Predicting Maintenance Effort with Function Points

Frank Niessink Hans van Vliet

Faculty of Mathematics and Computer Science, Vrije Universiteit Amsterdam
De Boelelaan 1081, 1081 HV, Amsterdam, The Netherlands
E-mail: {F.Niessink, J.C.van.Vliet}@cs.vu.nl

Abstract

Function Point Analysis (FPA) is a well-known method to measure the functionality of a system, from the user's point of view. Both Albrecht's original model and a local variant we studied assume that effort is primarily related to the size of a change. Analysis of data gathered on a major system over a period of 18 months does not confirm this relation. Rather, our data suggests that the size of the component to be changed has a much larger impact on effort than the size of the change itself. Furthermore, the various corrective factors of the function point model do not help to improve effort estimates in the environment we studied. Finally, we found that expert estimates outperform the function point estimates, while analogy-based estimates slightly outperform the expert estimates.

Keywords

Maintenance Effort Prediction, Function Points, Maintenance Function Points, Analogy-based Estimation.

1 Introduction

Function Point Analysis (FPA) is a well-known method to measure the functionality of a system, from the user's point of view. It does so by calculating the number of function points of a system in a two-step process:

1. The functional size of a system is calculated by assigning a weight to each *individual* function. The sum of these weights is termed the Unadjusted Function Points (UFP).
2. At the level of the *complete* system, a number of predefined application characteristics, such as processing complexity and transaction rate, result in a Value Adjustment Factor VAF.

Multiplying UFP and VAF yields AFP: the Adjusted Function Points.

The version of the FPA-model generally used is the one published in 1983 [4]. Later refinements concern clarification of the counting rules, not the structure of the model.

Albrecht claims that function points are 'an effective relative measure of function value delivered to our customer' [3]. Various other researchers have also found a strong relation between the number of function points and work effort; see e.g. [12, 5, 13, 10, 2]. FPA is a popular method for development effort prediction, even though various researchers have criticized the underlying methodology [17, 8].

Models to predict maintenance effort generally use Lines Of Code (LOC) as the primary size-related factor influencing maintenance effort [14, 11]. Estimating the size of a change in terms of LOC added/deleted/updated however may well be as difficult as estimating the size (in LOC) in development projects. The latter is known to be notoriously difficult, and results of using LOC as the primary means to predict development costs have been mixed, to say the least. Jørgensen [11] assumes 'that the maintenance task size can be reasonably accurately predicted and that the use of our LOC based size measure is meaningful.' He did consider Function Points as a candidate size measure, but noted various disadvantages: Function Points are not very meaningful on small maintenance tasks, change oriented tasks and on maintenance tasks not concerned with user functionality. The latter is obviously true. We do not know whether the other claimed disadvantages are true or not.

As part of the SoftCalc model for estimating maintenance costs [16], Sneed proposes the use of Function Points to measure both the size of the application and the size of the change. However, the article does not present a validation of the model with empirical data. The only other publication we know of which describes an empirical assessment of the relation between maintenance effort and (variants of) FPA is [1]. See also section 2.

Under the auspices of NESMA, the Dutch branch of the International Function Point User Group (IFPUG), a variant of FPA has been developed which is aimed at predict-

ing *maintenance* effort. A variant of this variant is used by the Dutch Ministry of Transport, Public Works and Water Management as the primary basis for billing the client organization for maintenance efforts. Data on 140 maintenance tasks carried out from March 1995 to September 1996 are analyzed in this paper.

The remainder of this paper is organized as follows. Section 2 introduces the FPA variant used to model maintenance effort. Section 3 describes the organization and system for which data has been gathered. Section 4 investigates the relations between maintenance function points, expert estimates for maintenance tasks and actual hours spent. Section 5 gives our conclusions.

2 From Development Function Points to Maintenance Function Points

To obtain the Unadjusted Function Points of a system, Albrecht distinguishes 5 function types and 3 complexity levels for each function type. The 5 function types are: External Input, External Output, External Inquiry, Internal Logical File and External Interface File. The first three are transaction function types, and the last two are data function types. The UFP associated with an individual function depends on the type of the function and the complexity level associated with that function; see table 1. For transaction functions, the complexity level is determined by the number of file types referenced (FTR) and the number of data element types of those files (DET). For data functions, the complexity level is determined by the number of record element types (RET) and the number of data element types of the Internal Logical File or the External Interface File in question. As an example, table 2 indicates the complexity levels for Input Functions.

