Function point analysis : evaluation of a software cost estimation model

Function point analysis: evaluation of a software cost estimation model

F. J. HEEMSTRA and R. J. KUSTERS

University of Technology, Department of Industrial Engineering and Management Science, Section Management Information Systems and Automation, Research Group Estimation and Control of Software Development, Post Box 513, 5600 MB Eindhoven, The Netherlands

The merits of function point analysis are discussed. Function point analysis was chosen because it is one of the most widely used aids for software cost estimation. In the analysis we use data from a large survey of Dutch organizations, from an experiment on the effectiveness of software cost estimation models and from a field study aimed at the adjustment part of the FPA model. Conclusions show that it is indeed widely used and that it performs reasonably well as a product sizer, but that the adjustment part of the model is less useful.

Introduction

In this paper we present some empirical data on the usefulness of function point analysis (FPA), a method used for estimating and controlling software projects.

An extensive investigation by the University of Arizona gives figures for overruns (Phan *et al*, 1988). Table 1 shows some results of this research.

Table 1 Overruns of costs and duration (Phan *et al*, 1988)

Degree of overruns	*Percentage of respondents (N = 191)*	
	Overruns' cost (%)	*Overruns' duration (%)*
Always or often	41.0	32.9
Sometimes	42.5	49.6
Seldom or never	16.5	17.5
	100.0	100.0

In other publications, problems with software cost estimation and control (SCE) are indicated with figures from everyday practice (Jenkins *et al*, 1984; Thambain & Wilemon, 1986; van Genuchten & Fierst van Wijnandsbergen, 1986; van Lierop *et al*, 1991). It is alarming that it is so difficult for organizations to control the development of software.

Given these problems, it is interesting to look at the methods provided to alleviate them. Among these methods, SCE models and related tools have often been discussed in the literature. We will focus on one of the most popular of these tools, namely FPA. On the basis of empirical data we attempt to answer the following questions:

- Is FPA actually used in practice?
- How is FPA used in practice (by whom, in what stage of software development, what kind of applications, and so on)?
- How good are the estimates made with FPA?
- Are models based on function points better then models based on lines of code?
- How useful are the FPA adjustment characteristics?

The data comprises:

- A comprehensive survey of software cost estimation in Dutch organizations.
- An experiment regarding the use of software cost estimation models.
- A field study of the applicability of the FPA cost drivers.

We start the next section with a general introduction to SCE models and more specifically to FPA. The rest of the paper deals with the evaluation of FPA. In three subsequent sections the three empirical studies are described. In the first of these the results of a survey on software cost estimation are presented and the results pertaining to FPA are highlighted. Next, in the second of these we discuss the usefulness of function points as sizers. In this discussion, use is made of the results of a field study on the effectiveness of software cost estimation models. In the third of these we look at one specific, and from our perspective weak, aspect of the FPA model using results from the field study. The paper ends with conclusions and recommendations.

Received. 4 April 1991 Accepted. 8 October 1991
© 1991 Operational Research Society Ltd

Software cost estimation models and function point analysis

Software cost estimation models

In the literature one can find a large number of methods for determining expected software development costs. Most of them are a combination of the following methods that were first described by Boehm (1981):

1. estimates made by an expert,
2. estimates based on reasoning by analogy,
3. estimates based on price-to-win,
4. estimates based on available capacity,
5. estimates based on the use of parametric models.

Only methods 1, 2 and 5 provide estimates for the development costs. We will concentrate on the last method. Most software cost estimation models used nowadays are two-stage models (Heemstra, 1989). The first stage is a sizer and the second provides a productivity adjustment factor.

In the first stage, in one way or another, an estimate regarding the size of the product to be developed is obtained. In practice, several sizing techniques are used. The most well-known sizers currently are function point analysis (Albrecht & Gaffney, 1983) and lines of code (Boehm, 1981). But other sizing techniques such as DeMarco's function weight method (DeMarco, 1982, 1984) and SEER SSM (Theta Analysis & Systems, 1990) are used. The result of a sizing model is the size/volume of the software to be developed, expressed in the number of lines of source code, statements or function points.

