Early FP Estimation and the Analytic Hierarchy Process

Luca Santillo (luca.santillo@gmail.com)

Abstract

Several methods exist in order to estimate the size of a software project, in a phase when detailed Function Point counts cannot be performed. While the Derived Estimation (or Algorithmic Model) Methods result in more accurate, objective, and repeatable estimates, these need always to be calibrated through a set of historical (statistical) data; this also means that estimation for atypical cases or new software typologies cannot be provided because of this "calibration from the past".

The paper shows how the combination of the Early FP estimation method with the quantification technique of subjective judgements, known as Analytic Hierarchy Process, can result in a enhanced, multi-approach estimation method. The EFP method provides a breakdown, hierarchical structure of the software functional items, while the AHP technique, based on pair-wise comparisons of all the items (and sub-items) of the hierarchy, provides the determination of a ratio scale of relative values between the items, through a mathematical normalization. Consequently, it is not necessary either to evaluate the numerical value of each item, or to use statistical calibration values, since the true values of only one or few components are propagated in the ratio scale of relative values, providing the consistent values for the rest of the hierarchy. This methodological combination can be even improved by using as reference a set of few well-known functions, structured into Catalogues of Functionalities, if present.

This new combined estimation method is robust to errors in the pair-wise comparisons, and self-consistent because of the redundancy and the normalization process of the comparisons. A natural extension to this approach is the link with the general estimation of other variables of the software project, as effort or quality, by means of the same reasoning.

1. Introduction

A Function Point count always requires the exact identification and classification of all logical files and elementary processes in the system to be measured. Several methods are known for estimating the number of FP before accessing to all the detailed information about the system, and can be divided into two main categories, Direct Estimation (or Expert Opinion) Methods and Derived Estimation (or Algorithmic Model) Methods; both of these have their own pros and cons, as shown in *Table 1* (see [4], [5] for a more detailed analysis). Mostly, while direct estimation can provide quick, but not accurate values, derived estimation can provide more precise estimates, but strongly depending on empirical-statistical data.

Table 1. Pros and cons of the FP estimation methods.

Category	Strengths	Weaknesses
Direct Estimation Delphi Technique Analogy	Suitable for atypical projects	Difficult to justify the results
		Depend on the expertise
Derived Estimation Extrapolation Sampling Average Complexities	Objective and repeatable	Calibrated to past, not future
	Calibrated to historical data	Not suitable for atypical projects
	Easy for non-experts	Depend on the availability of historical data

2. Early Function Point (EFP)

The Early FP estimation method, in order to provide better estimates of the software size, combines different approaches, making use of both analogical and analytical classification of functionalities [4]. Moreover, EFP lets the estimator use different levels of detail for different branches of the system (*multi-level approach*): the overall global uncertainty in the estimate will then be the weighted sum of the individual components' uncertainties. The multi-level approach let us exploit all the knowledge we have on a particular branch of the system, without asking questions difficult to answer in an early stage or, on the contrary, leaving some detailed information on a portion of the system unused.
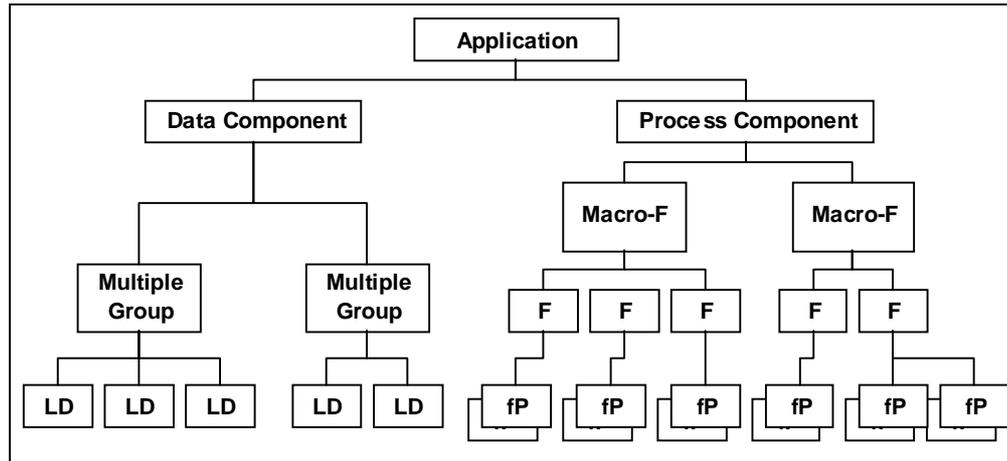

Figure 1. Hierarchical functional decomposition in EFP model.

