Task Effort Fuzzy Estimator for Software Development

RODRIGO PALUCCI PANTONI[1]
EDUARDO ANDRÉ MOSSIN[2]
DENNIS BRANDÃO[3]

EESC/USP - Sao Carlos Engineering School, University of Sao Paulo
Electric Engineering Department
Avenida Trabalhador São-Carlense, 400
13566-590 - São Carlos (SP) - Brazil
[1]rodrigoppantoni@yahoo.com.br
[2]emossin@yahoo.com.br
[3]dennis@sel.eesc.usp.br

Abstract. Software estimation has been one of the biggest challenges in Computer Science for the last decades. This practice is essential for research and development companies, since it can provide cost and deadline forecasting, for example. Most of the traditional techniques such as function points, regression models, COCOMO, etc, require a long-term estimation process, which is unworkable for greatly dynamic companies who demand short-term task estimation, that is, in days. Furthermore, such techniques need historical data from past similar tasks, which may not exist when estimating an original task, a common situation for companies developing high technology. A well-known technique for short-term task effort estimation is the Use Case Points method, which only needs Use Case documents to start the estimation process. Based on this context, this work aims to apply the Fuzzy systems theory to estimate the effort for original software tasks in short-term, in a research and development company, based on the Use Case Points technique. Information from a real software project is provided for this methodology and used to validate the results. Results are validated by comparing the tasks planned with the Fuzzy estimator and the Use Case Points technique, and the expended execution time.

Keywords: Effort estimator, Fuzzy systems, Use Case Points, software development, software engineering.

(Received December 04, 2007 / Accepted March 26, 2008)

1 Introduction

Software estimation has been one of the biggest challenges in computer science for the last decades [3], due to its uncertain nature. Estimation is essential for research and development companies, since it can provide cost control, delivery accuracy, among many other benefits.

Based on this context, new paradigms such as Fuzzy logic may be an alternative to estimate task effort in software development.

This work intends to apply the Fuzzy Systems Theory to estimate the effort to develop original software tasks, also estimating the effort time to develop tasks as soon as possible in the software lifecycle, which would improve the original technique created using Use Case Points.

Data information used in this work is provided from a real software project developed by Smar Equipamentos Industriais [15]. Smar develops original fieldbus automation software systems and has more than fifteen patents in the USA and Europe.

Results are validated by comparing original tasks planned in a real project using the Use Case Points technique and Fuzzy, and the expended execution time.

2 Problem Definition

Developers should be able to estimate the time to develop software tasks by comparing similar tasks that have already been developed. However, estimating tasks has an uncertain nature, since it depends on several and usually not evident factors, and it is hard to be modeled mathematically [11] [13]. Researchers have proposed several methodologies or models based on Function Points [2] [8], Use Case Points [16], COCOMO (COnstructive COst MOdel) created by [1], among others. These models are also applied to other techniques (models), for example, in ordinary multiple regression, although there are techniques also using regression models only (statistic modeling technique).

Some researches related to software estimation using Fuzzy have been conducted in the past years in an attempt to improve the traditional techniques, as proposed by [4] and [17], who use the function points estimation technique together with Fuzzy logic; other researches use Fuzzy with the COCOMO model, as proposed by [6] and [7]; there is also the work proposed by [5], which suggested a technique using Fuzzy, the COCOMO model and neural networks; and, finally, the work of [10] using multiple regression together with metrics such as the *McCabe complexity* to classify complexity, *Dhama coupling* to classify coupling and lines of code to classify the size.

The traditional models mentioned above are based on the experience with similar tasks, such as information related to lines of code, coupling between modules, etc. The nature of the details and the large amount of information characterize a long estimation process using traditional models (in weeks/months). Different from the techniques mentioned previously, the Use Case Points technique is used worldwide because it is characterized by its advance estimation right at the beginning of the software lifecycle, that is, the time to estimate tasks is short (in days).

As noted, traditional models use project data from similar tasks executed in the past, but due to this fact it is not possible to use traditional techniques, even if combined with Fuzzy or neural networks, when the goal is to estimate original tasks that have no data for comparison. The Use Case Points technique is the exception, because it is only necessary to provide the specification of requirements and Use Case documents. Therefore, the Use Case Points estimation technique aggregates two important characteristics considering the work involved in this article: short-term estimation (1) and task estimation based on few references from past experiences (2).

