SADIO Electronic Journal of Informatics and Operations Research

http://www.sadio.org.ar/ejs/

vol. 14, no. 1, pp. 22-41 (2015)

Method of Estimating Costs of a Software Web Product

Jorge E. Diaz Villegas
Universidad de la Frontera, CEIS, Chile
jorge.diaz@ceisufro.cl

Gabriela Robiolo
Universidad Austral, FI, Argentina
grobiolo@austral.edu.ar

Abstract. The costing of a product is a key factor in the marketing process. Its proper calculation can attract customers, which will ensure a company's life and its business expansion. These considerations have driven the Centre for the Study of Software Engineering at Universidad de la Frontera (CEIS-UFRO) to develop a method to define the cost of a web software product, based on use cases and productivity. This method is adaptable to the particular characteristics of any development process, any development team, any product and any company. This article describes the method and performs an initial validation by describing a quasi-experiment designed for Web applications developed by groups of three to five people. We have proved that: a. the method may be reproduced, b. effort estimation is sensitive to the definition of productivity, c. the subjectivity introduced by the estimators does not invalidate the method. For a complete validation of this method, different web products and a larger number of estimators with different levels of experience should be incorporated in a future replication.

Keywords: effort estimation, UCP, software costs, experimental software engineering

1 Introduction

The possibility to early estimate effort during the development process of a product is important for software development companies since effort is one of the main cost factors of a software product [1]. Effort definitely depends on product size, and these two variables depend on the human resources available for product development, so it is of

utmost importance for a company to be able to accurately estimate effort as soon as the project starts.

Although an exact estimation is desirable for all companies, we have noted it is particularly critical for small and medium–sized companies because an error in the estimation of the cost of a major project can drive a company to financial crisis. Besides, since appropriate cost calculation can gain customers -which will ensure both the life and expansion of a company-, determining the cost of a software product is a key issue in the marketing process of any product development.

Consequently, different methods to early estimate effort have been developed, based on a variety of concepts, such as the final size of a product (Line Code [2]) and the expected functionality of a product (Function Points [3] COSMIC [4]). The drawback of these methods [3, 4, 5, 6] is they generally do not include all aspects required for cost calculation. In addition, other specialists have developed methods based on use case requirements definition, such as Use Case Points (UCP) [5]. The advantage of estimating based on use cases is that it allows developers to make an early estimate, with a deeper understanding of the requirements to be developed.

These are the facts that have motivated the Centre for the Study of Software Engineering at Universidad de la Frontera (CEIS-UFRO), Chile, to develop a method to estimate the cost of software web products, based on early estimation techniques constructed on use cases and productivity, which includes complementary aspects that may lead to defining the cost of a product.

This method has been experimentally developed over the past five years from the experience gained at CEIS-UFRO from the development of more than ten projects of Web software products and by adapting techniques for early effort estimation. In this context, the phrase "web products" refers to the development of software services that users can access through a Web server via the Internet or an intranet, by using a browser. It has been successfully applied to Web projects bigger than 1,000 MH developed by teams of three to five developers. These project characteristics are standard for the small and medium-sized companies in Temuco (Chile), where Universidad de la Frontera is located. It is important to note that the CEIS-UFRO method is adaptable to the particular characteristics of any development process, of any development team, and to whichever techniques and tools may be used for the development of any product by any company.

In this article, which is an extension of a previous one [7], the CEIS-UFRO method is explained in more detail and the related works are deeply analyzed. Moreover, in this paper we present a quasi-experiment through which a set of estimates made on a product is analyzed and compared to the actual value of the required effort. Finally, we discuss related work and draw our final conclusions.

2 Product Cost Calculation Method

This customizable method comprises two distinct tasks: a) estimation of the effort required to develop a product, measured in man-hours (MH) and b) estimation of the product cost. To accomplish the first task, the first step is to define the workflows of

the lifecycle of the product and the percentage of the effort that each workflow represents in the whole lifecycle. Afterwards, the productivity of the workgroup is defined by using previously recorded data or by estimating such value if previous data do not exist. Then, the product size is estimated, based on a use cases technique [5]. Finally, based on the product size and the workgroup productivity, the effort is calculated. The second task is to calculate the cost of the product, using the product effort as an input.

