A hybrid effort estimation model for perfective maintenance: a real experience

Emanuel Irrazábal[1,2], Zurisadai Benjamin Osorio[3], Javier Garzás[1,2]

[1] Kybele Consulting S.L., Madrid, Spain
{emanuel.irrazabal, javier.garzas}@urjc.es
[2] University Rey Juan Carlos - Kybele Research, Madrid, Spain
{emanuel.irrazabal, javier.garzas, esperanza.marcos}@urjc.es
[3] Technological Institute Orizaba, Veracruz, México
elpkbj@gmail.com

Abstract. *This paper presents the results of a hybrid estimation model for perfective software maintenance applied in a real context. The resulting model integrates heuristics and parametric methods to improve the maintenance estimation. One of the main objectives of this model is the inclusion of functionalities that provide zero function points. These functionalities can be supported by the model through impact points. Likewise, the micro function points and weighted risks have been included to provide a more realistic estimate. Finally, the model was implemented in a SME where the estimates were very approximates to the real effort. As result, the model could be implemented in a small organization with a stable maintenance process and effort deviation was 11% after applying the model.*

Keywords: estimation model, perfective maintenance, effort, impact points, micro function points, methods.

1 Introduction

In smalls organizations there is debate around the fact that size estimation is an important aspect of managing and controlling project delivery [1]. However, the organization must establish a mechanism to estimate the size of the software project to assess the level of effort needed to build the required deliverable [1]. The importance of having an accurate (or accurate enough) sizing mechanism has a direct impact on the efficiency of resource use and management of budgets [2]. Although the tasks related to estimate are hard, they are necessary. Therefore there are techniques to estimate in early phases of software lifecycle [3].

On the other hand, perfective maintenance is the most used in relation to other types of maintenance in the software industry according to [4], because the most of the software industries incorporate new features to their applications, either by customer request or to generate a new version of their product for marketing. When the estimates are made to calculate the effort in maintenance process, the result of obtaining such estimates is called maintenance cost prediction [5] or maintenance project effort estimation [6], where the software maintenance is defined as “a process of modifying a software system or component after delivery to correct faults,

improve, performance other attributes, or adapt to a changed environment" [7], in other words, the maintenance is the process associated to the change.

A maintenance project is the most difficult to estimate [8] and there is a challenge related to estimate a maintenance project under dynamically circumstances (risks) [9]. A number of estimation models [6, 11-16] for software development have been broadly adopted such as size measures as source lines of code, function points, etc. The models do not describe the estimation techniques used and their application in real context oriented to small organizations. Results about the impact on estimates of very small functionalities that are difficult to estimate were not provided because these models are not focus in these functionalities. Also the models have not enough risk factors related to software maintenance that may impact the estimated effort.

The purpose of this paper is to present a hybrid estimation model based on heuristic and parametric methods. Also, the model applies micro function points [17] to estimate the make more realistic estimates and impact points [18] to measure the functions with zero function points (both are explained later in this paper). An experience and of using the model is described in a real context. The model was implemented in a SME (Small and Medium Enterprise) from Mexico, where results were very satisfactory.

The remainder of the paper is organized as follows. Section 2 addresses the related works to this research and a more detail of objective of the estimation model proposed. Section 3 mentions the main objective of the estimation model proposed and needs to satisfy by this proposal. Section 4 describes the method based on IPs count, their objective, scope, considerations, and integration with the estimation model. Section 5 details the micro function points and Section 6 mentioned the risk referred to integrate the model. The structure of the estimation model proposed is described in Section 7. The experience and results are mentioned in Section 8 and finally Section 9 presents the conclusions, the model's scopes and suggestions for possible future work.

2 Related Works

There are few models and researches that address the maintenance estimation based on software size estimation techniques and estimation methods, in this section are mentioned some. The model proposed by [11] was tested in a controlled environment and it used the LOC as main factor to generate estimates. Model raised by [12] and models analyzed by [13] apply only LOC in the estimates generation. These three works observe that there is a dependency to the programming language when this measure is used to estimate. Some difficulties were derived from the variation in the definition on source codes itself and counting the new, changed, or reused lines delivered. These researchers conclude that it is necessary a good relationship between the maintenance team members. Thus, the estimates are provided and the risks are mitigated with more accuracy. These models produced an effort variation of 10% compared to estimates.

Other researches [14, 15] founded that expert estimates outperform the function point estimates, while analogy estimates slightly outperform the expert estimates.