This work focuses on estimating the time effort to implement a specific characteristic in a software system by combining part of the Use Case Points technique with the Fuzzy theory. It is important to emphasize that the initialization phase (based on RUP - Rational Unified Process) [14] and a small part of the software lifecycle documentation (Use Case documents) should have been concluded in order to qualify some of the information from the Fuzzy system (the inputs) and therefore obtain a better accuracy.

In this work, we consider that "effort" is the necessary amount of days used to execute a task. Effort is not the duration because duration is the total time including non-working days such as public holidays and Sundays, the time spent in the tasks not related to the project, etc. The effort considers only working days, without intervals, and executing only this very task.

3 Problem Solution

The variables generated from software metrics are used in software project management models. For similar tasks, and this case is not considered in this article, other inputs are typically used to estimate future tasks, such as lines of code information, coupling among modules, etc, and used in multiple regression estimation, for example. Usually, researchers try to estimate with maximum accuracy, that is, with a minimum error (minutes or few hours, at most), and the time needed for the estimation does not matter.

However, software tasks estimation does not seem to have the accuracy desired by researchers. For this reason, this work focuses on other attributes used as inputs for the estimation system, aiming at a short-term estimation time and providing significant results. Such attributes are easily and quickly obtained for the estimation, which means that it is not necessary to minutely count and analyze each line of code to estimate a task that may just require a little more development effort time than the effort time required to estimate this task.

To apply the estimation methodology using Fuzzy, this work includes a case study related to a project from Smar Equipamentos Industriais named Syscon, a fieldbus network configuration software for industrial automation systems. The inputs, the Fuzzy rules and the pertinence functions were defined according to the development process for the software mentioned above.

The Fuzzy estimator proposed in this article has three inputs and one output. The inputs are the complexity to modify the modules, the experience of developers in the C++ programming language and in the project, and the size of the alteration. The output from the inference system is the effort time estimated in days (Figure 1).

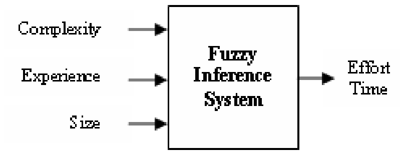

Figure 1: Fuzzy Inference System

To compose the Fuzzy effort estimator, the so-called "fuzzification" transforms the linguistic variables and their terms or sets. Figures 2 to 4 show the graphs from the pertinence functions for the Fuzzy inputs. Figure 5 shows the pertinence function for the system output.

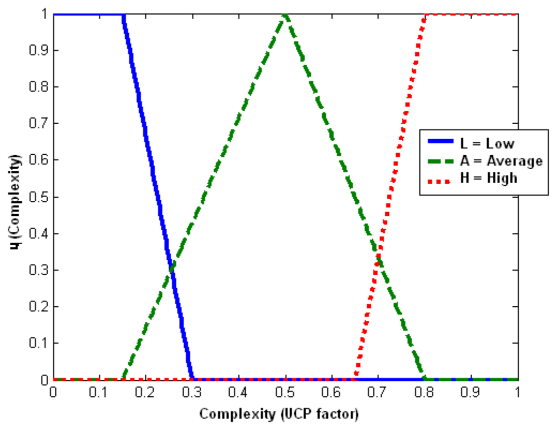

Figure 2: Complexity to modify the modules

The terms Low (L), Average (A) and High (H) were defined for the variable "Complexity" in Figure 2.

The terms Low (L), Average (A) and High (H) were defined for the variable "Experience" in Figure 3.

The terms Small (S), Medium (M) and Large (L) were defined for the variable "Size" in Figure 4. Finally, the terms Very Low (W), Low (L), Average (A), High (H) and Very High (G) were defined for the output of the Fuzzy system in Figure 5.

To calculate the complexity entered in the Fuzzy system, the complexity factor (*technical complexity factor*) from the Use Case Points method [16] was used, which includes functional and non-functional requirements (macro requirement) for the alteration. This work will not describe the factor calculation in details due to the large amount of steps needed to obtain this factor. For further details, consult [16], [9] and [12].