	Complexity level		
Function type	Low	Average	High
External Input	3	4	6
External Output	4	5	7
External Inquiry	3	4	6
Internal Logical File	7	10	15
External Logical File	5	7	10

Table 1: Function types and their weights

Albrecht's general formula for determining the number of Function Points in a development project already takes into account that such a project generally builds on existing functionality [4]:

$$AFP = UFP_{deleted} \times VAF_{pre} + (UFP_{changed} + UFP_{added}) \times VAF_{post} \quad (1)$$

File Types	Data Element Types (DET)		
Referenced (FTR)	1-4	5-15	> 15
0-1	Low	Low	Average
2	Low	Average	High
> 2	Average	High	High

Table 2: Complexity levels for Input Functions

where

- AFP = Adjusted Function Points for the new system,
- $UFP_{deleted}$ = Unadjusted Function Points deleted from the system,
- $UFP_{changed}$ = Unadjusted Function Points changed in the system, as expected at completion of the project,
- UFP_{added} = Unadjusted Function Points added to the system,
- VAF_{pre} = Value Adjustment Factor of the system at the start of the project,
- VAF_{post} = Value Adjustment Factor of the system at project completion.

Though one might be tempted to use this formula for maintenance tasks too, there are at least two reasons for not doing so:

1. removing functionality from a system will most likely be cheaper than developing that same functionality, and
2. adding a small amount, say, n Function Points, to a large function of, say, N Function Points, will most likely be cheaper than adding N Function Points to a function of size n.

The Dutch Function Point User Group NESMA developed a variant of the above formula to remedy these effects. In this model, the various Unadjusted Function Point counts are multiplied with a Maintenance Impact Ratio (MIR), indicating the relative impact of a change. Our data is based on a variant of this variant. The latter is described below, as far as needed to discuss the dataset and its analysis.

Obviously, MIR equals 1 for functions added to the system. For functions deleted from the system, MIR is set at 0.2 in our environment. To determine MIR for changed functions, a scheme similar to that for the complexity levels of functions is used. For example, when a transaction type function is to be changed, that change will affect a subset of the file types referenced and their data element types. If the change involves a small subset of FTR and DET, it is

reasonable to assume the change to cost a small fraction of the initial development cost. If the change involves a large fraction of FTR and/or DET, then that change may cost up to the initial development cost. The situation is even worse in the latter case, since such a change not only incurs an effort approximating that for the initial development, but an extra effort to undo (remove) the original function as well.

In our environment, a 5-point scale is used for MIR-levels for transaction type functions. The resulting scheme for determining MIR is given in table 3. The mapping to numerical values is given in table 4. Here, %FTR denotes the percentage of file types changed, i.e. %FTR = ($FTR_{changed}$ / FTR_{pre}) × 100%. The percentage of data element types changed (%DET) is defined in a similar way. Multiplication of UFP with MIR yields the number of Unadjusted Maintenance Function Points (UMFP).

Abran [1] found that most adaptive changes are small. To handle these, he proposes a scheme in which the lower complexity levels are further refined. This essentially amounts to applying a Maintenance Impact Ratio, part of the time.

For the situation we are considering, many of the Value Adjustment Factors distinguished in [4] do not apply, simply because we are considering one system only, so that their value is the same for each proposed change. We did retain three of the Value Adjustment Factors, and added a new one. More importantly, the possible influence of these factors is taken into account at the level of an individual transaction or data function, rather than the complete system. So, for example, the complexity of a given function influences the effort to change *that* function; the complexity of other functions, or the system as a whole, does not. The possible effect of our Value Adjustment Factors is simply multiplicative, as for instance in the COCOMO-model [6], and occasionally additive.

The following Value Adjustment Factors are distinguished:

1. Functional Similarity. This is the reusability factor of [4], though with a quite different semantics. If a change request incurs similar changes in other functions, it is reasonable to assume a copying effect from the first such change to the next ones. In that case, all but the first such change get a multiplicative adjustment of 0.75.