EFP identifies software objects, like logical data and functionalities provided by the software to be estimated, on the following hierarchical scale (see *Figure 1*): macro-functions, functions, micro-functions, functional primitives and logical data groups with various ranges. Each of these objects is assigned a set of three FP values (minimum, most likely, and maximum estimate) based on a empirical-statistical table, as shown in *Table 2*.

Table 2. Scale ranges and numerical EFP assignments.

	Ranges or FP correspondence	Numerical Assignments		
		Min EFP	Avg. EFP	Max EFP
Simple LD	Low complexity ILF/EIF	5	6	7
Average LD	Avg. complexity ILF/EIF	8	9	10
Complex LD	High complexity ILF/EIF	13	14	15
Low Multiple LD	2-4 ILF's/EIF's	14	18	22
High Multiple LD	5-8 ILF's/EIF's	27	39	51
Input fP	EI	4	5	7
Output fP	EO	5	6	8
inQuiry fP	EQ	4	5	7
MF	CRUD	16	18	20
Small F	5-12 estimated primitives	45	56	67
Medium F	13-19 estimated primitives	73	91	109
Large F	20-25 estimated primitives	106	133	160
Small MF	2-3 estimated functions	151	215	280
Medium MF	4-7 estimated functions	302	431	560
Large MF	8-12 estimated functions	603	861	1119

A logical data group (LD) is a group of logical related data, classified on a five-level scale of general complexity: Simple, Average, Complex, Low Multiplicity and High Multiplicity. The first three levels correspond exactly to those considered in standard FP rules, while the other two are for particularly complex macro-files, grouping different actual logical files (*Table 2*). In the actual version of EFP, the difference between internal and external files is neglected, since is not always clear at the estimation stage.

A functional primitive (fP) is the smallest user-significant information process, with autonomy and significance characteristics. fP's are to be classified as input, output, or inquiry; conceptually, these correspond to the elementary processes of the standard FP Analysis, and are the lowest possible level of functional analysis.

A micro-function (mF) is the so-called CRUD set of 4 primitives (working on a specific data group): Create, Read, Update and Delete.

A function (F) is a set of generic primitives, usually an operational sub-system of the application responding to a specific set of users' requirements. F's can be Small, Medium or Large depending on whether they group specific estimated ranges of primitives, (*Table 2*).

A macro-function (MF) is a set of generic functions, usually a relevant, large sub-system of the system, or even a whole application by itself. MF's can be Small, Medium or Large depending on whether they group specific estimated ranges of functions (*Table 2*).

Each identified item is assigned the corresponding set of FP values (minimum, average, and maximum), then the values are summed up to provide the Unadjusted FP estimate (these numerical assignments are subject to improvement on the basis of statistical analysis for actual projects, as from the ISBSG research); as for the most of the other estimation methods, the Adjustment Factor is determined similarly to the standard IFPUG method. Estimates provided by this method may be denoted as "detailed", "intermediate" or "summary", depending of the detail level chosen (or forced) for the early classification of functionalities.

EFP proves quite effective, providing a response within ± 10% of the real FP value in most real cases, while the savings in time (and costs) can be between 50% and 90%, with respect to corresponding standard counts (see [3] and [8] for further details).

3. The Analytic Hierarchy Process (AHP)

The Analytic Hierarchy Process ([6], [7]) provides a means of making decisions or choices among alternatives, particularly where a number of objectives have to be satisfied (*multiple criteria* or *multi-attribute* decision making). The theoretical derivation of the method is here discussed. The reader should have some familiarity with linear algebra; otherwise, a quick way to perform the calculation is suggested at the end of this section. It's worth noting that the method can be applied even if not all theoretical results are known.

Let's assume that *n* items are being considered with the goal of providing and quantifying *judgements* on the *relative weight* (*importance, priority, size*) of each item with respect to all the other items. The first step (*design phase*) set the problem as a hierarchy, where the topmost node is the overall objective of the decision, while subsequent nodes at lower levels consist of the criteria used in arriving at this decision. The bottom level of the hierarchy consists of the *alternatives* from which the choice is to be made, i.e., the *n* items we wish to compare.