In the second stage it is estimated how much time and effort it will cost to develop the software of the estimated size. First of all, the estimate of the size is converted into an estimate in nominal man-months of effort. Since this nominal effort takes no advantage of knowledge concerning the specific characteristics of the software product, the way the software product will or can be developed, and the production means, a number of cost-influencing factors (cost drivers) are added to the model. The effect of these cost drivers must be estimated. This effect is often called a productivity adjustment factor. Application of this correction factor to the nominal estimation of effort provides a translation into a more realistic estimate.

Some models, like FPA, are focused more on the sizing stage. Others, like the well-known COCOMO model (Boehm, 1981) on the productivity stage and some models, like Before You Leap (Gordon Group, 1986/1990) cover both stages. The two stages in SCE models are presented in Figure 1.

Function point analysis

Function point analysis (FPA) was developed by Albrecht (1979), and made widely available through user groups (such as Guide and Share). Albrecht was looking for a method to measure productivity in software development. For that purpose he developed FPA as an alternative to the number of lines of code. It is currently one of the most widely used models for software size estimation. The model has been refined since its origin (Rudolph, 1983; Jones, 1986; Symons, 1988). The principle of FPA is simple and is based on the number of functions the software has to fulfil. These functions are related to the types of data the software uses and generates. Within FPA the software is characterised by the following five functions:

1. the external input type,
2. the external output type,
3. the external inquiry type,
4. the logical internal file type,
5. the external interface file type.

For each of these five types the number of simple, average and complex occurrences that are expected in the software is estimated. By weighting each number with an appropriate weight and adding them, a number is obtained, the nominal number or the unadjusted number of function points. This indication of nominal size is then adjusted for the influence of the information processing complexity giving the adjusted number of function points. For example, a single function point in a system with a lot of data communication, highly complex processing and so on is harder to realise and costs more effort and time than a function point in a system without these demands. Instead of one function

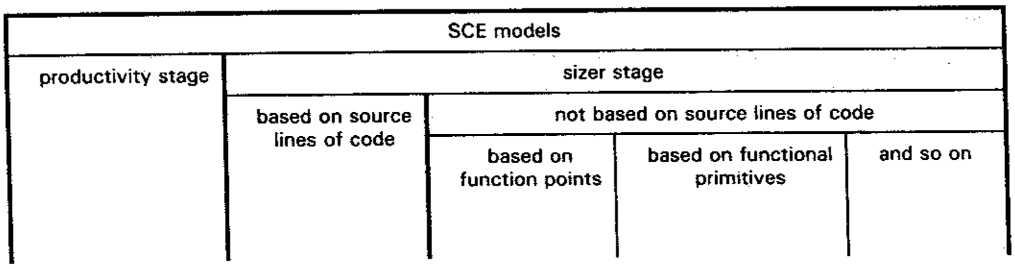
SCE models
productivity stage
sizer stage
based on source lines of code
not based on source lines of code
based on function points
based on functional primitives
and so on

Figure 1 The two stages of SCE models

point the characteristics of the software and the software environment demand an adjustment to, for example, 1.2 function points.

The final result of FPA is an estimate of the size of the software expressed in the number of adjusted function points. This number must be translated into an estimation of effort and duration. This translation however is not a part of FPA. An SCE model with a productivity stage is required to carry out this last step. The principle of FPA is presented in Figures 2 and 3.