Some studies [6, 15, 16] have addressed the most used parametric method: the function points. However, the authors have carried out changes in the original method to estimate a maintenance projects. The main conclusions and limitations encountered are:

- The size of the resulting system is not a simple sum of existing size and the size of the new component. The resulting system can even be smaller than initial one (when a part of functionality is being removed).
- Mainly, models have not specific adjustment factors for maintenance.
- The size in a component to be changed has a much larger impact on effort than the size of the changes.
- A situation where the functional size is zero frequently occurs and expert opinion should be used.
- The estimates with wide tolerance range during early phases of software lifecycle affect mainly to small development teams. These small organizations can not waste too much effort because this strongly impacts their benefits.

3 The Aim of the Proposed Estimation Model

According to the recommendations of [10] and based on previously identified needs, it is necessary a model that can be applied in broader categories of software applications and a certain level of external validation to apply different estimation techniques. Hybrid model proposed will have the next characteristics:

- Acceptable tolerance range and effort estimation determined by micro function points expert opinion and researches, statistical techniques.
- Quantitative representation in the functionality changes that provide zero function points. This will be done by the impact points.
- Specific adjustment factors risks involved in software maintenance.
- The model will work with analogy, algorithmic and statistical methods and will incorporate expert opinion to balance the risks.

The proposed estimation model has been tested in a real context, to increase and improve the validation. The recommendation given in [19] has been considered in this research.

4 A start toward the solution: Impact Points Count

The FPs (*Function Points*) has problems measuring other functions that may be impacted by a specific change (zero FPs). For example, changes to static Web pages or populating code tables that are not related or used by the FP. Junhai [20] addresses this problem, but he did not reflect it in a counting unit. He only identified accounting small functions with LOC. Other studies indicate that zero FP works involved 15 or 20 percent in maintenance projects [18].

Impact Points are used to measure zero FP functionality impacted in a project. Based in the IPs description, this measure provides the next advantages:

- It is a single measure for all impacted functionality. It is an addition to the measurement program in the case study.
- It is independent of the technology context and implementation techniques.
- It is based on the IFPUG FP methodology.
- It is not requires an extensive set of guidelines to be developed, or extensive training for employees already familiar with FP counting.

For example, impact points can count validations and confirm boxes, which provide zero FPs.

To implement IPs in an organization and in an estimation model, it is important to assess and define the projects types and situations that will use them (in this case estimation related to perfective maintenance).

Some steps to consider are:

- Develop a list of non-FP countable situations and projects then categorize them by type and volume (e.g., rate changes, new products, etc.).
- Conduct IP counts on a representative sample of projects from the non-FP countable project list.
- Record the effort and delivery platform for non-FP countable projects
- Assess the productivity rates (IPs per hour) to determine trends and any further breakdowns/measures needed.
- Develop templates to use going forward to avoid conducting IP counts on all non-FP countable projects.

If the rate changes typically impact the same functions, then the same IP count would be used each time.

4.4 Integration with the Estimation Model

Based on the recommendations described before, the estimation model required the definition of the functions with zero PF, which are shown in Table 1. The functions should not maintain files and do not calculate, derive or maintain data. It is possible to add many more features to the table for more granularity level.

Table 1. Example of *Functions* considered to be counted with Impact Points

Classification	Functionality
Table updates	Add rate
	Delete rate
	Create table
	Delete table
	Update table
Code/Text Changes	Add help
	Update help

5. Improving the Estimation: Micro Function Points

The micro function points [17] were developed based on the reasoning that an EI (External Input) having 1 DET (Data Element Type) and 0 FTRs (Referenced Files) get scored as 3 (unadjusted) function points, while an EI with 15 DETs and 1 FTR also is scored as 3 function points. The resulting equations were obtained from multiple regression from the combinations of DETs, FTRs and the corresponding function point counts. The study description, analysis and the next equations were obtained from [17].

This FP change makes the estimated effort more adjusted and real in the application to maintain. Also these estimates help to the least waste in time and resources to maintain small features; this is vital, especially in organizations and small teams. It does not recommend using the micro function point count approach for counting applications that have at least 30 different functions. The next section mentions risks to use in the estimation model to complement the software size estimation technique addressing the possible risks in a maintenance project.

6. Complementing the Model: Maintenance Risks

There are many researches that address the importance of the risk in a project. Sommerville [22] asserted that productivity factors concerning the environment of software maintenance can include characteristics such as module independence, programming language, programming style etc.

According to [11] the maintenance effort is affected by a large number of factors such as size and types of maintenance work, personnel capabilities, the level of programmer's familiarity with the system being maintained, processes and standards in use, complexity, technologies, the quality of existing source code and its supporting documentation.

