve the correlations by reducing the effect of inputs and outputs and by increasing the effect of file references. The effect of adding weights is greatest with the uniform distribution. The changes with the normal

Table 12. Mean Correlations per Diagram Type with Weights of Mark II

Diagram Type	Name File Reference %	U[0, 1] UFP1	U[0, 1] UFP2	N(0.5, 0.0225) UFP1	N(0.5, 0.0225) UFP2	N(0.5, 0.0625) UFP1	N(0.5, 0.0625) UFP2
Basic diagrams	100%	0.80	0.80	0.83	0.84	0.81	0.82
	0%	0.76	0.75	0.79	0.79	0.76	0.76
External inquiries	100%	0.72	0.74	0.77	0.80	0.76	0.77
	0%	0.66	0.66	0.70	0.72	0.68	0.69
External interfaces	100%	0.52	0.46	0.68	0.63	0.60	0.54
	0%	0.51	0.44	0.66	0.62	0.58	0.52

Table 13. Total Correlations per Distribution with Weights of Mark II

Name File Reference %	U[0, 1] UFP1	U[0, 1] UFP2	N(0.5, 0.0225) UFP1	N(0.5, 0.0225) UFP2	N(0.5, 0.0625) UFP1	N(0.5, 0.0625) UFP2
100%	0.75	0.72	0.82	0.81	0.80	0.78
0%	0.75	0.72	0.81	0.80	0.78	0.77
all together	0.68	0.66	0.73	0.72	0.71	0.70

Table 14. Mean Correlations per Diagram Type with File References Only

Diagram Type	Named file Reference %	U[0, 1]	N(0.5, 0.0225)	N(0.5, 0.0625)
Basic diagrams	100%	0.95	0.97	0.97
	0%	0.95	0.96	0.96
External inquiries	100%	0.94	0.96	0.95
	0%	0.94	0.95	0.94
External interface	100%	0.95	0.96	0.95
	0%	0.94	0.96	0.94

distributions are in a similar direction. From Tables 12 and 13, we can also see that there is no significant difference between the equations UFP1 and UFP2.

We have found that it is possible to improve the correlation between Mark II function points and Albrecht's unadjusted function points by increasing the weight for file references and/or decreasing the weights for inputs and outputs in the Mark II formula. Table 14 shows the correlations when the weight for inputs and outputs is zero, i.e., when we ignore the inputs and outputs and take into account only the number of file references. The regression equations in Table 15 confirm the result.

We also tested the effect of the system size to the achieved correlation by dividing the test material into two parts. The first part (Part 1) contained the descriptions below the average Albrecht's function point size and the second part (Part 2) contained the descriptions above the average size. The resulting correlations are shown in Table 16. The

Table 15. Results from Regression Analysis with File References Only

Diagram Type	n	r	Regression Equation			σ_e
Basic diagrams	1130	0.97	UFP =	0.58 +	0.13 * TUFP	1.04
External inquiries	1120[1]	0.96	UFP =	1.16 +	0.14 * TUFP	1.18
External interfaces	1130	0.96	UFP =	-0.26 +	0.12 * TUFP	1.18

[1]For inquiries, we cannot draw a DFD with one process only.

Table 16. Total Correlations per Distribution with System Size

Named File	U[0, 1]		N(0.5, 0.0225)		N(0.5, 0.0625)	
Reference %	Part 1	Part 2	Part 1	Part 2	Part 1	Part 2
100%	0.76	0.77	0.84	0.84	0.81	0.77
0%	0.77	0.72	0.85	0.82	0.80	0.72
all together	0.66	0.63	0.73	0.68	0.69	0.63

Mark II values were calculated from equation (6). Clearly, the correlation values support Symons' (1991) opinion (see section 2.3) of the effect of the system size on the correlation. The correlation is better with smaller system sizes.

6. SUMMARY AND CONCLUSIONS

By implementing the calculation of Albrecht+s unadjusted function points (TUFP) and Symons' Mark II into a CASE tool, we have been able to compare and statistically analyze size measures based on these metrics using different types of randomized DFDs. To our knowledge, this is one of the first statistically reliable comparisons of different product size metrics where we have control of the type and size of the system. In fact, none of the earlier studies have sufficiently provided information of their analysis data.

Our approach leads to simulate real product size data in business applications in the early phases of the life-cycle. Simulation was accomplished by generating basic diagrams with a limited number of data stores and processes and then by adding new characteristics into these diagrams. The motivation behind this was to explain how differences in DFDs affect the correlation between these metrics. Obtained results show that different system types used in this study affect the correlations, although the effect was not as clear as found in our previous study when we compared Albrecht's FPA with DeMarco's (1982) Function Bang (Rask, Laamanen and Lyytinen 1992).

One limitation of the study is that it basically simulates "real world" DFDs. Because the generation of alternative DFD structures was based on an exhaustive and systematic enumeration of possible DFD structures, we believe that the simulation provides important and valuable information of size metrics with real data. What we do not know now is how common different DFD structures are in real business applications and this remains a research task. Another limitation is, of course, the use of DFD as a basis for complexity calculation. As DFDs do not provide any information of concurrence, timing or other resource constraints, or required user interface, it is impossible to include these aspects in the size estimation. Finally, we have not yet tested how well the algorithms compute the size of the system when compared to competent experts. This deals with the validity of simulations and it has to be addressed in future by some empirical studies.

The measurements confirm only partially Symons' (1991) doubt that the metrics do not correlate well. Based on our study of Mark II, this holds only for seize values which do not include weights (or when the weights are equal to one). By applying Symons' weights, it is possible to improve the correlation. If we take into account only the number of file references for Mark II, we get quite a good correlation. This "reduced" Mark II would be much easier to count when compared with Albrecht's FPA and therefore applicable over a number of environments. With regard to the system size, our results support the Symons' opinion that the Mark II method gives a higher UFP score as the system size increases than does the Albrecht method.