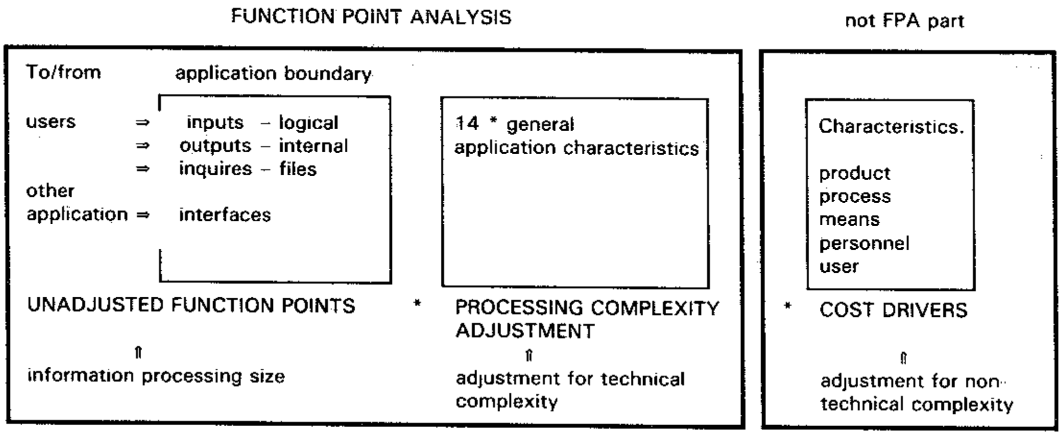
FUNCTION POINT ANALYSIS
not FPA part
To/from
application boundary
users ⇒ inputs – logical
⇒ outputs – internal
⇒ inquires – files
other application ⇒ interfaces
14 * general application characteristics
Characteristics.
product
process
means
personnel
user
UNADJUSTED FUNCTION POINTS
*
PROCESSING COMPLEXITY ADJUSTMENT
*
COST DRIVERS
information processing size
adjustment for technical complexity
adjustment for non-technical complexity

Figure 2 Global overview of FPA.

FUNCTION COUNT ← max range. factor *2 →

Type ID	Description	level of information processing function			total
		simple	average	complex	
IT	External input	_* 3=_	_* 4=_	_* 6=_	
OT	External output	_* 4=_	_* 5=_	_* 7=_	
FT	Logical internal file	_* 7=_	_* 10=_	_*15=_	
EI	External interface file	_* 5=_	_* 7=_	_*10=_	
QT	External inquiry	_* 3=_	_* 4=_	_* 6=_	
FC	Total unadjusted function points				

⇑ max range factor 2 5 ⇓

GENERAL INFORMATION PROCESSING CHARACTERISTICS

Characteristics	DI	Characteristics	DI
C1 Data communications	–	C8 On-line update	–
C2 Distributed functions	–	C9 Complex processing	–
C3 Performance	–	C10 Re-usability	–
C4 Heavily used configuration	–	C11 Installation ease	–
C5 Transaction rate	–	C12 Operational ease	–
C6 On-line data entry	–	C13 Multiple sites	–
C7 End-user efficiency	–	C14 Facilitate change	–
Total degree of influence			

DI values

Not present or no influence	= 0	Average influence	= 3
Insignificant influence	= 1	Significant influence	= 4
Moderate influence	= 2	Strong influence, throughout	= 5

FC (function count)	≡ total unadjusted function points
PC (process complexity)	≡ total degree of influence
PCA (process complexity adjustment)	= 0 65 + 0 01 * PC
FP (function point measure)	= FC * PCA

Figure 3 Overview of function point analysis.

Figure 2 gives a global overview while Figure 3 goes into more detail.

Function points are conceptually different from lines of code because they attempt to measure software function rather than software product size. For certain applications, particularly in information systems and data processing, function points are in fact used as a size measure. Many existing commercial SCE models have as sizing techniques FPA or FPA lookalikes. We mention here the models SPQR/Checkmark (Jones, 1986), Before You Leap (Gordon Group, 1986/1990), Asset-R (Reifer, 1987), Estimacs (Rubin, 1985), BIS (BIS/Estimator, 1987), Estimate/1 (Arthur Anderson, 1987). There are also function point extensions for real-time systems (IITR, 1987; Reifer, 1989). However, these are seldom used.

On the use of function point analysis

Estimation methods, models and FPA

To determine the actual use of FPA in practice, data were obtained in a survey (Siskens *et al*, 1989). This survey, carried out by the Eindhoven University of Technology, gives an overview of the present state of the art of the estimation and control of software development projects in Dutch organizations. The most interesting conclusions are:

- 35% of the participating organizations do not make an estimate of software development costs and time;
- 50% of the responding organizations record no data on a continuing project;
- 57% do not use cost-accounting;
- 80% of the projects executed by the participating organizations have overruns of budgets and duration;
- the mean overruns of budgets and duration are 50%;
- 62% of the organizations that make an estimate, base their estimations on intuition and experience; only 16% use formalised estimation methods like cost-estimation models.