Belady related the maintenance efforts with productivity effort, complexity of software design and documentation, and the degree of familiarity with the software [23]. Jorgensen's effort prediction model includes variables such as the cause of the task, the degree of change in the code, the type of operation on the code (mode), and the confidence of the maintainer [24]. However, models that have been studied [6, 14, 25] do not include the quantitative evidence generation derived from factors that may affect the process or maintenance project. A perfective maintenance project in regular conditions has an effort variation of 20% over the initial estimation [3]. In case more tolerance range is needed that assumed by the model for project delivery, it is recommended a meeting with the stakeholders to reach agreement. The percentages related to effort variation may be smaller if the organization improves the accuracy of the risk weighting and their impact in the maintenance process.

Then, for an estimated total percentage, the average risk is calculated. The result is the tolerance range for the delivery of the maintenance projects. It is recommended that the tolerance range does not report to the maintenance team, to have performance estimation under ideal conditions. The tolerance range is only considered for negotiation with the client.

7. Estimation Model Structure

In this section will be described the hybrid estimation model. First part of the model contains a specific template to use the IP count and the description of features this count uses. Template shows categories and their features, each feature has a field to introduce the number of IP in that classification. This section of the model uses the ascending method because the estimates have a very high granular level. If there is an accounting function with IP and it is not in the categories, it is necessary to add it and mention it to the maintenance team.

The second and third parts have a template to apply the micro function point count (parametric method) based on the fields of a traditional FP count. However, this templates use the micro function point formulas to calculate the total FP. These parts are applied to insert or delete functionality.

The last 2 parts are the most important. The principal section has the historical data from the projects, inputs and outputs, and all the risks. The fields of historical data projects are: Project's name, technology, FP to insert, FP to delete, FP to change, IP to insert, IP to delete, IP to change, (that 6 fields have their related efforts), effort estimate, effort estimate with risks, effort variation, percentage risk and real effort. The effort variation is the percentage representation of the subtraction of effort estimate minus effort estimate with risk.

This area allows applying the analogy method. The analogy method is used to filter the fields from projects with similar characteristics to the maintenance project to develop. The main characteristic to filter the projects is the size based on FPs or IPs. Filtration may contain some fields (e.g. IP to insert) and not all necessarily. When the projects are already filtered the next part of the model allows introducing the values of the new maintenance project according to the count performed to its features.

It should be emphasized that if there are not projects with similar characteristics it is necessary to calculate the average score of this characteristic, through of the sum of all the values of this characteristic (e.g IP to delete) divided by the total effort in all the projects with this characteristic. After that, the value can be entered in monte carlo section in field named "average". Figure 2 shows some fields of this part of the model from some projects of the SME that applied the estimation model.

Project's name	Technology	FP to insert	Effort FPI	FP to delete	effort FPD	FP to change	effort FPC	IP to insert	Effort IPI
P1	Java	16.77	140	9	22.5	0	0	10	6.3
P2	Java	16.15	138	0	0	0	0	6	2.6

Fig. 2. Some fields of historical data projects. *Project's name, technology, FP to insert, Effort in FP to insert, FP to delete, Effort in FP to delete, FP to change, Effort in FP to change,IP to insert, Effort in IP to insert.*

The final part of the model has a statistic method based on monte carlo simulation, which is very applicable and reliable for estimation [25-27]. The simulation takes the average score obtained from the risks and the standard deviation for its iterations.

Figure 5 shows monte-carlo simulation in the estimation model while some tests were performed.

	Total effort	5000.00	Efforft Variation Estimate	12%	Duration	208.30
IP to insert						
				Standard deviation	460.78	
Average	2302			Effort Estimate	2368.62	
Standard deviation	684			Minimum expected	2297.86	
				Maximum expected	2439.38	
	Iterations	150				
	Iteration	Effort				
	1	1915.69362				
	2	3128.96982				
	3	2416.95779				
	4	2204.74885				
	5	2176.73405				

Fig. 5. Some fields about the implementation of monte carlo simulation in the estimation model. IPs to insert*: Average, Standard deviation, total iterations, iteration and it effort, effort estimate, minimum expected and maximum expected. Total effort estimate, effort variation estimate and duration*, these 3 fields are from the total maintenance project.

8. Experience and Results

The estimation model was used in a micro SME from Veracruz, Mexico with one-half year experience in software development for electronic billing. The organization was composed by 22 persons between administrative staff and development team.