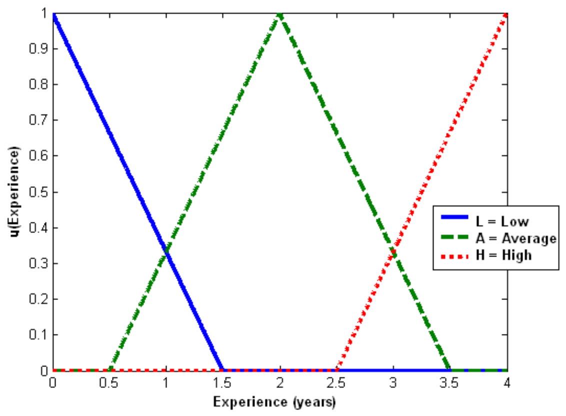

Figure 3: Experience of Developer

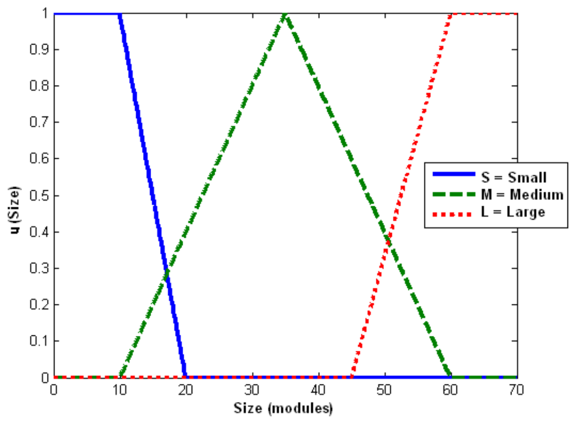

Figure 4: Size of the alteration

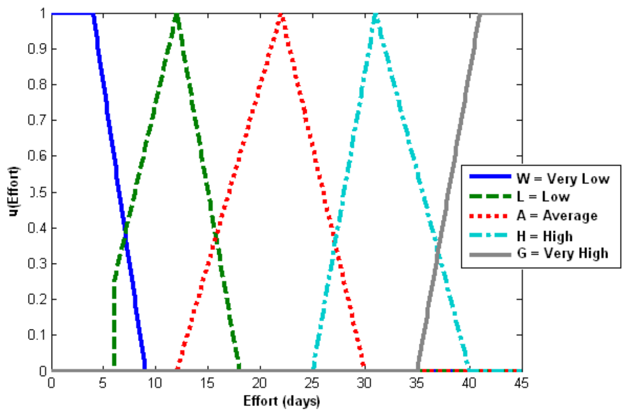

Figure 5: Effort time

Table 1: Fuzzy Rules

$Complexity$	$Size$	$Experience$	$Effort$
L	S	L	W
L	M	L	W
L	L	L	L
A	S	L	L
A	M	L	A
A	L	L	H
H	S	L	H
H	M	L	G
H	L	L	G
L	S	A	W
L	M	A	L
L	L	A	A
A	S	A	L
A	M	A	A
A	L	A	H
H	S	A	A
H	M	A	H
H	L	A	G
L	S	H	W
L	M	H	L
L	L	H	H
A	S	H	A
A	M	H	A
A	L	H	H
H	S	H	A
H	M	H	H
H	L	H	G

The experience of developers related to the C++ programming language and to the project is determined according to the number of years of professional experience. The universe of discourse considered in this work varies from one to four years in the current project.

The size of the alteration uses the number of modules (cpp and c files) that will be altered and created, that is, the sum of those numbers is represented by the linguist variable "Size".

A specific software task may be estimated by the system entering, for example, "0.9" as the complexity (this value was calculated with the formula technical complexity factor), which implies a high complexity; "35" as the size (after estimating the number of files to be altered), which characterizes a medium size; and then, "1" as the experience of developer characterizing an experience between low and average. Based on these inputs and the Fuzzy linguistic rules, the system returns the estimated effort.

The rules from the Fuzzy inference system that contain the linguistic variables and their term sets related to the project are described in Table 1. It is important to note that those rules were adjusted or calibrated, as well as all pertinence level functions, in accordance with the tests and the characteristics of the project.

All rules use the connective "and" between the antecedents, as indicated in the example below regarding the first line of Table 1.

If Complexity is "Low" and Size is "Small" and Experience is "Low" Then the Effort is "Very Low".