In Table 2 an overview is given of the frequency with which current SCE methods are used.

Table 2 Use of software cost estimation methods.

Method	*Use (%)*	*Number*
Expert judgement	25.5	94
Analogy method	60 8	226
Capacity problem	20.8	77
Price-to-win	8 9	32
Parametric models	13.7	51

Note. an organization can use more than one method $N = 388$

The figures show that most organizations make use of information from past projects in some form or another. This mostly works on an informal basis, because only 50% of the participating organizations record data on completed projects. The information will be stored in the memories of the estimators. Estimates based on expert judgement and the capacity method are popular.

Looking more in detail at the use of SCE models, it can be seen that FPA is by far the most popular model. 51 organizations use SCE models and 45 of these use FPA (see Table 3). This result is not surprising because 82% of the responding organizations mainly develop business applications and FPA has been developed for estimating this kind of application.

Table 3 The use of SCE models. The total number of organizations that answered this question was 51. Several answers were allowed.

Model	*Number*	*%*
Function point analysis	45	52
Other models	36	42
COCOMO	4	5
ESTIMACS	4	5
Putnam/SLIM	3	4
PRICE-S	2	2
SPQR	0	0

The overview shows that the category of Other models is large. In analysing the answers it was not possible to find out if these models were in-house-developed or commercially available models. However, we have a strong impression that most of these models are in-house-developed, because many organizations gave an answer like 'an adapted version of FPA' or 'something like FPA'.

It is interesting to see if FPA is used in combination with other estimation methods. Table 4 gives this overview.

From the figures shown in Table 4 it can be seen that there is a difference in the frequency distribution of the use of estimation methods between FPA users and non-FPA users. Among FPA users the expert judgement method is less frequent and the analogy method is more frequently used. The relative use of (other) estimation models by FPA users is higher than by non-FPA users. These figures are not surprising. The analogy method can only be used on the basis of data from completed projects. This kind of data is a condition for a justified use of FPA. For that reason the step from the use of FPA to using the analogy method is a small one. This may be an explanation of the high percentage (85.7%). The low use of the expert judgement in combination with FPA can be explained by the fact that the judgement of the expert is already involved when using FPA. There are several explanations for the high percentage (57.1%) of respondents using FPA in combination with other models. A positive explanation would be that

Table 4 FPA in combination with other estimation methods ($N = 45$).

FPA used in combination with	*Frequency (%)*	*Another SCE model used in combination with*	*Frequency (%)*
Expert judgement	11.9	Expert judgement	25.5
Analogy method	85.7	Analogy method	60.8
Other models	57.1	Other models	20.8
Price to win	7.1	Price to win	8.9
Capacity problem	23.8	Capacity problem	13.7

organizations that use FPA are model-minded and the step to using other models is made easy. There are also certain benefits in checking one model against another and different stages in the life cycle may benefit from the use of different models. A more pessimistic outlook is that FPA users have little confidence in the results of estimates made by FPA, and for that reason are looking to other models as well.

Characterisation of FPA users

Analyzing the results of the survey in more detail, it can be seen that 53% of the FPA users are organizations with more than 500 employees and 56% of these users can be found in organizations with more than 20 employees in the EDP department. Furthermore, FPA in most cases (48%) is used to estimate small projects (less than 12 man-months - see Table 5).

Table 5 On the use of FPA

Organization size (number of employees)		*EDP department size (number of employees)*		*Size of project (man-months)*	
Number	*%*	*Number*	*%*	*Size*	*%*
20–49	2.2	< 2	2.2	< 12	48.1
50–99	11.1	2–9	31.1	12–48	25.9
100–199	15.6	10–19	11.1	49–200	14.8
200–499	17.8	> 20	55.6	> 200	11.1
> 500	53.3				

Almost all FPA users record data on past projects (96%). The figures for cost accounting (73%) are less good. Compared with the overall figures, presented in the next section, it is clear that the figures of the FPA users are far better (see Table 6).

Table 6 Recording and cost accounting by FPA users.

	FPA users (%)	*Overall (%)*
Recording	95.6	50
Cost accounting	73	43

FPA and overruns of budgets and duration

One of the main items in the field study were the overruns of budgets and duration experienced by the respondents. Table 7 gives an overview of the budget overruns of FPA users and non-FPA users.

