Empirical Assessment of Methods for Software Size Estimation

Aleš Živkovič, Marjan Heričko and Tomaž Kralj
University of Maribor, Faculty of Electrical Engineering and Computer Science, Institute of Informatics
Smetanova 17, SI-2000 Maribor
ales.zivkovic@uni-mb.si, http://lisa.uni-mb.si

Keywords: Software metrics, Function Points, Software Size Estimation, Empirical Analysis

Received: July 20, 2003

In the software industry, many projects fail due to both the misjudgment of a project's size and faulty estimates correlated to this elementary metric. Several methods for software size estimation are present. The Function Points Analysis (FPA) method, however, is most frequently put into practice. After Albrecht introduced the FPA method, several variations evolved. All methods share the same fundamental idea, but differ in procedural steps and metric units. A descriptive approach is usually used for method comparison. To avoid the weaknesses of a descriptive approach, a mathematical model is defined and used for theoretical comparison. The complexity of the mapping functions prevent detailed comparisons -- consequently only general characteristics become evident. Characteristics exposed with a formalization of the rules were further studied in different test scenarios using historical data from past projects. Empirical results showed some limitations of the mapping function and anomalies in the data set used. The possible reasons for deviations in the data set were also analyzed.

1 Introduction

Software size estimation is a crucial element in a project manager's decision-making process, with regard to the project's duration, budget and resources. In the past, different methods were developed. Albrecht introduced the function point analysis method in 1979 [1], since then it has been the target of many scientific studies [4, 5, 6]. Some modifications have also been made resulting in new methods like Feature points, Full Function Points, Function Weight, Function Bang, Mk II Function Points Analysis, COSMIC-FFP and NESMA.

A comparison of different methods, based on verbal descriptions, lack the formalism needed to understand and compare them. In this paper, a mathematical foundation for describing the methods is established first, and then three popular methods are mapped into the universal form and compared. To compare the mapping functions, the empirical method is used. The paper is divided into four sections. In the first section, the methods for software size estimation are briefly presented. The subsequent section introduces the formal model for representing software-sizing methods. The third section describes test scenarios and presents results. The conclusion and plans for future work can be found in the last section.

1.1 Function Points

The idea behind function points is quite simple [2]. Every information system processes some data that can be stored in the application database or is gained from external applications. Four operations are performed on data records: create, read, update and delete. Besides that, information systems use several query functions for data retrieval and report construction. Each record consists of several fields of basic data types or another record that can be further decomposed. The FPA method quantifies: the number of fields in each record, the distinct operations performed on these records, and the number of these operations that are necessary to perform a business function. The sum over all business functions, multiplied with some empirically determined weights, represents unadjusted function points. The final calculation is made using a value adjustment factor (VAF) that measures system complexity. Figure 1 shows an overview of the tasks performed within the scope of the FPA method.

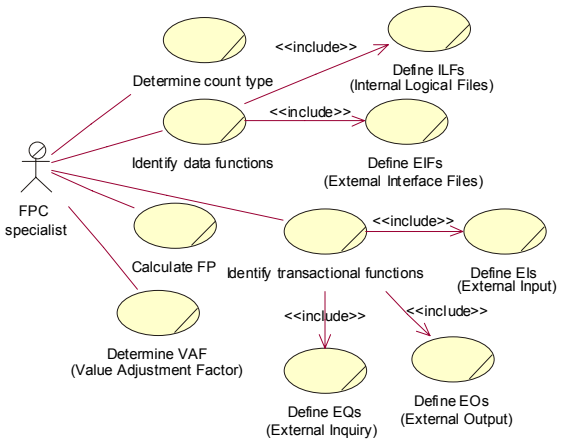

Figure 1: Business Use Case diagram for FPA method

1.2 COSMIC-FFP

The COSMIC-FFP method [10] reached standardization in 2003 as ISO/IEC 19761 and is the only method in accordance with ISO/IEC TR 14143 [7]. Its approach to size measurement is different from the original FPA, since data elements do not contribute directly to the size. The focus of interest is on data movement, which is defined by units of measure called Cosmic functional size units (Cfsu). In [10] the conversion factor for function points is given based on a sample application portfolio of 14 applications from two different systems. In general, the conversion factor is close to one, when comparing unadjusted function points. Methods distinguish between four different data movements (entry, exit, read and write), and the sum of their size represents the size of the system measured. Beside raw measurement rules, the method clearly defines its applicability in different circumstances (e.g. software domain, project phase). Its popularity among practitioners is growing.