The projects were 57% related to maintenance in their applications and other provided by customers and their market was only regional. They had 7 projects, 4 of them was projects related to perfective maintenance, 3 of which were their developments and 1 was provided by the client. They used a process based on Agile Mantema [19]. They wanted to have quantitative evidence in the duration of their projects based on effort to be more successful in their offers to the customer.

Information was introduced to the model from 4 projects where the effort was recorded by PSP (*Personal Software Process*). The projects could be counted (after their termination and based on requirements) with IPs and estimated with micro function points. All the features with their times were recorded with strict accuracy, this provided to the model reliable records of maintenance team's ability. Time in each feature corresponds to total effort in planning, analysis, design, development, testing and deployment phases.

The risks relating to these projects could not be accurately quantified, so was used the experience of some maintenance team members associated with the previous projects to calculate the average score related to the risks. The projects and some of their characteristics are in the table 3 (the fields have been abbreviated).

Table 3 Description of some data from the historical projects. *Name's project, technology, FP to insert, Effort in FP to insert, FP to delete, Effort in FP to delete, FP to change, Effort in FP to change,IP to insert, Effort in IP to insert, effort variation and percentage risk.*

Name	Tech	FPI	EFPI	FPD	EFPD	FPC	EFPC	IPI	EIPI	EV	%R
P1	Java	16.77	140	9	22.5	0	0	10	6.3	7%	21%
P2	Java	16.15	138	0	0	0	0	6	2.6	5%	21%
P3	Java	16.7	145	11.8	26.2	0	0	8	4.1	8%	11%
P4	Java	20.2	230	0	0	6.1	48.4	4	2.2	2%	11%

A perfective maintenance project was estimated, it had similar characteristics to projects P1and P2. This project did not estimate the effort in each phase of software lifecycle because an analysis about historical distribution of effort in each phase did not exist. The distribution of effort was made based on expert opinion of the leader maintenance team.

Results are quite satisfactory because they have an 11% in average effort variation in planning, analysis, design, development, testing and deployment phases. The team cohesion, meetings and experience were the most important factors to minimize effort variation and solve problems faster. Figure 6 shows the effort variation in the project.

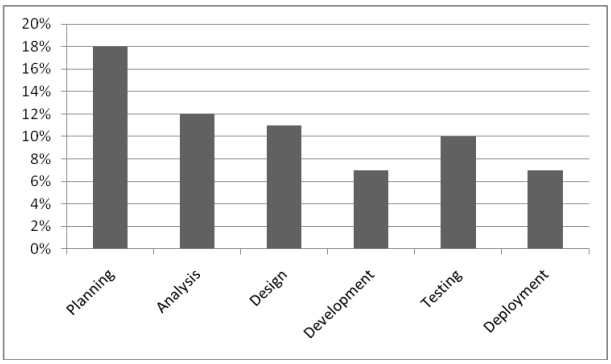

Fig. 6. Effort variation in *planning, analysis, design, development, testing and deployment phases.*

9. Conclusions

This paper present a hybrid effort estimation model for perfective maintenance tested in a SME and some results of effort estimates. The model applies the function points, impact points count and an estimation method based on micro function points.

The model integrates reasonably the expert opinion method to mitigate the impact in the risks estimation during the maintenance project. Also it carries out the analogy method to select similar projects and implement statistics formulas (algorithmic method) to provide reliable estimates.

The hybrid model provided the integration of some estimation methods to estimate effort according to the capabilities and experience in the organization. The Micro Function points, quantitative evidence of factors risk, and the statistical method provided more real effort estimates in the maintenance projects.

The IPs provides a high granularity level to measure functionalities with zero FP and they can adapt to the understanding of the maintenance team. Also, the IPs may vary to incorporate a complexity related to this IP type. New functionalities with zero FP are ease to incorporate to the model. So, it is possible to estimate these functionalities in the future. Size of maintenance software project can be estimated if the effort of the functionalities to insert, change or delete are managed separately. No matter if it is large or small.

It is possible to implement the model in small organizations with a well defined maintenance process and to provide good estimations and a minimum effort variation in a SME. The data reliability provided by statistical estimation depends of stability in the maintenance process implemented in the organization (no matter if it is based on agile or traditional methodology).

As future work, it expects to implement the model in different context (organizational and technologic) and test it in bigger maintenance projects to provide deeper validation. Likewise, it is necessary more detailed in the definition related to the factors risks for better understanding of them and a formal technique to weight these factors risks.

Acknowledgment. This work is part of e MODEL-CAOS project (TIN2008-03582/TIN) financed by the Spanish Ministry of Education and Science (TIN2005-00010/). Special thanks to CONACYT in México for to finance the research visit to Rey Juan Carlos University in Madrid, Spain.