These figures refer only to organizations that make estimates of development projects, record data on past projects and carry out cost accounting. From the 597 organizations that participated in the survey, only 160 satisfied these three demands. 32 of these 160 organizations were FPA users and 128 were non-FPA users. Remember that overall there were 45 FPA users; this means that only 72% record data on completed projects *and* do cost accounting. To give a more detailed impression of the overruns, the figures have been split up in overruns for the categories small projects

Table 7 Cost overruns of budgets for the FPA users and the non-FPA users.

Overruns	*Small projects*		*Medium-sized projects*		*Large projects*		*Very large projects*	
(%)	*FPA (%)*	*Other (%)*	*FPA (%)*	*Other (%)*	*FPA (%)*	*Other (%)*	*FPA (%)*	*Other (%)*
0	25.0	32.8	–	15.9	5.6	11.3	9.5	19.0
< 10	41.7	46.7	43.5	35.5	22.1	38.7	23.8	28.6
10–49	33.3	18.0	56.5	41.1	61.1	41.9	42.9	28.6
50–100	–	2.5	–	4.7	5.6	3.2	23.8	14.3
> 100	–	–	–	2.8	5.6	4.8	–	9.5
Total	100	100	100	100	100	100	100	100
Number of organizations	24	122	23	107	18	62	21	21

(< 12 man-months), medium-sized projects (12–48 man-months), large projects (49–200 man-months) and very large projects (> 200 man-months). The 128 non-FPA users are indicated by 'other' in the table. For each category the number of organizations is given. For example, there are 24 organizations that use FPA to estimate small-sized projects. 25% of these 24 organizations have no overruns of budgets, 41.7% have overruns of less than 10%, and so on. Overruns of more than 50% do not exist in this category. Analyzing these results, the next hypothesis has been tested: organizations that use FPA have fewer overruns than organizations that do not use FPA.

The Wilcoxon rank sum *W* test was used to test this hypothesis. After testing, this hypothesis had to be rejected. A detailed analysis showed a significant negative relationship between the use of FPA and overruns. This means that FPA users have higher overruns of budgets than do non-FPA users. The detailed analysis also showed that this relationship is only valid for large projects. A possible explanation for this disappointing result is given by de Haas (1988) and Rabbers (1988). They point out that the implementation and use of FPA is not easy. Generally it take years before FPA can be successfully implemented in an organization. According to de Haas and Rabbers, many FPA users do not pay sufficient attention to this implementation problem. As a result, FPA is often not used carefully.

FPA and the method of cost estimation

In the survey much attention was paid to the way cost estimation was done in an organization. Questions were formulated such as:

(1) How many times is an estimation made during a software development project?
(2) When during development is an estimation made for the first time?
(3) In what detail is an estimation made (split up into phases, activities)?
(4) Who makes the estimate?

The answers to questions (1) and (2) are given in Table 8, to question (3) in Table 9 and to question (4) in Table 10. As can be seen from Table 8, the majority of FPA users make an estimate only once during a development project. This is not surprising because FPA is used mainly for small projects, as mentioned before. Table 8 also shows that FPA is mainly used in the early stages of software development. This is not surprising either, because FPA claims to be applicable early in the life cycle.

Table 8 Frequency of estimation, and the point at which an estimate is made for the first time.

Frequency of estimation	*%*	*First point of estimate*	*%*
Once	45.6	After the first global description	59.0
Twice	22.0	After specification phase	30.1
3–5 times	10.4	After design phase	10.9
> 5 times	22.0		

The only remark about Table 9 is that FPA does not actively support a detailed estimate. We cannot obtain a distribution of effort for phases and activities by using FPA.

Table 9 Detail of estimate.

Level of detail	*%*
Complete project	28.1
Per phase of project	32.9
Per activity per phase	39.0

Table 10 shows that only a few organizations using FPA have a independent department or person responsible for cost estimation. We do not believe there should be such a department/person. It is not wise to split up the responsibility for making an estimate on the one hand, and for adhering to it when doing the project on the other hand. To guarantee a high level of commitment, the developers themselves have to be involved in the estimation process.