The second step (*evaluation phase*) requires pair-wise comparisons to be made between each two items (of the given level of the hierarchy), with respect to their contribution towards the factor from the level immediately above them. The comparisons are made by posing the question 'Of two elements *i* and *j*, which is *more important / larger* with respect to the given factor and how much more?'. The strength of preference is expressed on a ratio scale of 1-9. A preference of 1 indicates *equality* between two items while a preference of 9 (*absolute*

importance) indicates that one item is 9 times larger or more important than the one to which is being compared. This scale was originally chosen, because in this way comparisons are being made within a limited range where perception is sensitive enough to make a distinction [6].
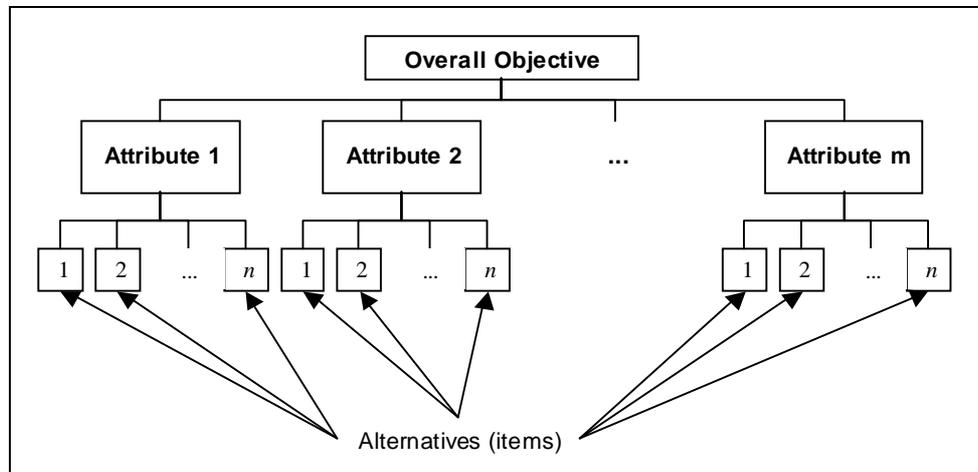

Figure 2. Generic hierarchy scheme.

These pair-wise comparisons result in a *reciprocal n-by-n matrix A*, where $a_{ii} = 1$ (i.e., on the diagonal) and $a_{ji} = 1/\, a_{ij}$ (*reciprocity property*, i.e., assuming that if element i is "x-times" more important than item j, then necessarily item j is "$1/x$-times" more important, or equally "x-times" less important than item i).

Suppose firstly that we provide only the first column of the matrix A, i.e., the relative importance of items 2, 3, .., n, with respect to item 1. If our judgements were completely *consistent*, the remaining columns of the matrix would then be completely determined, because of the transitivity of the relative importance of the items. However we do not assume consistency other than by setting $a_{ji} = 1/\, a_{ij}$. Therefore we repeat the process of comparison for each column of the matrix, making independent judgements over each pair. Suppose that at the end of the comparisons, we have filled the matrix A with the exact relative weights; if we multiply the matrix with the vector of weights $w = (w_1, w_2, …, w_n)$, we obtain:

$$Aw = \begin{bmatrix} a_{11} & a_{12} & \mathbf{L} & a_{1n} \\ a_{21} & a_{22} & \mathbf{L} & a_{2n} \\ \mathbf{M} & \mathbf{M} & & \mathbf{M} \\ a_{n1} & a_{n2} & \mathbf{L} & a_{nn} \end{bmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \mathbf{M} \\ w_n \end{pmatrix} = \begin{bmatrix} w_1/w_1 & w_1/w_2 & \mathbf{L} & w_1/w_n \\ w_2/w_1 & w_2/w_2 & \mathbf{L} & w_2/w_n \\ \mathbf{M} & \mathbf{M} & & \mathbf{M} \\ w_n/w_1 & w_n/w_2 & \mathbf{L} & w_n/w_n \end{bmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \mathbf{M} \\ w_n \end{pmatrix} = n \begin{pmatrix} w_1 \\ w_2 \\ \mathbf{M} \\ w_n \end{pmatrix}$$