2.1 Effort estimation

To complete the first task, which is to calculate the estimated effort, it is necessary to perform the following steps:

Define the percentage of effort of each workflow within the whole product effort. Once the development model, tools and techniques have been defined, the estimator identifies the workflows of each product development and decides what percentage within the total effort each workflow will require. For example, Table 1 suggests the intervals for a Web project, which are based on the historical effort records of the CEIS-UFRO development teams. These intervals are defined for a particular development strategy and they are based on the historical registration of effort for similar projects, in a span of ten years. Note that each development team has set the percentages and the definitions of the stages based on their own development strategies, and on their product and project characteristics.

Table 1. Range of partial percentages of effort for a software Web product

Workflow	Range	Quasi-experiment Percentage
Overview	(3%-5%)	5%
Requirements	(17%-25%)	25%
Analysis	(10%-15%)	15%
Architecture and Technical Design	(10%-15%)	10%
Construction	(20%-30%)	30%
Testing	(10%-20%)	15%

Construction is the workflow which is the easiest and most tangible to measure, so construction is the basis on which the total effort that is required for the entire lifecycle is estimated. In fact, the effort of each use case is calculated as the effort of a set of artifacts, where the effort of each artifact is estimated for the construction workflow and then extrapolated to the other workflows (see columns Mean $Effort_C$ and Mean Effort in Table 3).

Productivity Definition. Productivity (PR) is defined as the quotient between effort and UCP [8]. There are two possible scenarios in which the calculation will take place:

a. The productivity of the working group is known. Therefore, the average historical values recorded for the Construction workflow of the artifacts are used.

b. The productivity of the working group is unknown, so productivity is calculated based on the experience of skilled development engineers, by disaggregating the use cases into artifacts and estimating the effort for each artifact. Table 2 shows the description of the different artifacts. Seven types of artifacts may be identified, based on the experience developed at CEIS-UFRO. The software artifacts definition depends on the software architecture, the language features, and the type of tools (framework), libraries and reusable components. The following steps are performed in order to obtain the productivity value:

- *Estimate the average effort required by each type of artifact in the workflow during the Construction stage*. One or more estimators define the value of the effort required for each type of artifact and then the average of their estimation values is calculated (see an example in Table 3 where four estimators -EE1 to EE4- estimate the effort based on their experience and the value of Mean $Effort_c$ is calculated).
- *Calculate the total effort for each artifact type*. The total effort is the effort required during the whole product development cycle, not only during the construction workflow. By applying the rule of three and considering that the mean effort values are defined for the Construction workflow, which in fact, is the 30% of the total effort, it is possible to obtain the effort value for each artifact in the total lifecycle (see columns Mean $Effort_C$ and Mean Effort in Table 3).
- *Translate Effort into Use Case Points*. The highest value estimated for an artifact is considered the reference unit and the rest of the values are proportionally calculated (see columns Mean Effort and ARP*j* in Table 3).

  Table 3 illustrates how to calculate productivity for a Web application. In this example, the Construction workflow represents 30% of the total life cycle. The highest effort value in this set of artifacts is 23.33, which is the value for "Exceptions". Thus, 1 Use Case Point is 23.33, and the values of the other artifacts of the Use Case Point (ARP*j*) are proportionally calculated with the rule of three.
- *Set the productivity value*. The team productivity is equal to the highest effort value divided by one UCP.

Table 2. Artifacts descriptions

Artifact	Description
Entry Screens	Data entry forms
Output Screens	Show results, actions and reports
File Management	Manages printing, uploads and / or downloads PDF, JPG, DOC files

Data transaction	Manages database query, entering, updating and deleting data
Exception	Alternative data processing which implements exceptions
Operation	Performs mathematical calculations and logical operations which are part of the business rules
WebService	Enables interoperability of modules (SOAP, REST)

Table 3. Artifact Effort Estimation and its translation into percentage of Use Case Points