1.3 Mark II FPA

In 1988, Charles Symons developed a variation of the FPA method [3] adding several new steps into the measurement process. Additional steps are bound into calculating the effort, productivity and influence in the technical complexity of a specific solution. A functional size itself is calculated as the weighted sum over all logical transactions of the input data element types (N_i), data entity types referenced (N_e) and output data element types (N_o). For the weights, the industry average is used with values Wi=0.58, W_e=1.66 and W_o=0.26. Compared to the original FPA method, the major difference is that MK II FPA is a continuous measure with linear characteristics. Therefore MK II FPA produces increasingly higher size estimates for projects with more than 400 function points. The primary domain for MK II FPA is business information systems. If applied to other domains, special attention has to be given to components with complex algorithms, since sizing rules do not take into account their contribution. For use with real-time systems, additional guidelines may be necessary [3].

2 FPC formal model

All methods for software size estimation lack formal foundations in their origin descriptions. There were some attempts [8, 9] in the past to add formalism to functional size measurement. Fetcke's model is applicable to different methods since it introduces an additional level of abstraction. The approach proposed by Diab et al. is designated to COSMIC-FPP and has a specific purpose. In our research, the model defined by Fetcke is used as a basis and further refined by the definition of a mapping function.

2.1 Generalized representation

According to measurement theory, every measurement can be represented as a function that maps empirical objects into numerical. The FPA method defines a function that maps a software system into a number. That number represents the size of the system. Since the FPA method is technologically independent, it introduces its own concept for representing a software system. The abstraction of a software system is data oriented and has two steps.

1. The software documentation is transformed into elements defined by the method.
2. Method elements are mapped into a numerical value representing the size of the system expressed in function points.

The procedure is presented in Figure 2 as a UML activity diagram. It shows a specific example of where Software Requirements Specifications (SRS) serve as an input to the FPA elements identification process. Elements are identified according to rules, and the outcome is a data-oriented abstraction of the software system. The second activity represents the mapping function. Several tables are used for the transformation of a separate element count into function points. The final result is the number of function points.

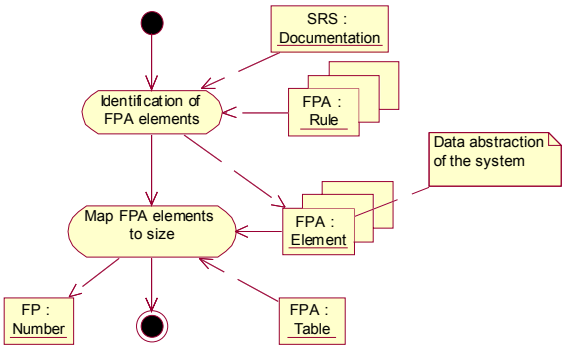

Figure 2: Data abstraction steps for FPA method

2.2 Data-oriented abstraction

Different methods enumerated in the introduction use different names for data abstraction elements; the rules for element identification are different, mapping functions also differ. However, similarities exist that can be described by the following core concepts:

- The user concept covers the interaction between a user and the system.
- The application concept represents the whole system as an object of the measurement.
- The transaction concept is the logical representation of the system's functionality. Transaction is the smallest independent unit of interest.
- The data concept deals with the subject of change within the system. The data element is the smallest unit of user observation.
- The type concept simplifies data handling via the abstraction of individual data elements.