So, to recover the (overall) scale from the matrix of ratios, we must solve the problem:

$$Aw = nw, \text{ or } (A\text{-}nI)w = 0,$$

that is a system of homogenous linear equations (I is the unitary matrix). This system has a nontrivial solution if and only if the determinant of *(A-nI)* vanishes, i.e., n is an *eigenvalue* of A. Notice that A has *unit rank* since every row is a constant multiple of the first row and thus all its eigenvalues except one are zero. The sum of the eigenvalues of a matrix is equal to its *trace* and in this case, the trace of A is equal to n. Thus n is an eigenvalue of A and we have a nontrivial solution, unique to within a multiplicative constant, with all positive entries. Usually the normalized vector is taken, obtained by dividing all the entries w_i by their sum.

Thus given the comparison matrix we can recover the scale. In this *exact* case the solution is any column of A normalized. Note also that in the *exact* case A is consistent, i.e., its entries satisfy the condition $a_{jk}=a_{ji}/a_{ki}$ (*transitivity property*). However in the real cases we cannot

give the precise values of w_i/w_j but estimates of them, the *judgements*, which in general are different from the actual weights' ratios. From matrix theory we know that small perturbation of the coefficients implies small perturbation of the eigenvalues. Therefore, we still expect to find an eigenvalue, with value near to n: this will be the *largest eigenvalue* (λ_{max}), since due to the (small) errors in the judgement, also other eigenvalues are different from zero, but still the trace of matrix (n) is equal to the sum of eigenvalues (some of which can be complex).

The solution of the largest eigenvalue problem, i.e., the weight eigenvector w corresponding to λ_{max}, when normalized, gives a unique estimate of the underlaying ratio scale between the elements of the studied case. Moreover, the matrix whose entries are w_i/w_j is still a consistent matrix, and is a consistent estimate of the "actual" matrix A. A itself need not be consistent (for example the judgements could have stated that item 1 is more important than item 2, 2 is more important than 3, but 3 is more important than 1!). It turns out that A is consistent if and only if $\lambda_{max} = n$ and that we always have $\lambda_{max} \geq n$. That's why we take as a "consistency index" (*CI*) the (negative) average of the remaining eigenvalues, which is exactly the difference between λ_{max} and n, divided by the normalizing factor (n-1):

$$CI \equiv \frac{-\sum_{i=2}^{n} l_i}{n-1} = \frac{l_{max} - n}{n-1}, \quad l_{max} = l_1$$

To measure the error due to inconsistency, we can compare the *CI* of the studied case with the average *CI* obtained from corresponding random matrices with order n. *Table 3* shows the random average consistency indexes for various n (*CI$_n$*). Revisions in the pair-wise comparisons are recommended if the consistency ratio (*CR*) between the studied *CI* and the corresponding *CI$_n$* is considerably higher than 10%.

Table 3. Consistency indexes of random reciprocal matrices of order n.

N	1	2	3	4	5	6	7	8	9	10
CI_n	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

This consistency ratio *CR* simply reflects the consistency of the pair-wise judgements and shows the degree to which various sets of importance relativities can be reconciled into a single set of weights. In the above example, (1 larger than 2, 2 larger than 3, and 3 larger than 1) the consistency score would be poor, and would be considered a violation of the axiom of transitivity. AHP tolerates inconsistency through the amount of redundancy of judgements. For a matrix of dimension n only (n-1) comparisons are required to establish weights for the n items. The actual number of comparisons that can be performed in AHP is n(n-1)/2. This redundancy is conceptually analogous to estimating a number by calculating the average of repeated observations: the resulting set of weights is less sensitive to errors of judgement.

A **quick way** to find the weight vector, if one cannot solve exactly the largest eigenvalue problem, is that of normalizing each column in A, and then average the values across the rows: this "average column" is the normalized weight vector w. We then obtain an estimate of λ_{max} dividing each component of Aw (= $\lambda_{max}w$) by the corresponding component of w, and averaging. Finally, we can compute *CI* (and the corresponding *CR*) from this estimate of λ_{max} in order to verify the goodness of the judgments.

So far, we have illustrated the process for only one level in the hierarchy: when the model consists of more than one level then hierarchical composition is used to weight the eigenvectors by the weights of the criteria. The sum is taken over all weighted eigenvector entries corresponding to those in the lower level, and so on, resulting in a global priority vector for the lowest level of the hierarchy. The global priorities are essentially the result of

distributing, or propagating, the weights of the hierarchy from one level to the next level below it. For the purpose of applying AHP to FP estimation, this multi-level weighting is not required, as shown in the following section.