Table 10 Those involved in making an estimate.

Function	*Frequency (%)*
Controller/management	28.1
Development team	59.4
Project estimator	9.4
Project management	93.8
Customer	43.8
Other	9.4

FPA users and their problems

The FPA users were asked to indicate the main problems encountered in using FPA; the results are shown in Table 11.

Table 11 Problems in using FPA.

Problem	*Frequency (%)*
Subjectivity in determining input	28
Lack of uniformity of definitions	9
Problems with calibration of FPA	13
Problems with sizing	26
Insufficient insight in project to determine model parameters	20
Other	4

Two points are particularly interesting in these replies. The first is that FPA users need support in the definition and interpretation of terms used within FPA. Subjectivity unmistakably plays an important role in the use of FPA. Experienced users have solved this lack of objectivity by using rules of thumb in counting and weighting the function points. The second is the high percentage of FPA users who have problems with sizing the software. This is remarkable because FPA has been developed as a means to estimate software size and is normally considered to be a strength of the model.

FPA: usefulness of the sizer

In the previous section, some results concerning the use of FPA in Dutch organizations were presented. From this it can be seen that it is by far the most widely used model and thus warrants closer examination. It can also be seen that the results obtained by organizations using the model were, if anything, even worse then those obtained by other organizations. This shows that the use of a model by itself will not necessarily lead to improvements. In this section, we take a closer look at the foundation of FPA, namely its principle of expressing the size of a product in function points. For this, we will use the results of an experiment performed last year in a large multi-national corporation.

The object of this study was to obtain an evaluation of several software cost estimation models. In an experimental setting we asked fourteen experienced project managers to estimate effort and duration of a project. For this, data on a real project were used. The managers were asked to:

- make a first estimate,
- make an estimate using an automated version of a model,
- to do this again, using another model,
- to make a final estimate, using all information obtained up till then.

The results of this experiment are presented in Table 12. A more detailed description of this experiment can be found in Kusters *et al* (1990).

Before starting this experiment a selection of models was made. The intention was to use only the most promising models, since this was an expensive experiment and we wished to be as efficient as possible. Several selection criteria were used, of which one was the demand that KLOC (thousands of lines of code) was not to be used as a sizing measure. As mentioned before, some models use this measure for software size, but this is problematic. It is not easy to estimate the size of a product in lines of code in a relatively early phase in the development. Van Vliet (1987) draws a parallel to the conception of a novel. How can an author estimate the number of pages of a new book? FPA seems to be a better alternative for the sizing problem. Referring to the author of the book, he surely has some ideas about the number of characters, the location and the time and scope of the book. FPA is based on that same idea. Instead of estimating the number of lines of code, an indication of functional aspects of the software to be developed must be given.

In the study we assumed that lines of code could not possibly give a proper indication as to the size of the product. In order to test this assumption the project leaders were asked to estimate the number of lines of code needed for the system. At the same time, since both models used in the experiment support FPA, estimations of the number of functions were obtained. This gives the opportunity to compare both sizers. Direct comparison is impossible, since the measuring scales are not compatible, so we decided to use the coefficient of variation to see which sizer is the most consistent. The results were as follows:

Coefficient of variation	
• using lines of code	0.59
• using FPA	0.23

On the basis of this material it is clear that lines of code as an estimator for size in an early stage of systems development is outperformed by a readily available alternative, namely function points. This conclusion is confirmed by the fact that only seven out of the fourteen project managers thought themselves capable of making an estimate of size in lines of code. Also, during a discussion which concluded the experiment, there was a consensus on this point.

We think that this is a sufficient indication for the usefulness of function points as a means for estimating the size of a system. In the next section we will have a closer look at the technological complexity adjustment part of the FPA model.

Table 12 Summary of experimental results

	Effort (man-months)				
	First estimate	*Estimate using Model A*	*Estimate using Model B*	*Final estimate*	*Actual*
Average	28.4	27.7	48.5	27.7	8
Standard deviation	18.3	14.0	13.9	12.8	

Model A was Before You Leap (Gordon Group, 1986/1990), Model B was Estimacs (Computer Associates, 1986)

FPA: usefulness of the cost driver part of FPA

The cost driver part of FPA contains fourteen characteristics concerning the information processing complexity of the software. Each factor is capable of changing the nominal size estimation by 5%. The total effect of the adjustment will cause the final estimate to end up between 0.65 and 1.35 of the nominal size.