References

1. Furulund, K. M., Molokken-Ostvold, K.: Increasing software effort estimation accuracy using experience data, estimation models and checklists. Presented at Quality Software, 2007. QSIC'07. Seventh International Conference on Quality Software (2007).
2. McConnell, S.: Software Estimation: Demystifying the Black Art. Microsoft Press (2006).
3. Piattini, M, García, F., Garzás, J., Genero, M.: Medición y Estimación Del Software: Técnicas y Métodos Para Mejorar La Calidad y Productividad Del Software, pp. 121-127. Ra-Ma (ed.) (2008).
4. Frazer, A.: Reverse engineering- hype, hope or here? Software Reuse and Reverse Engineering in Practice, pp. 209-243. PAV Hall (ed.) (1992).
5. Granja, J. C., Barranco M. J.: A method for estimating maintenance cost in a software project: A case study. Journal of Software Maintenance: Research and Practice 9(3), pp. 161-175 (1997).
6. Ahn, Y., Suh, J., Kim S., Kim H.: The software maintenance project effort estimation model based on function points. Journal of Software Maintenance and Evolution: Research and Practice 15(2), pp. 71-85 (2003).
7. IEEE.: IEEE standard 610.12–1990 IEEE standard glossary of software engineering terminology (1990).

8. Chemuturi. M.: Software Estimation Best Practices, Tools & Techniques: A Complete Guide for Software Project Estimators, pp. 41. J. Ross Publishing (ed.) (2009).
9. Shukla R., Misra A.: Ai based framework for dynamic modeling of software maintenance effort estimation. Presented at Proceedings of the 2009 International Conference on Computer and Automation Engineering (2009).
10. Riaz, M., Mendes, E., Tempero, E.: A systematic review of software maintainability prediction and metrics. Presented at Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (2009).
11. Nguyen, v., Boehm, B., Danphitsanuphan, P.: A controlled experiment in assessing and estimating software maintenance tasks. Information and Software Technology (2010).
12. Tan, H. B. K., Zhao, Y., Zhang, H.: Conceptual data model-based software size estimation for information systems. ACM Transactions on Software Engineering and Methodology (TOSEM) 19(2), pp. 1-37 (2009).
13. Jodpimai, P., Sophatsathit, P., Lursinsap, C.: Analysis of effort estimation based on software project models. Presented at Communications and Information Technology, 2009. ISCIT 2009. 9th International Symposium on Communications and Information technologies (2009).
14. Niessink F., Van Vliet, H.: Predicting maintenance effort with function points. Presented at International Conference on Software Maintenance 1997 (1997).
15. Kitchenham, B.: Counterpoint: The problem with function points. IEEE Software 14(2), pp. 29 (1997).
16. Antoniol, G., Fiutem, R., Lokan, C.: Object Oriented Function Points: An Empirical Validation(2003)
17. Tichenor Ch.: A way to count micro function point. Metric Views, vol. 4, pp. 13-15 (2009).
18. Holmer R.H.: Measuring Maintenance Activities within Development Project," Metric Views, vol. 3, pp. 12-15 (2009).
19. Pino F.: Integrated Framework for Software Process Improvement in small Organizations, pp 77-115 (2010).
20. Junhai,M., Lingling, M.: Comparison study on methods of software cost estimation. Presented at E-Business and Information System Security (EBISS), 2010 2nd International Conference on e- Business and Information System Security (2010).
21. IFPUG Counting Practices Committee.: Function Point Counting Practices Manual. Release 4.2.1 (2005).
22. Sommerville, I., Ransom, J.: An empirical study of industrial requirements engineering process assessment and improvement. ACM Transactions on Software Engineering and Methodology 14(1), pp. 85-117 (2005).
23. Belady, L., Lehman, M.: An introduction to growth dynamics in statistical computer performance evaluation (1972).
24. Jorgensen, M.: Experience with the accuracy of software maintenance task effort prediction models. Software Engineering, IEEE Transactions on Software Engineering, pp. 674-681 (2002).
25. Diev S.: Software estimation in the maintenance context. ACM SIGSOFT Software Engineering Notes 31(2), pp. 1-8. (2006)
26. Burch, R., Najm, F. N., Yang, P., Trick, T-N.: A monte carlo approach for power estimation. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on Very Large Scale Integration Systems, pp. 63-71 (2002).
27. Liu, J. S.: Monte Carlo Strategies in Scientific Computing pp. 1, 28-48. Springer (ed.) (2008).