AHP has been already used in the software engineering field in at least two interesting cases: in [9] we see how AHP can help improve the choice of the numerical parameters of the FP software size model, if it has to map exactly the user perception of "quantity of functionality" provided by the software to be measured; in [1] a full hierarchical model is proposed in order to predict the project attributes (mainly effort), using only a single true data and its relative weights with respect to other data.

4. Merging EFP and AHP

The analogy between the hierarchical functional decomposition of EFP and the intrinsic hierarchy of AHP can be very confusing; we must recall that the nodes in different levels in a AHP hierarchy carry very different meaning (going from the objective level, to the attribute level, to the alternative level), while in the EFP approach the decomposition is made only in order to separate different ranges (or groups) of functionality. This means that the elements of a EFP hierarchy are indeed all homogenous with respect to the attribute to be estimated, i.e., the functional size. So there is no strict correspondence between the hierarchical structures in the two techniques, but still a strong tie can be found. Although AHP was developed as a mathematical method for prioritizing the alternatives, we can easily recognize that what we called *importance* (or priority) is just an extensive property as many others, as software functional size, too (or al least as software functional size *should be*, too – see below).

When estimating the software functional size (number of FP), the only criteria is the size itself. Consequently, we can consider a simple AHP hierarchy, with only one level (and the objective "estimated size" above it); the nodes of this level are the n items listed by the estimator, eventually prior to the functional decomposition (limiting ourselves to the process component, this list includes all from possible primitives to macro-functions; a similar reasoning can be easily extrapolated for the data component, too – in fact, this should be done separately, since these data and process components are separated in the FP model, too, thus it could be conceptually very difficult to compare the size of a logical data group with the size of a process, even for a FP expert).

Note that the n "functional items" are only listed (still not classified), and they can be put all together, in despite of their class, in a single AHP level. If we consider the size as the only attribute above this single comprehensive level, we can provide our judgments about the relative size for each pair of the n items in the list, regardless of their effective EFP class. We can therefore apply all the AHP process to this level, in order to reveal the relative average ratios between the items.

As in the regular EFP approach, we should be aware of the fact that we cannot mix in the item list a function, for example, *and* the functional primitives which belong to *that* function. But, in the multi-level approach, it's possible to have a list mixing primitives, micro-functions, functions and even macro-functions (each of those for a different system "branch"). By applying the regular EFP approach, we should exactly assign each item in the list with a specific class, and evaluate the range (small, medium, or large) in case of F or MF. This can now be avoided, since we only have to estimate the relative size between each pair of items, regardless of which class they belong to in the functional decomposition. So we could estimate, for example, that "item (function) i" is as large as (contains as many primitives as) "item j", while "item (function) j" is 10 times "item (primitive) k", and "item k" is nearly 1/12 of (as contained 12 times in) "item i". Note that the scale needs not to grow by integer numbers, and isn't necessarily limited above.

It's noticeable that this proposed approach, i.e., relative weighting of functional items, cannot be performed at the very elementary level of measurement, at least as long as the FP metrics is not able to follow punctually the size for a single data attribute in a file or a single file reference in a process. This means that it would be meaningless to compare numbers of data element types (DETs) or numbers of file types referenced (FTRs), since the number of FP in the actual rules does not increase linearly (regularly) with those attributes.

Thus, we should think of applying the pair-wise comparisons, for the process component, at the level of primitives and higher groupings, as micro-functions, functions and the above. On the data component side, the items to be compared should be logical single and multiple data groups. After having achieved the normalized vector containing the average relative weights between the items, via the AHP technique, we only have to give a single value in FP for only one of the n items, to propagate this size through the relative weights, and to obtain the final estimation. The next section provides some explaining examples of this first combined approach. Section 6 provides further hints to link the EFP and AHP methods, and possible extensions.