In order to test the usefulness of this part of the model we took a closer look at the use of FPA in one large organization. The organization is the national sales organization of a manufacturer of computer hardware. It also provides a range of software services for its clients. In the organization the use of FPA has been mandatory for several years. Management even went as far as making FPA a strategic product for the organization. The organization was one of the first to use FPA in the Netherlands, so providing us with the best possible source of information.

Within this organization it was seen that some project managers obtained better results with FPA than others. It was the feeling of management that these good results were caused by some people having a better understanding of the adjustment part of FPA. It was therefore the object of the study to capture the knowledge of one of the better FPA users in an expert system, thus providing this knowledge to all users.

We started by looking at nine major projects to see the way in which FPA had been used, paying specific attention to the cost driver part. To our surprise, we noted that for one-third of the projects the cost driver part of FPA had not been completed. For the other projects the average value of the correction factor was 1.01. This is suspiciously close to 1, i.e. no effect.

We then interviewed nine project managers, all FPA users, in order to find out what lay behind this. Their first reaction was that they considered the adjustment characteristics to be 'unimportant'. When following this up it appeared that they thought the definition of the characteristics was insufficiently accurate. Furthermore it was their opinion that a number of characteristics were outdated.

A series of in-depth interviews with the acknowledged FPA expert, a project manager with many years' FPA experience, confirmed this view. This expert did not use the FPA adjustment part either. Instead he calculated some kind of productivity factor. In the remainder of the study a small expert system was built, providing partial support in determining this factor (Bruns, 1989).

From this field study it may be deduced that the usefulness of the cost driver part of FPA is relatively low. We believe this is partly due to the way the effect of the cost drivers is passed on to the nominal estimate. If we look at the COCOMO factor 'complexity', which is defined in a way similar to the FPA factor 'complex processing', we see that the effect of the COCOMO factor by itself is just as large as the effect of all FPA factors combined. And it must be admitted that the COCOMO supposition is the more realistic one.

Another problem, which was already noted in the interviews, is that the FPA adjustment does not cover the most important technological complexity characteristics and that some characteristics are trivial; for instance, the characteristic 'online data entry'. It is difficult nowadays to find information systems where online data entry is not a regular feature.

Conclusions

In this paper we discussed the merits of the FPA model. FPA was chosen because it is one of the most widely used aids for SCE. In the discussion we used data from a large survey of Dutch organizations, an experiment on the effectiveness of software cost estimation models and a field study aimed at the adjustment part of the FPA model.

The survey confirmed that FPA is widely used, at least in the Netherlands. This in itself is a development that is to be recommended. The standard use of such a tool should provide organizations with information that can be used to learn from their previous experience in a more methodical way. However, using the tool by itself will not solve all the problems in this area. This can be illustrated by the relatively poor results obtained by organizations using the model. They scored worse than organizations that did not use the model.

However from the experiment it could be seen that FPA performed quite well as a sizing measure. It certainly outperformed lines of code as an estimator within the setting of our experiment. It remains to be shown how the two metrics compare in a setting where a more methodical approach towards estimating lines of code (e.g. Delphi) is taken. However these results confirm that FPA as a sizing measure is more acceptable for the estimators. This indicates that they will be more likely to use it. The common use of sizing and estimating methods should advance the state of the art within an organization. In such a situation FPA can play the role of a common 'language', allowing the various parties to communicate on a lower level of abstraction, namely the FPA inputs.

Finally, the field study showed that the adjustment part of the model is less than satisfactory. Experienced users showed no faith whatsoever in the adjustment characteristics. A subsequent analysis of the characteristics confirmed this view. However it could be questioned whether any model is capable of performing well in this respect. In our experience there are more arguments against than for the idea of a small set of generally applicable cost drivers. This shows that the use of a model will always have to be approached with some care. A model is not a machine where questions are entered at one side and correct answers appear at the other end.