2. Performance Sensitivity. A function is considered performance sensitive if it uses a sufficiently large database table in at least one location. In that case, a multiplicative adjustment of 1.15 is used.

3. Complex Function. We distinguish two levels of complexity: complex and very complex. If a complex (very complex) function is added, such incurs a multiplicative adjustment factor of 1.5 (3). If a complex (very complex) function is changed, the additive adjustment is 3 (12).

4. Standard Functionality. Some changes can be easily accommodated with, because they map well onto the primitives of the 4GL being used. If these cases, a multiplicative adjustment factor of 0.5 is used. This factor is not present in [4].

3 The FAIS System

FAIS is a large financial information system, custom-developed for the Dutch Ministry of Transport, Public Works and Water Management. FAIS is written in a 4GL (Uniface) and uses the Sybase DBMS. The size of the system is approx. 14,000 Function Points. It has been in operational use since the summer of 1994. FAIS is used at 28 locations spread over the country. It has 1200 end users.

The FAIS support organization has three functional units:

- Customer Contacts handles all contacts with the users of the system. This unit develops user specifications for change requests, and takes care of user training.

- Functional Maintenance does the design, implementation and testing of all changes of the system. Corrective maintenance and small changes are handled by one sub-unit (the helpdesk); planned maintenance and new releases are handled by another sub-unit.

- Technical Maintenance takes care of the technical infrastructure and installs new releases.

The maintenance data we are analyzing in this paper concerns planned (i.e. non-corrective) maintenance activities. An incoming change request is first analyzed by Customer Contacts. This results in one or more change proposals at the level of a FAIS-function. A FAIS-function is a unit of application functionality which can be separately invoked from a menu by the user of the system. Data is collected at the level of a change to a single FAIS-function: actual time spent to design, implement and test the change, an expert estimate of the change effort, and the number of maintenance function points of the change.

Customer Contacts and Functional Maintenance independently determine the number of Maintenance Function Points for each change to a FAIS-function. They have to come to an agreement about this number as well as the description of the change, which are then both frozen. The Maintenance Function Points are next used to bill the client

%FTR	%DET				
	$\leq 25\%$	$\leq 50\%$	$\leq 75\%$	$\leq 100\%$	$> 100\%$
$\leq 25\%$	Very Small	Small	Average	Large	Very Large
$\leq 50\%$	Small	Average	Large	Very Large	Very Large
$\leq 75\%$	Average	Large	Very Large	Very Large	Very Large
$\leq 100\%$	Large	Very Large	Very Large	Very Large	Very Large
$> 100\%$	Very Large	Very Large	Very Large	Very Large	Very Large

Table 3: MIR levels for transaction type functions

MIR-level	Very Small	Small	Average	Large	Very Large
Value	0.125	0.25	0.5	0.75	1.2

Table 4: MIR values for transaction type functions

organization. The actual planning of maintenance activities by Functional Maintenance is based on an expert estimate per change to a FAIS-function. This expert estimate is made by Functional Maintenance. We do not know the extent to which the Function Point estimates have influenced the expert estimates. The opposite influence is unlikely, given the detailed counting guidelines that have been used.

3.1 The dataset

As noted in section 2, the function points measurement process includes a data function type measurement and a transaction function type measurement. Since FAIS does not interface other systems, the number of External Interface Files is 0. Changes to Internal Logical Files often span more than one FAIS function, or even more than one change request. Such changes may moreover involve data conversions. Though data on this type of change has been collected, they are at a level which is incompatible with the level at which function points are counted. Our analysis is therefore restricted to the transaction function types.

One transaction function changed	90
Multiple transaction functions changed	50
Number of changes	140
Total hours spent	6655
Total of expert estimates	6765
Total AMFP	648.88
Hours spent per AMFP	10.26