5. Numerical examples

Three examples are reported, with a common starting point and some specific differences between them, to investigate the effect of specific changes on the estimation result. In every case we have $n = 5$ items, and we assume that the comparisons made between the 1st item and each of the remaining items (i.e. the first column / row of the comparison matrix A) are the "best" estimates; eventual inconsistency is put in the remaining comparisons (i.e. between 2nd, 3rd, 4th, and 5th item). The common starting point for the three example is as follows:

We assume item 1 as reference unity of the sistem, and the ratios against item 1 are such that the item j is j-times bigger than item 1 (item 2 is twice item 1, item 3 is three times item 1, an so on). If these were the only and exact ratios, the first column of A would contain exactly the entries 1, 2, 3, 4, 5. The only non-zero eigenvalue would be equal to n, i.e. $\lambda = 5$. The corresponding eigenvector is (1,2,3,4,5), not normalized, or (1,2,3,4,5) / (1+2+3+4+5) = (1/15, 2/15, 3/15, 4/15, 5/15), normalized. If item 1 is for example recognized as a primitive corresponding to, let's say, 4 FP, then the overall value would be: (1+2+3+4+5)*4 = **60 (FP)**.

5.1. Case a) Systematic under-estimation

Let's suppose in the first example that the same or another estimator provides the following not consistent, systematic *under-estimated* ratios between the item from 2 to 5, with respect to the first comparisons:

$A_{3,2} = 1$ (instead of 3/2, between item 3, 3-times item 1, and item 2, 2-times item 1)
$A_{4,2} = 1$ (instead of 4/2 = 2, between item 4, 4-times item 1, and item 2, 2-times item 1)
$A_{5,2} = 1$ (instead of 5/2, between item 5, 5-times item 1, and item 2, 2-times item 1)
$A_{4,3} = 1$ (instead of 4/3, between item 4, 4-times item 1, and item 3, 3-times item 1)
$A_{5,3} = 1$ (instead of 5/3, between item 5, 5-times item 1, and item 3, 3-times item 1)
$A_{5,4} = 1$ (instead of 5/4, between item 5, 5-times item 1, and item 4, 4-times item 1)

This corresponds to an average deviation of 22.8% with respect to the expected ratios (if the first estimates were the right ones).

If we now apply the AHP technique, as described in section 3, we easily find:
$\lambda_{max} = 5.076$, $w = (1, 2.956, 3.153, 3.350, 3.547)$, $CR = 1.7\% < 10\%$.

If again item 1 is a primitive corresponding to 4 FP, then the overall estimate is:
(1+2.956+3.153+3.350+3.547)*4 = **56.0 (FP)** ($\Delta = -6.6\%$)

5.2. Case b) Mixed under and over-estimation of the relative weights

In this second example the same or another estimator provides the following not consistent, alternate ratios between items from 2 to 5:

$A_{3,2} = 2\ (> 3/2),\ A_{4,2} = 1\ (< 4/2 = 2),\ A_{5,2} = 3\ (> 5/2)$

$A_{4,3} = 1\ (< 4/3),\ A_{5,3} = 2\ (> 5/3),\ A_{5,4} = 1\ (< 5/4)$

This corresponds to an average (not absolute) deviation of 2.2%, with respect to the expected ratios. We find:

$\lambda_{\max} = 5.123,\ w = (1, 2.173, 3.284, 3.567, 5.252),\ CR = 2.75\% < 10\%$.

If item 1 is a primitive corresponding to 4 FP, then the overall value is:

$(1, 2.173, 3.284, 3.567, 5.252)*4 = $ **61.10 (FP)** $(\Delta = 1.8\%)$.

In this case the under and over-estimated ratios are balanced.

5.3. Case c) Mixed case, with stronger inconsistency

In this third example the same or another estimator provides the following *strongly* not consistent ratios between items from 2 to 5:

$A_{3,2} = 1\ (< 3/2),\ A_{4,2} = 1\ (< 4/2 = 2),\ A_{5,2} = 1\ (< 5/2)$

$A_{4,3} = 4\ (>> 4/3),\ A_{5,3} = 6\ (>> 5/3),\ A_{5,4} = 5\ (>> 5/4)$

This corresponds to an average (not absolute) deviation of -61.67%, with respect to the expected ratios. We find:

$\lambda_{\max} = 5.1,\ w = (1, 3.242, 1.808, 3.45, 7.816),\ CR = 14.5\% > 10\%$.

If item 1 is a primitive corresponding to 4 FP, then the overall value is:

$(1, 3.242, 1.808, 3.45, 7.816)*4 = $ **69.3 (FP)** $(\Delta = 15.4\%)$.