References

ALBRECHT A J (1979) Measuring application development productivity In *Proc Joint SHARE/GUIDE/IBM Application Development Symposium*, Monterey, October 1979.

ALBRECHT A J and GAFFNEY J E (1983) Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Transactions on Software Engineering* **SE9(6).**

ARTHUR ANDERSON (1987) *Estimate/1, documentation Method/1: automated project estimating aid*

BIS/ESTIMATOR (1987) *User Manual version 5 0.* BIS Applied Systems

BOEHM B W (1981) *Software Engineering Economics* Prentice-Hall, Englewood Cliffs, New Jersey

BRUNS B (1989) Begroten van automatiseringsprojecten Master Thesis, Faculty of Informatics, University of Technology, Eindhoven

COMPUTER ASSOCIATES (1986) *CA-Estimacs User Guide, Release 5 0.*

DEMARCO T (1982) *Controlling Software Projects* Yourdon, New York

DEMARCO T (1984) An algorithm for sizing software products *Performance Evaluation Review* **12**, 13-22.

GENUCHTEN VAN M and FIERST VAN WIJNANDSBERGEN M (1989) An empirical study on the control of software development In *Proceedings of the International Conference on Organisation and Information Systems*, Bled, September 1989, pp 705-718

GORDON GROUP (1986/1990) *Before You Leap, User's Guide* (1986) *Before You Leap Mk. II, User's Guide* (1990)

HAAS B M G DE (1988) Function point analysis in short (in Dutch). In *Proceedings of the Conference on FPA in Movement: the Practice of FPA in the Netherlands and the USA*, NGI-SIC, Scheveningen, 21-22 November 1988, pp 13-22

HEEMSTRA F J (1989) *How Expensive is Software? Estimation and Control of Software Development* (in Dutch) Kluwer, Deventer, The Netherlands

IITR (1987) *A Descriptive Evaluation of Software Sizing Models* Prepared for headquarters USAF/Airforce Cost Center, Washington, DC 20330-5018, by IITR, 4550 Forbes Boulevard, Suite 300, Landham, MD 20706-4324

JENKINS A M, NAUMANN J D and WETHERBE J C (1984) Empirical investigations of systems development practices and results *Information & Management* **7**

JONES C (1986) *Programming Productivity* McGraw-Hill, New York

KUSTERS R, HEEMSTRA F J and VAN GENUCHTEN M (1990) Are software cost estimation models accurate? *Information and Software Technology* **32(4)**, 187-190

LIEROP VAN F L G, VOLKERS R S A, VAN GENUCHTEN M and HEEMSTRA F J (1991) Controlling software projects: is it better to have one project in control than ten in option? (in Dutch) *Informatie* **33(3)**, 198-200

PHAN D, VOGEL D and NUNAMAKER J (1988) The search for perfect project management *Computerworld* **September**

RABBERS R K (1988) Introducing FPA at PTT-Telecom (in Dutch) In *Proceedings of the Conference on FPA in Movement: the Practice of FPA in the Netherlands and the USA*, NGI-SIC, Scheveningen, 21-22 November 1988, pp 319-331

REIFER D J (1987) Softcost presentation to the third COCOMO Users' Group Meeting, Pittsburgh, Pennsylvania, November

REIFER D J (1989) *Asset-R Manual* Reifer Consultants, 25550 Hawthorne Boulevard, Suite 208, Torrance, California

RUBIN H A (1985) A comparison of cost estimation tools In *Proceedings of the 8th International Conference on Software Engineering*, IEEE,

RUDOLPH E E (1983) *Function Point Analyses, Cookbook* Private publication, 3/1983

SISKENS W J A M, HEEMSTRA F J and STELT H VAN DER (1989) Cost control of software projects; an empirical investigation (in Dutch) *Informatie* **31(1)**, 34-43

SYMONS R C (1988) Function points analysis. difficulties and improvements *IEEE Transactions on Software Engineering* **January**

THAMBAIN H J and WILEMON D L (1986) Criteria for controlling projects according to plan *Project Management Journal* **June**

THETA ANALYSIS & SYSTEMS LTD (1990) *SEER Manual.*

VLIET J C VAN (1987) *Software Engineering* Stenfert Kroese,