Artifacts	EE_1	EE_2	EE_3	EE_4	Mean $Effort_C$	Mean Effort	UCP% (ARP*j*)
Entry Screens	4	5	6	5	5.00	16.7	0.71
Output Screens	2	2	2	3	1.50	5.0	0.21
File Management	2	3	3	3	2.75	9.2	0.39
Data transaction	3	3	4	4	3.50	11.7	0.50
Exceptions	5	7	7	9	7.00	23.3	1.00
Operation	2	2	3	3	2.50	8.3	0.36
WebService	3	4	4	5	4.00	13.3	0.6

Estimating the size of the software product. The UCP value is calculated using formula 1.

$$UCP=UCPuc+UCPa \times TTF \times TEF. \quad (1)$$

in which UCPuc is the value of the UCP for the set of use cases, UCPa is the value of the UCP for actors, TTF is the Total Technical Factor and TEF is the Total Environment Factor.

UCPuc. For each use case, the number of artifacts is identified and classified by type. The total number of UCPuc is obtained using Formula 2.

$$UCPuc = \sum_{\substack{i=1 \\ j=1}}^{\substack{n \\ m}} ARNij \times ARPij \; . \quad (2)$$

in which ARN*ij* is the number of artifacts, ARP*ij* is the percentage of Use Case Points assigned to each artifact (see example in Table 3), i identifies the use cases and j the types of artifacts.

Actors. The UCPa value is equal to the number of actors multiplied by the assigned weight (see formula 3).

$$UCPa = \sum_{i=1}^{4} ANi \times AWi \; . \quad (3)$$

in which AN*i* is the number of actors of a certain type and AW*i* is the weight, according to the type of actor. Table 4 shows the weights assigned to the different types of actors. In this table, there is an additional type which was not present in the types of actors described in [5], which shows the original definition.

Table 4**.** Description of the types of actors and their weights

Type of Actor	Description	Weight
Simple	Another external system communicates with a defined API. The API may be implemented in a standard protocol, DLL, REST, SOAP, RPC.	1
Medium	Another system communicates through a proprietary protocol implemented on TCP / IP or corresponds to a user who communicates via command line.	2
Complex	An end user who interacts through a Web GUI.	3
Very Complex	A user interacts through a graphical interface of a Web application for special or administrative functions.	4

Technical Total Factor (TTF). Technical Factors (TF) and the weight of the factors that define the complexity of the product to be developed are the same as those reported in [5]. The estimators assign to each factor an influence value which ranges between 0 and 5, where 0 is no influence and 5 is great influence. The TTF is calculated using formula 4 [5].

$$TTF = 0.6 + (0.01 \times \sum_{1}^{13} TWi \times TIi) . \quad (4)$$

TWi is the TF weights and TIi is the influence value of each TF. Their influence on the project is evaluated in a range of 0-5, from none to very high influence.TF are described in detail in Table 6 (see Appendix).

Total Environment Factor (TEF). The Environment Factors (EF) are those included in [5], except for two, which were added later: "rigid planning" and "maturity of the development process". The estimators assign an influence value between 0 and 5 -from 0, no influence, to 5, great influence-. The TEF is calculated using formula 5 [5].

$$TEF = 1.4 - (0.03 \times \sum_{1}^{10} EWi \times EIi) . \quad (5)$$

where EWi are the weights of each EF and EIi the influence of each EF. EF are described in Table 7 (see Appendix).

Product effort calculation. The estimated effort (EE) of the product is calculated using formula 6.

$$EE\ [MH] = UCP \times PR \frac{[MH]}{[UCP]} . \quad (6)$$

where PR is the previously defined productivity.

2.2 Product cost calculation

Finally, the second task may now be performed, that is, calculating the cost of the product (PC), which is calculated using formula 7.

$$PC = (EE \times MHC) + ACH + EM + FC. \quad (7)$$

where,
EE is the estimated effort from formula 6.
MHC is the mean cost of a work hour of the human resources.
ACH is additional expenses of the project (travel, subcontracting services, special training, etc). These rates depend on the nature of the project and its location.
EM is Margin of error, average percentage calculated based on the perception of completeness of the use cases. The error decreases the more detailed the use cases specifications are. It helps to manage the contingencies the project may incur into. A good overview should include between 70% and 90% of the final system's functionality.
FC is fixed costs, all projects have fixed costs associated with physical infrastructure, administrative support staff and management. It ranges from 20% to 30% of the total cost of the project, depending on the company circumstances.