Table 5: Global information on dataset

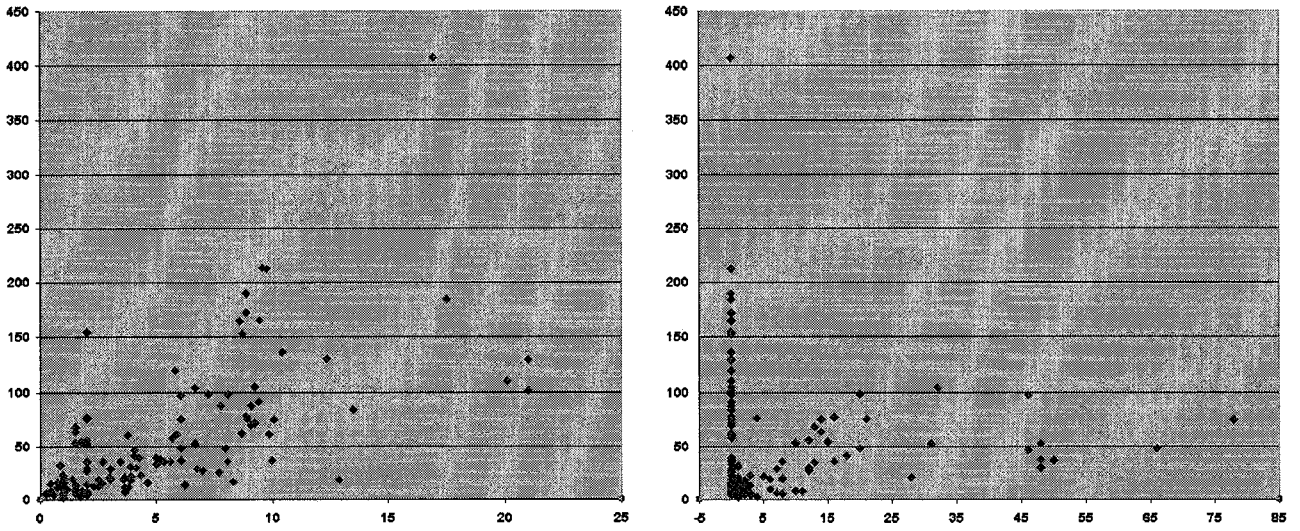
Our initial dataset has 327 entries at the level of a FAIS-function. For 175 entries, actual hours spent have not been recorded at this level. These mostly concern changes which take little time individually, such as the removal of one function. For 12 entries, the data were incomplete. The remaining 140 data points are used in the analysis below; of these, 90 concern a change in one transaction function, while 50 concern more than one changed transaction function. See table 5 for some global information on this data set and figure 1(a) for a scatter plot of effort versus Adjusted Maintenance Function Points (AMFP).

4 Analysis

In this section we will examine the suitability of our Maintenance Function Point (MFP) model to estimate the effort needed to implement change requests. We will compare the quality of the predictions of the MFP-model with the quality of the expert estimates from the dataset. Also, we will decompose the MFP to determine the contribution of model components to the estimate (cf. [2]). Finally, we will use analogy-based estimation as another means of assessing the suitability of the dataset for estimation.

There are several criteria to evaluate the predictions of a model [7]. The *coefficient of multiple determination* (R^2) is used to indicate the amount of variance that is accounted for by the independent variables in a linear model. Because R^2 tends to be an optimistic estimate of how well the model fits the population, we also provide the *adjusted* R^2 which compensates for the number of independent variables in the model.

We also use the Mean Magnitude of the Relative Error (MMRE) to indicate the relative amount by which the predictions over- or underestimate the real value. We use PRED(25) to indicate how many of the predictions lie within 25% of the real value. Conte et al. use the simul-



(a) AMFP (x-axis) vs. Work-hours

(b) $DET_{changed}$ (x-axis) vs. Work-hours

Figure 1: Effort scatter plots

taneous satisfaction of the two measures

$$\text{MMRE} \leq 25\% \text{ and PRED(25)} \geq 75\% \qquad (2)$$

as a criterion for acceptable model performance.

4.1 Assessing the maintenance function point estimates

First, we compare the expert estimates with those from the Maintenance Function Point (MFP) model. To predict the Work Effort (WE_{pred}) for change requests using maintenance function points we use the formula WE_{pred} = MFP-ratio $\times$ AMFP. We calculate the MFP-ratio using the total number of adjusted maintenance function points (AMFP) for this dataset and the total number of hours spent working on these 140 change requests, see table 5. The MFP-ratio for this dataset is 10.26 hours per AMFP. We compare the estimates of the MFP model with the expert estimates in table 6.