On a higher level of abstraction, an application is represented with data and transactional types. The data

type is a set of data elements handled within the system. The transactional type is a sequence of logical activities. Fetcke defined seven classes of logical activities [8]:

- *Entry activity*. The user enters data into the application.
- *Exit activity*. Data is outputted to the user.
- *Control activity*. The user enters control information data.
- *Confirm activity*. Confirmation data is outputted to the user.
- *Read activity*. Data is read from a stored data group type.
- *Write activity*. Data is written to a stored data group type.
- *Calculate activity*. New data is calculated from existing data.

In Figure 3, a UML class diagram for data-oriented abstraction can be found. Based on the abstract presentation, mapping for a specific method can be made.

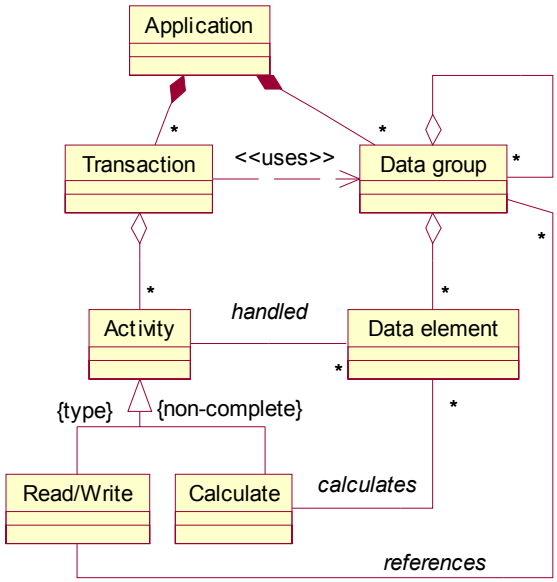

Figure 3: Relationship between FPC elements

2.3 Generalized structure

In this section we summarize the formal representation of concepts described in the previous section. An application closure H is defined as the vector of τ transactional types T and σ data group types F.

$$H = (T_1, \ldots, T_\tau, F_1, \ldots, F_\sigma) \qquad \textbf{(E 1)}$$

The transactional type T_i is a vector of activities

$$T_i = (P_{i1}, \ldots, P_{in_i}) \qquad \textbf{(E 2)}$$

An activity is further described by four attributes:

- its class $\theta_{ik} \in$ {Entry, Exit, Control, Confirm, Read, Write, Calculate},
- for read and write activities, the data group type referenced $r_{ik,}$
- the set of data elements D_{ik} handled and
- for calculate activities, the set of data elements calculated C_{ik}.

In the equation E3, i can have values from 1 to τ and represents the transaction activity it conforms to. k runs from 1 to n identifying activity within the transaction.

$$P_{ik} = (\Theta_{ik}, r_{ik}, D_{ik}, C_{ik}) \qquad \textbf{(E 3)}$$

The data group type F_j is a set

$$F_j = \{(d_{j1}, g_{j1}), \ldots, (d_{jr_j}, g_{jr_j})\} \qquad \textbf{(E 4)}$$

where the d_{jk} are data elements and the g_{jk} the designate sub-groups. j can have values from 1 to σ and represents the number of data types. k distinguishes between data elements and can have values from 1 to r.

2.3.1 Representation of the mapping function

In the previous section, a formal representation of transactional and data types was introduced. The system is composed of different data and transactional types. The number of data and transactional types, and their attributes, contribute to the size of the software system. Some methods also define the third component that has an influence on software size, and the technical complexity of the solution. The universal function that maps application attributes into size is therefore:

$$FPC(a) = \left(\sum_i FPC_1(t_i) + \sum_j FPC_2(f_j) \right) * FPC_3(TC) \qquad \textbf{(E 5)}$$

where

FPC(a) is the function that maps attributes of the application a into software size.
$FPC_1(t_i)$ is the function that maps transactional type t_i into size.
$FPC_2(f_j)$ is the function that maps data type f_j into size.
$FPC_3(TC)$ is the function that maps technical complexity of the anticipated solution for application a into a factor.

The total value for an application size is the sum of both parts multiplied by the factor of the solution's complexity. The factor can reduce or increase the overall size. However, it is not clear if the factor actually measures raw application size or is an attribute of the implementation and should be part of the function that

maps size to effort. In this research, the function of FPC_3 is not examined.