3 Quasi-experiment

In order to evaluate the effectiveness of this estimation method, a quasi-experiment [9] was designed. The effort estimation results obtained by using the first phase of the CEIS-UFRO method (cfr. 2.1) were compared to the actual hours spent on the development of a given product. In turn, the estimation errors thus obtained were compared to the errors reported for the UCP [10-13] method. The hypotheses to be verified were:

H0: The mean MRE (MMRE) obtained by applying the CEIS-UFRO estimation method equals the MMRE reported when using the UCP method.

H1: The MMRE obtained by applying the CEIS-UFRO estimation method is lower than the MMRE reported by the UCP method.

The non-parametric Wilcoxon method [14] was selected to test the hypotheses for a significance level of 0.05, because the distribution of the population of interest was assumed not to be normal.

In addition, the acceptability of the CEIS-UFRO method was evaluated by applying the criterion defined by Conte, Dunsmore, and Shen [2], which defines the prediction quality (PQ) for a set of n projects as the quotient between k and n, where k is the number of projects whose MRE is less than or equal to 25%. Estimation methods that have a PQ higher than 75% are considered to be acceptable [2].

The effort estimates were made in the context of a master's course (Software Projects Management) at Universidad de la Frontera. Nine practitioners (experimental subjects) who worked in the software industry, with an average experience of three years, participated in this experiment.

Each student applied the CEIS-UFRO estimation method to a project that builds a Physical Assets System (PAS) - experimental object -, which aims to record the physical assets of a company using a PC or a mobile artifact. This system was developed at the Department of Technology and Systems (T&S) of Universidad Austral. The requirements definition, based on 15 use cases and written at T&S, which was to be used for estimating was presented in a 21 page document. Together with the definition of requirements, the participants were given an overview of the deployment tools, language and database used in the actual system development. The PAS actual MH were calculated based on the tasks of construction workflow done on PAS. These values were obtained from a timesheet used to manage the T&S department. The construction workflow had been defined as 30% of the total product development effort, so the value corresponding to the effort of the entire product development was easily obtained.

The controlled aspects were that the actual effort of PAS was not released to the experimental subjects and that the use cases employed to estimate effort were the ones written by the actual PAS development group.

The CEIS-UFRO estimation method was applied with the following characteristics:

a. *Percentage allocated to each workflow*. The percentages of effort for the different workflows of the lifecycle were defined, and agreed on, based on the experience of the experimental subjects. Table 1 shows the performed selection.
b. *Productivity*. The values of Mean $Effort_C$, Mean Effort and UCP% in Table 3 were used, according to which, the selected productivity of the development team was equal to 23.33 [MH / UCP].

3.1 Results Description

A summary of the estimates made by the experimental subjects for the whole product is shown in Table 5. The calculation of the relative errors (REr), the magnitude relative error (MRE), mean (M) and standard deviation (SD) of the estimated effort were made using the actual PAS development effort, which was 1307 MH.

Table 5. Summary of effort estimations

ID	TTF	TEF	UCPuc	UCPa	UCP	EE	REr	MRE
1	1.02	0.86	39.07	10.00	42.83	999.47	24%	24%
2	0.98	0.92	53.50	10.00	57.25	1335.87	-2%	2%
3	1.03	0.89	38.32	10.00	44.08	1028.56	21%	21%
4	1.04	1.12	54.50	11.00	75.59	1763.73	-35%	35%
5	1.00	1.27	33.29	10.00	54.48	1271.26	3%	3%
6	0.85	1.13	48.75	10.00	56.43	1316.69	-1%	1%
7	0.76	1.19	44.04	10.00	48.87	1140.30	13%	13%
8	1.04	1.06	36.00	16.00	56.78	1324.87	-1%	1%
9	1.12	0.73	45.00	11.00	45.47	1061.01	19%	19%
M	0.98	1.02	43.61	10.89	53.53	1249.09	4%	13%
SD	0.11	0.18	7.57	1.96	10.07	234.88	18%	12%

The signs of the REr values show that five estimates were sub-valued (positive sign) and four were overvalued (negative sign). MRE values are within the range of [1% - 35%]. The mean and standard deviation of the MRE are 13% and 12%, respectively.