The system classes in our research are based on the function types of the FPA method. We have concentrated on studying the system types separately and have not paid attention to different system type combinations. Another area for further research is how the magnitude of the system type extensions affect the correlations between the metrics.

We argue that the automation of metrics calculation accomplished in this paper is an important step toward effective software project management. Using our data generation and analysis system, project leaders can carry out preliminary size estimations during analysis and specification phases and thus improve development productivity and reduce project risks. We plan to evaluate and compare the results by repeating the experiment by using real material from the software industry. If the industrial data can be fitted into our framework and the results are consistent with the results of our randomized material, the repetition justifies our claim of the necessity of a system classification in validating the size metrics. It would be also interesting to evaluate the equivalence between Albrecht's FPA and the reduced (counting the entity type references only) Mark II metric by using industrial material. Another avenue to assess the metrics with industrial material is to integrate our measurement components into current commercial CASE tools. To this end the algorithms have been published (Rask 1992) and they can be implemented in any of the commercial CASE tools that support SA.

It is also worth adapting the metrics to other development methods and techniques. For example, we can apply the metrics to the descriptions of Jackson's (1983) JSD method. It is also interesting to study new techniques such as neural networks for measurement purposes. It might be possible to teach neural networks to learn software metrics or cost equations. These are our future research objects.

7. REFERENCES

Albrecht, A. J. "Measuring Application Development Productivity." *Proceedings, Joint SHARE/GUIDE/IBM Application Development Symposium*, October 1979, pp. 83-92.

Albrecht, A. J., and Gaffney, J. E. "Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering*, SE-9(6), November 1983, pp. 639-648.

Banker, R. D.; Kauffman, R. J.; and Kumar, R. "Managing Development Productivity of the Computer Aided Software Engineering (CASE) Process with Dynamic Life Cycle Trajectory Metrics." Center for Research on Information Systems, New York University, Information Systems Department, Leonard N. Stern School of Business, December 1990.

Bourque, P., and Ct, V. "An Experiment in Software Sizing with Structured Analysis Metrics," *Journal of Systems and Software*, Volume 15, 1991, pp. 159-172.

Conte, S. D.; Dunsmore, H. E.; and Shen, V. Y. *Software Engineering Metrics and Models*. Menlo Park, California: The Benjamin/Cummings Publishing Company, Inc., 1986.

Cowan, J. B. "Quality Assurance Potential of Analyst/Designer Workbenches," *Information and Software Technology*, Volume 32, Number 1, January/February, 1990, pp. 46-52.

DeMarco, T. *Controlling Software Projects: Management, Measurement and Estimation*. New York: Yourdon Press, 1982.

DeMarco, T. *Structured Analysis and System Specification*. Englewood Cliffs, New Jersey: Prentice-Hall, 1979..

Dreger, J. B. *Function Point Analysis*. Englewood Cliffs, New Jersey: Prentice-Hall, 1989.

Hunter, R. "Lecture 1: Software Measurement," *Software Tools 1990: The Practical Use of Software Metrics*. Blenheim Online, 1990.

Jackson, M. *System Development*. Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

Kemerer, C. F. "An Empirical Validation of Software Cost Estimation Models," *Communications of the ACM*, Volume 30, Number 5, May 1987, pp. 416-429.

Laamanen, P., and Rask, R. "A Prototype System for Automating the Measurement and Verification of SA Descriptions." Report A-1991-2. University of Joensuu, Department of Computer Science, 1991.

Londeix, B. *Cost Estimation for Software Development*. Reading, Massachusetts: Addison-Wesley, 1987.

Low, C. G., and Jeffery, D. R. "Function Points in the Estimation and Evaluation of the Software Process," *IEEE Transactions on Software Engineering*, Volume 16, Number 1, January 1990, pp. 64-71.

Rask, R. *Automating Software Size Estimation During Requirements Specification Phase*. Unpublished Ph.D. Dissertation, University of Joensuu, Department of Computer Science, 1992.

Rask, R., and Laamanen, P. "Test Material for the Automatic Comparison of Albrecht's Function Point and DeMarco's Function Bang Metrics." Report B-1991-1. University of Joensuu, Department of Computer Science, 1991.

Rask, R.; Laamanen P.; and Lyytinen K. "Automatic Derivation and Comparison of Specification Level Software Product Metrics in CASE Environment — A Statistical Analysis of Correlations between Function Points and Function Bang." Submitted for publication, 1992.

Shoval, P. "ADISSA: Architectual Design of Information Systems Based on Structured Analysis," *Information Systems*, Volume 13, Number 2, 1988, pp. 193-210.

Symons, C. R. "Function Point Analysis: Difficulties and Improvements," *IEEE Transactions on Software Engineering*, Volume 14, Number 1, 1988, pp. 2-11.

Symons, C. R. *Software Sizing and Estimating, Mk II FPA (Function Point Analysis).* New York: John Wiley & Sons, 1991.

Yourdon, E. *Modern Structured Analysis*. Englewood Cliffs, New Jersey: Prentice-Hall, 1989.

8. ENDNOTES

1. This classification WAS originally suggested by Albrecht and has the value of being straightforward and easy to use. The classification of functions into these classes is, however, subjective and fairly intuitive. Therefore, several researchers have criticized this part of FPA heavily (see Symons 1988; Banker, Kauffman and Kumar 1990). In this paper, our aim is not to criticize or evaluate the internal validity of suggested metrics. Therefore we shall use the original weights suggested by Albrecht. These can be easily changed, if need be, in our measurement system.