A generalized structure can now be used to define different methods. First, we will use it for representing the original FPA method.

2.3.2 Mapping for the FPA method

Since data functions from the FPA method correspond to data element type (F), data element type (DET) corresponds to data element and record element type (RET) is equivalent to sub-group defined in the generalized structure. The FPA method distinguishes between internal and external data requirements; generalized representation, however, defines more activity types than the FPA method. Therefore, we define external interface files (EIFs) as a data type that cannot be used in write type activities.
The mapping of the transactions is a bit more complicated and is summarized in Table 1. In the table, activities that are allowed in the transaction type are marked with X.

Table 1: Mapping of transactions into activities

		GENERAL STRUCTURE ACTIVITIES						
		Entry	Exit	Write	Read	Confirm	Control	Calculate
FPA	EI	X			X	X	X	X
	EO		X		X			X
	EQ	X	X		X	X	X	

The FPC functions for the FPA method would look like this:

$$FPC_1 = \sum_i W_{EI}(N_d, N_r) + \sum_j W_{EO}(N_d, N_r) + \sum_k W_{EQ}(N_d, N_r)$$
$$FPC_2 = \sum_l W_{ILF}(N_d, N_g) + \sum_m W_{EIF}(N_d, N_g)$$

(E 6)

where the W_{EI}, W_{EO}, W_{EQ}, W_{ILF}, W_{EIF} are functions that prescribe the number of function points for every FPA function identified in the measurement process. Function W has two parameters. For transactional functions, parameters are the number of data element types (N_d) and number of file types referenced (N_r), for data functions parameter N_g is used instead of N_r, representing the number of record element types. Functions W_x are the step functions represented by discrete values with the following range:

W_{ILF}={7, 10, 15}
W_{EIF}={5, 7, 10}
W_{EI}=W_{EQ}={3, 4, 6}
W_{EO}={4, 5, 7}

Given as an example, the step function W_{ILF} is defined as:

$$W_{ILF}(N_d, N_g) = \begin{cases} 7; & ((1 \le N_d \le 19) \wedge (1 \le N_g \le 5)) \vee \\ & ((20 \le N_d \le 50) \wedge (N_g = 1)) \\ 10; & ((1 \le N_d \le 19) \wedge (6 \le N_g)) \vee \\ & ((20 \le N_d \le 50) \wedge (2 \le N_g \le 5)) \vee \\ & ((51 \le N_d) \wedge (N_g = 1)) \\ 15; & ((20 \le N_d \le 50) \wedge (6 \le N_g)) \vee \\ & ((51 \le N_d) \wedge (2 \le N_g)) \end{cases}$$

(E 7)

2.3.3 Mapping for the MKII FPA

In the Mark II FPA method, data groups are called entity types and do not directly contribute to functional size. Therefore, FPC_2=0 in all cases. Logical transactions are broken down into activities. There are only three types of activities in MK II FPA, namely input, processing and output. Table 2 shows mapping for activities defined in generalized form. Notice that MK II FPA does not have an equivalent to the calculate activity, which is due to processing activity deals with existing entities, and which conforms with the read and write activity types.

Table 2: Mapping for MK II FPA

		GENERAL STRUCTURE ACTIVITIES						
		Entry	Exit	Write	Read	Confirm	Control	Calculate
MK II FPA	Input	X					X	
	Output		X			X		
	Processing			X	X			

$$FPC(a) = \sum_j (W_i * N_{di}) + (W_e * N_F) + (W_o * N_{do})$$

(E 8)

where the W_i is the weight for input elements and has a constant value of 0.58, W_o, is the weight for output elements with the value 0.26, W_e is the weight for entities referenced in processing with the value 1.66, N_{di} is the number of data elements used in the input activity, N_{do} is the number of data elements used in the output activity and N_F is the number of entities used in processing.