The generalized *Modus-Ponens* inference process was used in the Fuzzy calculations, in addition to the *Max-Min* composition operator, the *Mandani* implication operator, and the *Maximum* operator for aggregation. The defuzzification of the output "effort" used the *Mean of Maximum* technique in this work because the resulting values were more appropriate when compared to the other evaluated techniques (*Center of Area* and *First of Maximum*).

As mentioned previously, the pertinence functions, as well as the rules of the inference process, were adjusted. This adjustment implies altering the pertinence functions (Figure 2, 3, 4 and 5) and/or the linguistic rules (Table 1). It is quite usual to adjust results when modeling Fuzzy system, more specifically when the feeling of the developer is needed instead of an exact mathematical methodology as used for training neural networks, for example.

Twenty tasks or characteristics previously executed in the Syscon project were analyzed to define the adjustment. Then, the attributes (from the tasks and the developer) were included one after the other in the system and, in the end, the resulting output was monitored. After carefully analyzing those twenty tests, the adjusted pertinence functions and rules that were more appropriate for all of the twenty tasks were kept.

Observe that the twenty tasks or characteristics used for adjustment are not similar when compared to each other or to the new developed tasks, which validated the proposed estimator as detailed in the next sections.

4 Results

Once the Fuzzy system was developed, and the pertinence function and linguistic rules were included and adjusted, ten new, original tasks, not similar to the Syscon project, were estimated. These tasks will validate the proposed estimator. The estimation of the tasks has been executed for eleven days. Table 2 shows the values for each new task.

Table 3 shows the difference among the time estimated using UCP (Use Case Points) and Fuzzy, and the time actually taken by the developers, for the same tasks described in Table 2. EET is related to the Effort Executed Time, EUCP to Error UCP and EF to Error Fuzzy.

The information related to the UCP estimation was calculated considering one day of effort for each Use

Table 2: Estimated Characteristics

$Task$	$Complexity$	$Size$	$Experience$	$Effort$
1	0.11	9	0.4	2.6627
2	0.51	20	2	21.4014
3	0.15	15	4	3.2432
4	0.98	42	2	31.4314
5	0.53	12	1	11.3313
6	0.38	25	4	21.6016
7	0.80	22	4	31.7918
8	0.23	56	2	21.4815
9	0.64	8	2	11.5315
10	0.2	10	2.6	2.9830

Case Point. Notice that the adjustment process was also executed for the UCP estimation technique, which has been routinely applied at Smar for three years.

Table 3: Estimated versus Executed

$Task$	UCP	$Fuzzy$	EET	$EUCP$	EF
1	2.323	2.6627	2.50	-7.0800	6.5080
2	29.23	21.4014	18.0	62.3889	18.8967
3	5.04	3.2432	3.40	48.2353	4.6118
4	29.07	31.4314	29.50	-1.4576	6.5471
5	13.78	11.3313	9.50	45.0526	19.2768
6	16.98	21.6016	18.50	-8.2162	16.7654
7	26.86	31.7918	27.00	-0.5185	17.7474
8	17.19	21.4815	20.30	-15.3202	5.8202
9	9.89	11.5315	16.30	-39.3252	-29.2546
10	3.10	2.9830	2.50	24.0000	19.3200

The error between the Fuzzy estimation and the real effort and the error between the UCP estimation and the real effort were calculated for each task, and the percentage will help the analyses of the results.

5 Analyses of the Results

Since this is a Fuzzy estimator, the system output is composed by the Fuzzy algorithm through the rules associated to the three inputs, in accordance with the exact values provided. However, the input "Complexity" had the higher weight in most of the rules, among the three inputs considered in this work. This weight was determined by the rules from the inference system (Table 1) /labeltable1. Tasks number 6 and 7 in Table 2 can clearly show the influence of the complexity: the inputs "Size" and "Experience" have close values but the variable "Complexity" was "0.38" and "0.80", respectively. The complexity increased the amount of days of effort from "21.6016" to "31.7918".

Yet, for other projects, the feeling of the Fuzzy designer is essential to distinguish the most significant parameter, that is, the parameter that should have the higher weight. The most significant parameter will be determined after executing the tests of adjustment for the system and the analysis of the response.

Particularly in this work, an output with more terms or Fuzzy sets provided a better performance due to the high granularity (or discretization with more points) demanded from the results.