It is clear that the expert estimates outperform the MFP model on all quality figures. In the next sections we will explore whether other combinations of model components or other estimation techniques improve the results.

4.2 Models based on function point components

As described in section 2, the main steps taken when computing the number of AMFP for a change request are:

$$\text{UFP} \xrightarrow{\times \text{MIR}} \text{UMFP} \xrightarrow{\times \text{VAF}} \text{AMFP} \qquad (3)$$

We would expect the R^2 to increase in each of these steps as extra information is added to the maintenance function point count. Table 7 displays the regression models for UFP, UMFP and AMFP.

The values of R^2 suggest that inclusion of MIR in the model makes matters even worse. To find a possible reason for this behavior, we will have to consider the more detailed attributes that make up the value of MIR, like $DET_{changed}$. We can only do so for changes involving one transaction type function. For changes involving multiple transaction functions, the type of change may differ between the transaction functions that constitute the change. This reduction leaves us with 90 data points, 63 of which concern a change to one function, 19 concern the addition of a function and 8 belong to a rest category (such as the deletion of a function). We perform the regression analysis again on the two subsets for the same variables as in table 7, see table 8.

Note that we calculated the regression models for new transaction functions twice. Using the 19 new transaction function the regression analysis is quite unsuccessful. If we look at the work-hours for these 19 change requests, we observe that there is one outlier. One change request costs about four times the work-hours of any of the other 18 cases. Though the dataset with new transaction functions is too small to perform further analysis, the figures in table 7 do suggest that something is wrong in the estimates for changed functions. Because this set is sufficiently large (63 cases) we may try to improve the models by investigating alternatives. The fact that the R^2 for the regression

Equation	Std. err.	R^2	R^2 adj.	MMRE	PRED(25)
Expert Estimates	-	0.66	-	57%	39%
$WE_{pred} = 10.26 \times AMFP$	-	0.46	-	91%	22%

Table 6: Comparing the estimates

Equation	Std. err.	R^2	R^2 adj.	MMRE	PRED(25)
$WE_{pred} = 7.49 \times UFP - 15.06$	44.08	0.40	0.39	110%	26%
$WE_{pred} = 7.98 \times UMFP + 10.86$	47.27	0.31	0.30	149%	22%
$WE_{pred} = 8.96 \times AMFP + 6.01$	41.98	0.46	0.45	104%	22%

Table 7: Regression models for UFP, UMFP and AMFP

Changed transaction functions ($n = 63$)					
Equation	Std. err.	R^2	R^2 adj.	MMRE	PRED(25)
$WE_{pred} = 6.16 \times UFP - 9.82$	22.42	0.13	0.12	117%	19%
$WE_{pred} = 3.63 \times UMFP + 16.28$	22.77	0.10	0.09	126%	22%
$WE_{pred} = 3.25 \times AMFP + 16.85$	22.84	0.10	0.08	121%	19%
New transaction functions ($n = 19$)					
Equation	Std. err.	R^2	R^2 adj.	MMRE	PRED(25)
$WE_{pred} = 4.15 \times UFP + 4.01$	34.62	0.02	-0.04	130%	21%
$WE_{pred} = 4.15 \times UMFP + 4.01$	34.62	0.02	-0.04	130%	21%
$WE_{pred} = 3.12 \times AMFP + 13.70$	34.44	0.03	-0.03	120%	21%
New transaction functions ($n = 18$)					
Equation	Std. err.	R^2	R^2 adj.	MMRE	PRED(25)
$WE_{pred} = 7.78 \times UFP - 19.80$	8.16	0.51	0.48	58%	28%
$WE_{pred} = 7.78 \times UMFP - 19.80$	8.16	0.51	0.48	58%	28%
$WE_{pred} = 5.39 \times AMFP - 0.38$	6.31	0.71	0.69	47%	28%

Table 8: Regression models for changed and new transaction functions

model with UFP only is as low as 0.13 is not very surprising: we do not expect the work-hours needed to implement a change in a function to correlate well with the number of (development) function points of that function. We do expect, however, a fair correlation between work-hours and UMFP. The fact that the R^2 for UMFP is even lower at 0.10 suggests a flaw in the mapping from UFP to UMFP.