2.3.4 Mapping for COSMIC-FPP

As described in the introduction, the COSMIC-FFP method defines Cfsu as a unit of measure and introduces a different approach to software sizing. The method counts data movements that can be one of four types: entry, exit, read, and write.

$$FPC = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} \sum_{k=1}^{4} P_{ik}(F) \quad \textbf{(E 9)}$$

Equation 9 shows the mapping function. The sum across all identified transactions (T_i) is made in the first part of the equation. In the second part, transactions are broken down into activities (P_{ik}), where k runs from 1 to 4, since the method has only four types of activities. With the F in brackets, we have revealed that activity depends on data types, since data is the object of movement. Again FPC_2=0 and only FPC_1 contributes to the application size.

It can be seen from the equations E6, E8 and E9 that FPC functions of selected methods are multivariable, thus further research into them is complex. To observe them in specific situations, we have set a few test scenarios described in the text section.

3 Test scenarios

With methods for software size estimation two kinds of errors are likely to occur: a method error and a measurement error. A method precision is not formally defined nor statistically proven. Approximate values can usually be found that imply method accuracy. Since the behavior of the FPC function is dimmed, it is difficult to predict results in all circumstances. To analyze the basic characteristics of the FPC function for three selected methods, we have to construct three diverse scenarios. The findings help us choose the right method for the given problem domain.

A measurement error can be identified via an analysis of historical data. In the second part of our research, only the original FPA method was used to estimate the size of eight applications. The data gathered were used as the small dataset and compared with the industry average. The deviations in the dataset are analyzed in the second part of this section.

3.1 Empirical comparison of different FPC functions

In the first part of our research we set up three different scenarios, applied different methods and compared the results in order to find deviations between methods' FPC functions. In this research, we decided to apply the original FPA method, MK II FPA and COSMIC-FFP. For the first case we chose only one requirement from the bigger payroll application. The purpose of the selected requirement was to print out specific data in order to monitor the final account for a specific period of time. We named this function Account Control. Let's summarize the measurement technique for all three methods. Because we have chosen only part of the whole application, applying COSMIC-FFP, some steps were excluded from the counting procedure. We followed only the necessary steps in performing the task. In applying the original method and MK II FPA, the Value Adjustment Factor (VAF) was not calculated. Therefore, size is expressed in unadjusted function points (UFP). Lokan discouraged the use of VAF, according to his empirical analysis of FP adjustment factors [4]. The VAF was found not to improve the relationship between FPs and effort. For most projects, the VAF does not result in much change to the function point value [11]. To get considerably accurate results with the original FPA method, we added the contribution of data functions to the value of unadjusted FPs. Since only part of the system was sized, equation E13 was used. FPC_F represents the contribution of data functions and is calculated from the total contribution of data functions (FPC_2) divided by the number of data functions (l and m) and multiplied with the number of referenced data types (N_r).

$$FPC_F = \frac{FPC_2}{l+m} * N_r \quad \textbf{(E 10)}$$

Table 3: Results summary for the test scenarios

	SIZE IN FUNCTION POINTS		
	FPA	MK II FPA	COSMIC-FFP*
TS1	5. 5	10.5	5
TS2	14	33.1	18
TS3	3	3.2	5

* 1Cfsu treated as 1 FP [10]

Table 5 summarizes the results for all test scenarios. The row labeled TS1 shows results from the first test scenario, Account Control function. With 10 function points, MK II FPA produced twice the results of the other two methods. The reason for this lies in the three referenced entities that added 5 FP.

In the next scenario we measured function behavior, using many data element types (DET). In the original FPA method, the increased number of data element types did not influence the contribution of the transactional function to the final size. By comparison, MK II FPA reacts to all changes with a greater amount of FPs. In the example, personal data has to be entered for an employee. The number of attributes was set to 51. The second row in table 5 summarizes the results. The FPA method produced the smallest size, since its FPC function is a step function that cannot follow growth in data elements and referenced data types. The COSMIC-FPP method follows the change in the number of data elements with its read activity, however data elements are grouped for the entry activity. Consequently the final results are smaller than with MK II FPA. The MK II FPA has produced the greatest number of function points. The changes in the number of data elements directly influence the final amount with the factor 0.58. In our opinion, however, it is difficult to predict, in cases like this, how much more effort is necessary when we increase the number of data element types.