Most of the tasks planed with the Fuzzy estimator resulted in a more accurate estimative when compared to the UCP technique, and also compared to the time spent in executing the tasks. Among the ten analyzed tasks, only three of them (4, 6 and 7) resulted in a more accurate estimative using UCP estimation.

For some of the planed tasks, for example, tasks 2, 5, 9 and 10, the error percentage was high but the Fuzzy estimation is still better than UCP because the estimation was executed right at the first steps of the project lifecycle (a short-term estimation), and also because the technique was used to estimate tasks that have never been developed in a similar manner.

6 Conclusions

This article demonstrates that the effort estimation technique using Fuzzy may be a good alternative when estimating tasks with few previous references, because in this case there is no similar historic information data and/or developers intend to estimate tasks in a short-term deadline.

Evidently for long-term estimation, other methodologies (traditional or not) result in more accurate deadlines, since there are more input references from estimating models, for example, when factors use function points, COCOMO and multiple regression. On the other hand, considering the software development process model adopted by the company, a long-term estimation is not feasible because the development team is required to estimate the effort in days.

Another important detail in this work that differs from researches in the area is the type of the inputs selected for the proposed estimator system (complexity, size and experience), which would theoretically provide a lower accuracy when compared to techniques demanding fully detailed analyses, such as function points, COCOMO or multiple regression. However, this article shows that these inputs are significant for the estimation using Fuzzy systems, as demonstrated by the results, and, moreover, they are simple, easy and fast to be obtained in order to determine the estimation in the shortest time.

There is a decisive problem on how to interpret the development experience of different analysts, which causes the proposed methodology very dependent on the analyst and the project. There is no recipe to adjust the inference rules and the Fuzzy pertinence functions. Tests are crucial to discover the system model and, at the end, the best option for all obtained responses should be used. This is exactly the proposal of the Fuzzy logic for smart computing systems: providing a method that brings the computational decision process near to the human decision, through linguistic rules that generate fuzzy systems.

References

[1] Boehm, B. W. Software engineering economics. *Prentice Hall*, 1981.

[2] Bradley, M. Function point counting practices manual release 4.1. *International Function Point Users Group*, 1999.

[3] Brooks, F. P. The great challenges for half-century-old computer science. *Journal of the ACM*, 50(1):25–26, January 2003.

[4] Gray, S., A.; MacDonell. Applications of fuzzy logic to software metric models for development effort estimation. *IEEE Proceedings of NAFIPS'97*, 1997.

[5] Huang X., R. J., Ho D. and L.F., C. A neuro-fuzzy tool for software estimation. *20th IEEE International Conference on Software Maintenance*, page 520, September 2004.

[6] Huang X., R. J., Ho D. and L.F., C. A soft computing framework for software effort estimation. *Soft Computing, Springer*, 10(2):170–177, January 2006.

[7] Huang X., R. J., Ho D. and L.F., C. Improving the cocomo model with a neuro-fuzzy approach. *Applied Soft Computing Journal*, 7:29–40, 2007. Elsevier Science.

[8] Kemerer, C. F. Reliability of function points measurement: a field experiment. *Communications of the ACM*, 36(32):85–97, 1993.

[9] Kusumoto, F. I. K. H. S. M. Y., S.; Matukawa. Estimating effort by use case points: method, tool and case study. *Proceedings. 10th International Symposium on Software Metrics*, 2004.

[10] Martin, J. Y. C. T. A., C.L.; Pasquier. Software development effort estimation using fuzzy logic: A case study. *Sixth Mexican International Conference on Volume*, pages 113 – 120, September 2005.

[11] McCabe, T. A complexity measure. *IEEE Transaction on Software Engineering*, SE-2:308–320, December 1976.

[12] Mohagheghi, B. C. R., P. Anda. Effort estimation of use cases for incremental large-scale software development. *ICSE 2005. Proceedings. 27th International Conference on Software Engineering*, 2005.

[13] Pressman, R. Software engineering. *McGraw Hill*, 5th ed, 2001.

[14] RUP. <http://www.wthreex.com/rup/>. 2007.

[15] Smar. <www.smar.com>. 2007.

[16] Smith, J. The estimation of effort based on use cases. *Rational Software*, Cupertino, CA.TP-171., 1999.

[17] Xia W., H. D. and L.F., C. Calibrating function points using neuro-fuzzy technique. *21st International Forum on Systems, Software and COCOMO Cost Modeling*, November 2006.