Note that the value of CR suggest us that the comparisons should be revised, but the deviation between the expected and the estimated values could be considered still acceptable in mostly real situations.

6. Further discussion and conclusion

The examples above are very encouraging, but much investigation has still to be made. For example, very large cases (very high n) introduce difficulties in managing the items. From this perspective, is noticeable that the original AHP deals only with small n; a suggestion is to try to use homogenous clusters of items, and to make comparisons between these clusters.

However, the fact that only a single value is to be provided, besides of the relative weight estimates, does not mean that more than one true value cannot be used: obviously, if we know the value of items 1, 2 and 3, we also have more confidence in fixing more weights in the comparison matrix; *de facto*, in this way we *use* the richer information.

A strong possibility, which is worth of further investigation, is that of using a so-called Experience Data Base (or "Catalogue of Typical Elements"). A catalogue could contain for example all the elementary processes that may be identified in a generic software, and their average number of FP. In [2] is suggested to restrict the vocabulary used to describe the software, in order to quicken the transaction identification. One single, well-defined verb should describe each process (examples are: "update" as a primitive input, "calculate" as a primitive output, "read" (retrieving without updating) as a primitive inquiry, and so on). Once some of these typical elements are identified among the list of items, the comparison matrix would greatly benefit of the relative ratios between them. A generic case of "typical element" is the already cited micro-function, since the CRUD is very often denoted as "Management of [Logical Data Group]". Moreover, one could intentionally pick up from the catalogue just some clear examples of some classes of processes, as for example a primitive, a small function, a medium function, and a large function. Since the FP assignments for these elements are well-

known from the EFP method, we can set up the estimate comparing all the unknown items with respect to these known elements. Particularly, if a item to be estimated "falls" between two known elements (one being smaller, the other larger), the redundant correct relative weight estimation for that element has a positive influence on the overall estimate via the AHP approach.

There has also been, in the latest years, a significant evolution of AHP: the Analytic Network Process (ANP) has been developed with the capability to model *non-linear* relations between nodes. ANP allows the decision maker to leap beyond the traditional hierarchy, dealing with interdependencies such as feedback and dependencies among nodes, too. Although software metrics should aim to be linear, analysis of interdependencies with ANP could be useful in extending the proposed approach to catch the relationships between data and processes.

AHP is a powerful means for several tasks in the estimation and decision making field. The proposed combination with the EFP technique can solve those situation in which the only EFP does not provide good results, especially due to atypical or new situations, not collected in the historical statistics. Moreover, and finally, this approach can easily be inserted, as one of several sub-levels, in a more global use of the AHP for project prioritization and estimation, as depicted in [1]. This is also possible because the process is able to deal with both tangible and intangible factors, rescaling all factors to a common scale.

7. References

[1] Barker, S., Shepperd, M. and Aylett, M., "The analytic hierarchy process and almost dataless prediction", ESCOM-SCOPE 99, April 26-29, 1999, Herstmonceux Castle, Sussex, UK.

[2] Frallicciardi L. and Natale D., "Estimating Size Using Function Point Analysis and Analogy", ESCOM 97, May 1997, Berlin.

[3] Meli, R., "Early and Extended Function Point: a new method for Function Points estimation", IFPUG Fall Conference, September 15-19, 1997, Scottsdale, Arizona.

[4] Meli, R., "Human factors and analytical models in software estimation: an integration perspective", ESCOM-SCOPE 2000, April 18-20, 2000, Munich.

[5] Meli, R. and Santillo, L., "Function Point Estimation Methods: a Comparative Overview", FESMA 99, October 6-8, 1999, Amsterdam.

[6] Saaty, T.L., "The Analytic Hierarchy Process", McGraw Hill, New York, 1980.

[7] Saaty, T.L. and Alexander, J.M., "Thinking with Models", Pergamon Press, 1981.

[8] Santillo, L. and Meli, R., "Early Function Points: some practical experiences of use", ESCOM-ENCRESS 98, May 27-29, 1998, Rome.

[9] Wittig, G.E., Morris, P.M., Finnie, G.R. and Rudolph, E.E., "Formal Methodology to Establish Function Point Coefficients", IFPUG Fall Conference, September 15-19, 1997, Scottsdale, Arizona.