During the estimation process of PAS, all the experimental subjects considered there would be an ACH (see formula 7) in the development of the mobile application, except for the practitioner who made the estimate ID4, which is in fact the estimate that shows the highest error. The practitioner may have overloaded the estimate to compensate for the non-inclusion of the ACH, so due to this circumstance, this value could be considered an outlier.

It was possible to reject the null hypothesis in favor of the alternative one by applying the Wilcoxon method with p-value significance level equal to 0.05 and a p-value of 0.01.

If the PQ is calculated for the total number of estimates, we obtain a value of 88%. If estimate ID4 is not included, the coefficient is equal to 100%. These results are considered to be very good, as estimation methods are acceptable if their PQ values are higher than 75% [2].

It is also interesting to consider the mean and standard deviation of the MRE without including estimate ID4, since a value of 10% is obtained in both cases.

3.2 Discussion

It should be noted that through this experiment we could only compare the actual effort value to those which had been estimated, that is, only the first phase of this method could be put into practice (2.1 section). The reason for this was that it was not possible to calculate the cost of the product since we were unable to obtain the necessary data to do so due to confidentiality reasons. However, although this is a partial analysis of the method, it is still interesting because the estimated effort thus obtained is the basis on which the product cost may be calculated. Furthermore, it is worth noting that determining the mean cost of MH and expected utility are tasks which only companies may perform since they involve dealing with inside information, and besides, they are influenced by market factors, which have not been taken into account in this experiment.

Figure 1 shows the values of REr when the productivity varied within a [5.83-40.82] range. Figure 1 shows that a selected productivity of 23.33 was a good choice, since the MMRE = 13% and the Standard Deviation of MRE (SDMRE) = 12. When a 29.16 productivity was applied, the MMRE value was higher, 21%, and the SDMRE was also higher, 21%, which shows that the biggest errors occurred because the productivity had been over-estimated.

This analysis shows that the selected productivity of 23.33 is suitable for the application used as experimental object, but it also emphasizes the sensitivity of the CEIS-UFRO method regarding a variation in productivity. It is important to note that this productivity was agreed and defined by the quasi-experiment participants, based on their experience, and as a result of a discussion planned within the experiment. Obviously, to successfully apply the method, it is necessary to know the actual productivity value or to be able to establish a productivity close to the actual one. Also, 23.33 lies within the [15MH – 30MH] expected range [15], which depends on the characteristics of the development process. The effectiveness of the method is particularly sensitive to the definition of productivity and to the level of detail and completeness of the use cases.

It is advisable to consider two phases when estimating the cost of a product. The first one is a general preliminary estimate, which gives an idea of the product price, which will justify, or not, further studies. During the second phase, the Product requirements will be thoroughly defined, thus obtaining a set of functionalities (Product Backlog) and the cost.

The values of Tables 1-4 and 6-7 are the result of the implementation and refinement of the CEIS-UFRO method in more than 10 projects developed in small and medium-sized software companies in the public sector. For example, some of the applications thus developed are: management support software for the Regional Government of the Araucaria, which includes a system to apply for funds for sports, culture and civic assistance; an intranet development which centralizes information for the Regional Government; and an application that manages an immunization program for the Chilean Ministry of Health.

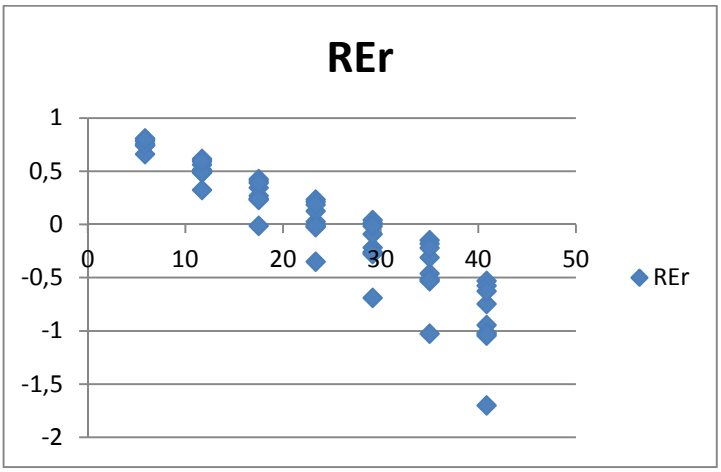

Fig. 1. Variation of REr when the productivity varies in the range of [5.83-40.82].