Our last test deals with real time applications. The original FPA method is already known to produce non-accurate results for applications in this group. How about the other two methods? We measured the size of

applications that regulate the temperature in a building. The results can be found in the third row (Table 5).
We can argue that only the COSMIC-FPP method produced a correct result, since the result of the Mk II FPA is almost the same as the result of the original FPA method, known not to produce accurate results within real-time systems. Therefore, the warning concerning the real-time application domain in the MK II FPA manual has to be taken seriously. The difference would be even greater if more sensors gathering data and controlling output were introduced.

3.2 Influence of subjectivity

In the second part of our research we selected eight different applications developed by the same company. Although it would be better to apply all three methods, only the original FPA method was used. The applications under consideration had to be developed in the same environment, for the same target platform, with the same tools and the same group. Consequently, only eight applications satisfied the criteria. We have marked applications with letters from A to H. Table 3 briefly illustrates all the selected applications.

Table 4: Short description of selected applications

A	Most recently developed application and currently in use. Analysis and design were carried out in a systematic manner. Therefore all the documentation was available.
B	A lot of documentation exists, describing the current state of the application. It was recently enhanced.
C	Older application with incomplete documentation. It was changed many times in the past without making the appropriate corrections in the corresponding documentation.
D, E, F	Applications had some kind of documentation that was not precise enough to perform the count.
G	Newer application with good documentation.
H	The largest application developed as an answer to the Y2K problem. It was developed under stress and lacks proper documentation.

In cases D, E, F and H we have used GUI forms and E-R model to perform the count. For these applications, the counting specialist took additional measures; consequently the counting speed was reduced.

Table 5: Results for the original method

Application	Number of FPs	Effort (hours)	PDR (h/FP)	PDR_{ISBSG}
A	102	726	7.1	3.6
B	141	1000	7.1	4.0
C	128	800	6.2	3.9
D	254	1600	7.5	5.1
E	472	3000	6.3	6.5
F	485	3000	6.2	6.5
G	624	4400	7.1	7.2
H	719	6000	8.3	7.6

Table 2 depicts the counting results for all eight applications. In the second column, the application's size is expressed in FPs showing that A, B and C are smaller applications; applications G and H are larger. In the third column, we can find the number of hours spent on implementation. The last two columns show the value calculated from the number of FPs and hours spent on implementation. It represents the amount of time spent implementing one FP and its so-called Project Delivery Rate (PDR). PDR_{ISBSG} is the value calculated from the ISBSG repository [11] data using equation E10 with the values 0.587 for constant C, and 0.390 for constant E. The FPC is the function point count expressed in function points. The calculated value represents the "normal" PDR for the project of a specific size, according to the repository's average. Comparing these two values, the performance of projects A, B, C and D is quite unsatisfactory, with deviation from 50% to 100% in hours spent implementing one function point.

$$PDR = C * FPC^{E} \quad \textbf{(E 11)}$$

The reason for the deviation in the results could be one of the following:

- An error occurred within the counting procedure, resulting in less function points.
- The project group performance was actually below the industry average.
- The documentation for the application does not include all features developed.

From the theoretical point of view, only the first case is interesting. Let us assume that the types of some elements were mixed. Equations E11 and E12 calculate the error. In the equation E11 the error for data functions is calculated. To get concrete numbers we need a ratio between data element types.