To further examine this, we try to construct a new MIR that will improve the regression model for UMFP. In the current MFP model, MIR is computed from a table which has as its inputs the percentage of DET changed and the percentage of FTR changed, see tables 3 and 4. We try to construct a better MIR by performing a stepwise regression on the components that make up MIR: $DET_{changed}$, DET_{pre}, $FTR_{changed}$ and FTR_{pre}. So we are trying to find the model of the form:

$$WE_{pred} = (\beta_0 + \beta_1 \times DET_{changed} + \beta_2 \times DET_{pre} + \beta_3 \times FTR_{changed} + \beta_4 \times FTR_{pre}) \times UFP \quad (4)$$

Doing so yields the following model for MIR′:

$$MIR' = 0.19 \times DET_{changed} + 0.02 \times DET_{pre} + 2.03 \quad (5)$$

Using this new MIR we recompute UMFP and AMFP – resulting in UFMP′ and AMFP′. If we use UMFP′ and AMFP′ on the subset of 63 changed functions we get the results as presented in table 9.

We now have a definite improvement in the step UFP $\xrightarrow{\times MIR}$ UMFP. Comparing table 3 with equation 5, we observe a striking difference. Whereas our original model assumes that effort is proportional to the relative size of a change, equation 5 suggests effort is proportional to the size

Changed transaction functions ($n = 63$)					
Equation	Std. err.	R^2	R^2 adj.	MMRE	PRED(25)
$\text{WE}_{\text{pred}} = 1.23 \times \text{UMFP}' - 3.88$	15.78	0.57	0.56	71%	21%
$\text{WE}_{\text{pred}} = 1.04 \times \text{AMFP}' + 1.27$	16.81	0.51	0.50	73%	32%

Table 9: Regression models with MIR′

of the component changed. This observation is in line with results reported in the literature (see, e.g. [11]): maintenance effort is highly correlated with size.

The step UMFP $\xrightarrow{\times \text{VAF}}$ AMFP, however, still does not improve R^2. Our attempts to improve VAF by considering its constituents have been in vain. Regression analysis on individual factors and combinations thereof yield quite different parameter values, suggesting that one factor acts as a correction to the other, rather than as an improvement of the model as a whole. This fits in with the observation that these adjustment factors are highly negatively correlated.

4.3 Analogy-based estimation

Analogy-based estimation [15] is an estimation technique in which we make predictions based on a few historical cases from a larger dataset. These historical cases act as analogies for the one for which we are making the prediction. Using the tool Angel (see [15]) we can automate the search process for the analogies. Angel determines analogies by computing the Euclidean distance between cases.

To assess the suitability of the dataset for making analogy-based estimations Angel uses jack-knifing; one by one each case is taken from the dataset and an estimate is determined using the remainder of the dataset. The estimates together determine the MMRE and PRED(25) quality figures. This process is repeated for each combination of attributes, yielding the best combination together with the quality figures for that combination.

If we let Angel assess our dataset using the maintenance function point components (FTR_{post}, FTR_{pre}, $\text{FTR}_{\text{changed}}$, DET_{post}, DET_{pre}, $\text{DET}_{\text{changed}}$, UFP, UMFP, MIR, VAF, AMFP), we obtain the following result:

Variables used for prediction	MMRE	PRED(25)
$\text{FTR}_{\text{changed}}$, $\text{DET}_{\text{changed}}$, DET_{pre}, UMFP, VAF, AMFP	39.8%	38%

The values for MMRE and PRED(25) are much better than those for the regression models in tables 6 and 7. A possible explanation is that the dataset is rather heterogeneous. The scatterplot in figure 1(b) illustrates this. Plots for other variables show similar patterns. If the number of attributes is sufficiently large and at the same time discriminates the cases well, analogy-based estimating is a promising alternative to more rigid regression-based models.

5 Conclusions

In this paper we used various FPA variants to predict maintenance effort. We started off with a variant which, like Albrecht's original model, assumes that maintenance effort is strongly determined by the size of a change. The performance of the resulting model proved to be rather poor on the dataset we studied.