If this method were applied in a company, it may need to be adjusted to the company's reality: the percentages of effort defined for each workflow (Table 1), the artifacts described in Table 2 and the effort estimation per artifact (Table 3). In the quasi-experiment, the values in Tables 1 to 3 did not vary, but the sensitivity of the method was tested by varying its productivity.

One wonders what the impact on the application of the method would be if the values in Tables 1-3, TF and EF were adjusted to a particular situation; the answer would be that a reduction of the estimation error would be expected.

Finally, with regard to the participants' opinions, they stated that this method was applicable to their work, since the Excel spreadsheet used to facilitate the calculations included several macros that simplified the estimation process. Based on this experience, practitioners can apply the method without further support and adjust the parameters to the particular realities of their teams. For the time being, there is no information about whether the practitioners have adopted the method in their daily practice or not.

3.3 Threats to validity

The use cases were not all described at the same level of detail and some basic use cases did not have any description, for example, a user deletion. This may have introduced a higher error in the estimation; however, this aspect does not invalidate the conclusions of the method, since all the persons involved – both the experimental subjects and the actual team which developed the application- used the same description of requirements. It would be interesting to study the variation of the MMRE if a requirements definition with a higher level of detail were used.

The estimators did not know the working group that actually performed the development of the PAS, but as the method is calibrated for 3-5 persons -a feature which was

shared both by the participants and the actual development team-, this aspect did not affect the experiment results.

It may be wondered whether the method was calibrated, or the productivity was selected, in knowledge of the value of the experimental object's actual effort. The fact is that the estimators that participated in the quasi-experiment in Temuco, Chile, only learnt about the actual effort of the experimental object when the experiment was completed and the data were analyzed in Buenos Aires, Argentina, two months later.

Regarding the influence of the experimental subjects' level of experience, the three experience levels the participants were at (low, medium and high) were considered adequate, though the number of experimental subjects involved (nine) was not. It is expected to compare results obtained by many more estimators, at different levels of experience, in future replications.

Each participant measured the number of use cases artifacts, the number of actors, TF and EF. All the participants used the same percentages of effort allocated to each development workflow of PAS (see Table 1), the same artifacts (Table 2) and UCP size per artifact (see Table 3 ARP*j* column), just as it was recommended by the creators of the method, since the UCP size values were empirically adjusted. The quasi-experiment allowed us to check the reproducibility of the method, and to understand that the subjectivity in the measurement introduced by the participants does not invalidate the method and that the productivity value is suitable for the selected application. But for a complete validation of the method, it would be necessary to perform a new experiment with different Web applications developed by groups of 3-5 people. This replication is also important to make a comparison between the CEIS-UFRO method and the UCP [5] method, in order to have more evidence of the importance of these authors' contribution.

4 Related Work

The CEIS-UFRO estimation method has three important characteristics that became the backbone of the study of related work: it was developed in a web application context, it can estimate the cost of a software product based on the size of such product, and the size measure was obtained from use cases.

There are several authors who have focused their research on Web applications. For instance, Reifer [16] introduced the concept of Web objects, a size measure which extended function points in order to capture the particular characteristics of Web applications. These Web objects added four predictors to function points: links, multimedia files, scripts, and web building blocks. He reported that by using this size measure he had been able to repeatedly predict the size of a Web application with reasonable accuracy. Furthermore, in order to estimate Web project costs, he later developed the WEBMO cost model, which is an extension of the COCOMO II. This method employed a mix of expert judgment and actual data from 64 projects, using linear regression techniques. The application of WEBMO increased the statistical accuracy of his estimating model and allowed him to take the characteristics of Web projects into account, via adjustments that he made to the WEBMO model cost drivers. In another

article, Reifer [17] used data from 46 finished industrial Web projects and obtained predictions which were "repeatable and robust." However, no information was given regarding the data collection or any statistical analysis of the collected data in his two previously mentioned publications, which does not facilitate the evaluation of his conclusions.