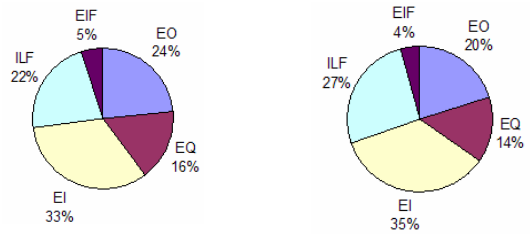

Figure 4: Relationship between FPA elements

The left graph in Figure 4 shows the ratio between FPA elements from the ISBSG repository with 238 projects in the sample, on the right is the graph for our sample. According to the industrial average, there are less then 20% of external interface files in the data functions for standalone applications developed from scratch. If we count all elements as internal logical files (ILF) the error made is around 5 %. In the opposite case, when we neglect ILFs, the error is 35 %.

$$E_{DF} = \frac{\sum_i W_{ILF}(F_i)}{\sum_j W_{ILF}(F_{ILF\,j}) + \sum_k W_{EIF}(F_{EIF\,k})}$$

$$E_{EIF} = \frac{N_{EIF}}{N_{DF}} * \frac{W_{EIF}(F_{ILF})}{W_{ILF}(F_{EIF})} \cong \frac{N_{EIF}}{N_{DF}} * \bar{e}_{EIF} = 0.185 * 0.3 = 0.0555$$

$$E_{ILF} = \frac{N_{ILF}}{N_{DF}} * \frac{W_{ILF}(F_{ILF})}{W_{EIF}(F_{ILF})} \cong \frac{N_{ILF}}{N_{DF}} * \bar{e}_{ILF} = 0.815 * 0.44 = 0.3586$$

(E 12)

The equation E12 calculates the maximal error for transactional functions. Since external inputs and external inquiries have the same weight, they are treated equally. If external inputs and external inquiries are both neglected and all transactional functions are treated as external outputs, errors could be as high as 16%. If external outputs are mixed with external inputs or external inquiries, the error is around 7%. The final numbers are specific for the case presented in the paper and must be recalculated for other types of applications.

$$E_{TF} = \frac{\sum_i W_{EI,EQ}(T_i)}{\sum_j W_{EI}(T_{EI\,j}) + \sum_k W_{EO}(T_{EO\,k}) + \sum_l W_{EQ}(T_{EQ\,l})}$$

$$E_{EI,EQ} = \frac{N_{EI,EQ}}{N_{TF}} * \frac{W_{EI,EQ}(T_{EI}\,/\,T_{EQ})}{W_{EO}(T_{EI}\,/\,T_{EQ})} \cong \frac{N_{EI,EQ}}{N_{TF}} * \bar{e}_{EI,EQ} = 0.67 * 0.25 = 0.1676$$

$$E_{EO} = \frac{N_{EO}}{N_{TF}} * \frac{W_{EO}(T_{EO})}{W_{EI,EQ}(T_{EO})} \cong \frac{N_{EO}}{N_{TF}} * \bar{e}_{EO} = 0.33 * 0.2 = 0.066$$

(E 13)

3.3 Method comparison

All compared methods use the same type of abstraction, based on requirements document for the software system. However, the FPC function that maps identified elements to the size is different and difficult to compare mathematically. Therefore we have set three diverse test scenarios to evaluate a method's performance. To be able to observe a function's behavior, test scenarios were simplified and may be unusual for real-world application. In the first test case, the characteristics of functions were analyzed. In the documentation for MK II FPA and COSMIC-FPP, the function is described as linear with respect to the number of data elements. The original FPA method measures the size of the transactional function according to its complexity. Transactional functions can have 3 to 7 function points regardless of their simplicity or complexity. The graph in Figure 5 shows the behavior of unadjusted function points for external inputs (EI) and external outputs (EO) with respect to the number of data element types for MK II FPA and the original FPA. From the graph it is easy to see when the threshold for the original FPA method is reached and higher values of data element types (DET) do not influence the number of unadjusted function points (UFP). In the COSMIC-FPP method, the behavior of the function depends on the grouping of data elements. For example, all 51 elements can be treated as one Cfsu in the case of entry activity. On the other hand, they influence the final size through the read activity contribution. The method is more complex to use than the other two compared in the test.