Further analysis revealed one particularly weak spot. In our environment the size of a component changed has a much stronger impact on the effort needed than the size of the change itself. The general form of the relation between effort and the size of the component/change is better described as

$$\text{Effort} \approx \text{constant} \times \text{size of the component} \times (1 + \epsilon \times \text{size of change}) \quad (6)$$

rather than as

$$\text{Effort} \approx \text{constant} \times \text{size of change} \quad (7)$$

On hindsight, this fits in well with other models that use size as the primary factor influencing maintenance effort.

The steps we have taken somehow constitute a calibration of the *structure* of the MFP model, rather than a calibration of its parameters only. Whether the need for such a structural calibration is caused by the particular organization we have been studying, or has more general merit, is still an open question.

Notwithstanding the improvements we have been able to make to the initial model, the results are still rather unsatisfactory. None of the presented models satisfies the criterion for prediction models mentioned in equation 2. Whether such is partly caused by the omission of relevant factors, such as the age of a component or its complexity, is not known. We have simply no other attributes available than those collected for the MFP model. We nonetheless suspect that the heterogeneity of the dataset remains in conflict with the 'one model fits all' flavor of regression-type models. In such cases analogy-based estimation seems to offer an interesting alternative.

Acknowledgements

This research was partly supported by the Dutch Ministry of Economic Affairs, project 'Concrete Kit', nr. ITU94045. Partners in this project are Cap Gemini, Twijnstra Gudde and the Technical Universities of Delft and Eindhoven. We furthermore acknowledge the support of Diederik Verwoerd, Peter Nooteboom, and Douwe Schumacher of the FAIS support organization at the Ministry of Transport, Public Works and Water Management.

References

[1] Alain Abran and Marcela Maya. A Sizing Measure for Adaptive Maintenance Work Products. In ICSM [9].

[2] Alain Abran and Pierre N. Robillard. Function Point Analysis: An Empirical Study of Its Measurement Processes. *IEEE Transactions on Software Engineering*, 22(12):895–910, December 1996.

[3] A.J. Albrecht. Measuring Applications Development Productivity. In *Proceedings Application Development Symposium*, pages 83–92. SHARE/GUIDE, 1979.

[4] A.J. Albrecht and J.E. Gaffney. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Transactions on Software Engineering*, 9(6):639–648, 1983.

[5] R.D. Banker and C.F. Kemerer. Scale economies in new software development. *IEEE Transactions on Software Engineering*, 15(10):1199–1205, 1989.

[6] B. Boehm. *Software Engineering Economics*. Englewood Cliffs, N.J: Prentice-Hall, 1981.

[7] S.D. Conte, H.E. Dunsmore, and V.Y. Shen. *Software Engineering Metrics and Models*. The Benjamin/Cummings Publishing Company, Inc., 1986.

[8] Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. Int. Thomson Computer Press, second edition, 1997.

[9] IEEE Computer Society. *Proceedings of the International Conference on Software Maintenance (ICSM '95)*, Nice, France, October 16-20 1995.

[10] Ross Jeffery and John Stathis. Function Point Sizing: Structure, Validity and Applicability. *Empirical Software Engineering – An International Journal*, 1(1):11–30, 1996.

[11] Magne Jørgensen. An Empirical Study of Software Maintenance Tasks. *Software Maintenance: Research and Practice*, 7:27–48, 1995.

[12] C.F. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30(5):416–429, 1987.

[13] C.F. Kemerer and B.S. Porter. Improving the Reliability of Function Point Measurement: An Empirical Study. *IEEE Transactions on Software Engineering*, 18(11):1011–1024, 1992.

[14] H. Dieter Rombach, Bradford T. Ulery, and Jon D. Valett. Toward Full Life Cycle Control: Adding Maintenance Measurement to the SEL. *The Journal of Systems and Software*, 18(2):125–138, May 1992.

[15] Martin Shepperd, Chris Schofield, and Barbara Kitchenham. Effort Estimation Using Analogy. In *Proceedings of the 18th International Conference on Software Engineering*, pages 170–178, Berlin, Germany, March 1996. IEEE Computer Society Press.

[16] Harry M. Sneed. Estimating the Costs of Software Maintenance Tasks. In ICSM [9], pages 168–181.

[17] C.R. Symons. Function point analysis: difficulties and improvements. *IEEE Transactions on Software Engineering*, 14(1):2–11, 1988.