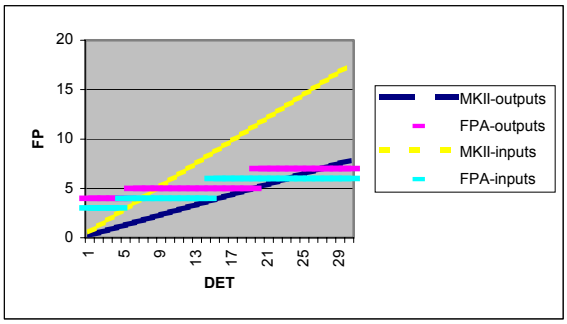

Figure 5: The difference between MK II FPA and original FPA

In the last test of the first scenario, two presumptions were relied upon. The first one was that original FPA does not perform well measuring real-time applications and the second was that COSMIC-FPP does. The results confirmed the generally accepted opinion that MK II FPA and original FPA perform poorly with real-time applications and COSMIC-FPP is the most appropriate method for that domain.

In the second test scenario, an anomaly in the PDR values appeared. We compared original values with values calculated from the ISBSG repository. In the relationships between the FPA elements, it is possible to calculate errors for an element type mismatch. Although the error could rise up to 35 % in our case, the reason for the anomaly was either a poor group performance or incomplete documentation.

4 Conclusion

Software size is an important attribute, which we can use to manage the software development process. It is easy to calculate the effort and costs of a project and to monitor its progress. We can predict the software's size based on experiences from past projects or we can use methods and empirical data. Many projects from the past that relied on a project manager's intuition failed. Thus, we suggest using some method. In this paper, several methods were evaluated and compared in diverse scenarios. The results showed that the consistent use of a selected method in the same environment gives a better understanding of both the project size and the delivery rate. Anomalies in the results become quickly evident and can be analyzed with the help of the general representation of methods for software size estimation, proposed by Fetcke and supplemented with the mapping functions presented in this paper. The mathematical representation of the method and its rules, make them more evident and easier to analyze. The comparison showed that it is important to choose the most appropriate method for the given problem domain to exclude the possibility of anomalies in the results. Our research confirmed that the MK II FPA has some advantages compared to the original FPA method, notably when a lot of DETs are present in the

application. However, both methods performed poorly in the case of real-time applications and system software. The COSMIC-FPP method gives better results with a higher number of FPs.
Function points have suffered a lot of criticism that has discouraged their use in practice. With the data from past projects and with the industry average, both problems could be overcome, namely the problem of early estimates and the human factor problem.
In the future, we will try to apply the second test scenario to another two methods, theoretically compare the mapping functions and express the error.

6. References

[1.] J. Albrecht (1979), *Measuring Application Development Productivity*, Proceedings of Joint SHARE, GUIDE and IBM Application Development Symposium, str. 83-92.
[2.] M. Bradley, ur. (1999), *Function Point Counting Practices Manual, Release 4.1*, International Function Point Users Group (IFPUG), Westerville
[3.] *Mk II Function Point Analysis, Counting Practices Manual*, The UK Software Metrics Association., http://www.uksma.co.uk/public/mkIIr131.pdf
[4.] C.J. Locan (2000), *An empirical analysis of function point adjustment factors*, Information and Software Technology, 42, 649-660
[5.] C. Yau, H. Tsoi (1998), *Modeling the probabilistic behavior of function point analysis*, Information and Software Technology. 40, 59-68
[6.] J.J. Dolado (1997), *A Study of the Relationships among Albrecht and Mark II Function Points*, Lines of Code 4GL and Effort, Journal Systems Software, 37, 161-173
[7.] ISO/IEC 14143-1:1998, *Functional size measurement*, ISO standard, 1998
[8.] T. Fetcke (1999), *A Generalized Structure for FPA*, IWSM'99
[9.] H. Diab, M. Frappier, R. St-Denis (2001), *A Formal Definition of COSMIC-FFP for Automated Measurement of ROOM Specification*, Proc. Fourth European Conf. Soft. Measurement and ICT Control, pp. 185-196
[10.] *COSMIC-FFP Measurement Manual*, version 2.2, January 2003
[11.] P.R. Hill (2001), *Practical Project Estimation*, ISBSG