As regards measuring size to estimate cost, it is worth discussing Ruhe et al.'s study [18] in which they also used Web objects as a size measure in the COBRA™4 (Cost Estimation Benchmarking and Risk Analysis) method. COBRA is a method that aims to develop an understandable cost estimation model based on a company's specific dataset. It uses expert opinion and data on past projects to estimate development effort and risks for a new project. The author compared the prediction accuracy obtained using COBRA to that attained employing expert opinion and linear regression. The former obtained a MMRE=17% and Pred(25)=75%, giving COBRA the most accurate results, which are, in fact, similar to those obtained in the experiment presented in this article.

Another interesting study is the one performed by Mendes and Mosley [19], who presented a survey of eleven Web cost estimation models which had been described in the literature up to 2005. In their study they highlighted that there is no standard to sizing Web applications since they can be created using diverse technologies, such as several varieties of Java (Java, servlets, Enterprise java Beans, applets, and Java Server Pages), HTML, JavaScript, XML, XSL, and so on. Consequently, they suggested the need for a standard size measure, so that we may better compare and contrast results. We believe that the decomposition of use cases into artifacts, which is suggested in our article, may be considered a solution to this problem, as it may be applied to different types of requirements definition and to different technologies.

Another important aspect of size measuring that Mendes and Mosley emphasized in [19] was the use of automated tools to collect data. They discovered that none of the surveyed papers employed automated tools to measure size, and they believe automated tools are important to reduce the errors inherent to data collection. Finally, the authors urged the Web engineering community to plan and run formal experiments, as these will lead to the building of a wider body of knowledge where findings may be generalized into a wider context. In fact, our method does not use automated tools, which is a limitation that should be dealt with in future work.

It is worth mentioning that estimating the cost of a software product based on the size of such product is a vision shared by several authors [1], [3], [5, 6], which is also shared by the eleven authors included in the above mentioned survey [19].

Regarding the use of use cases to calculate size, we can point out that there are several elements in common between Karner's approach to measure functional size and that adopted by the CEIS-UFRO method, but there are also some differences which are worth highlighting. For example, although the CEIS-UFRO method takes the concept of UCP from Karner [5], the CEIS-UFRO method defines the complexity factors of use cases, the types of actors, and the EF differently. For instance, Karner defines the complexity of use cases in terms of transactions and number of analysis objects, while the CEIS-UFRO method defines it in terms of artifacts. Besides, the CEIS-UFRO method considers four levels of complexity for actors, adding one more level to those defined

by Karner. As regards EF, CEIS-UFRO adds two environmental factors to those proposed by Karner: “maturity of the development process” and “rigid planning” and Karner’s “familiar with objector” has been changed to “familiar with the project domain” by CEIS-UFRO. However, the TF are quite similar in both methods; the only differences lie on the weights for the factors “distributed system” and “no reuse source code”.

There were other authors who also worked with use case points. For example, Diev [20] defined a heuristic for the implementation of use case points and rules for UML models, so that use case points may be counted. Anda [21] presented an experience which used use case points to estimate the effort of a specific system. In his study, there were four different companies, based on the same definition of requirements, which developed four similar applications. Each company was asked to use a different development process in order to compare the different effort values obtained. The article shows a significant difference between the estimated effort, which was calculated based on use case points, and the actual effort of the developed product, which led to the conclusion that the development processes affect the cost estimates more than usually expected. In our empirical study, we do not have this flaw, as the actual PAS developers worked with the Unified Process [22] and the estimators then took into account that this process had been used. Moreover, Mohagheghi et al. [13] introduced some variations in the use of use cases in a big scale industrial software development, adapting the Use Case Point method. They believed that each transaction and each alternate flow in a use case is, in fact, a new use case. In a similar way, Yavari et al. [23] introduced other metrics to determine the complexity of use cases, and they focused on the specification of use cases and the flow of events.

Likewise, Braz and Vergilio [10] adapted Use Case Points to be used with more detailed use cases, introducing the USP (Use Case Size Points) as a size metric which may measure the internal structure of a use case. The USP measure the functionality of the structures and sections of a use case, by counting the number and weight of scenarios, actors, and pre and post-conditions. They also introduced the FUSP (Fuzzy Use Case Size Points), another metric that adds concepts from the "Fuzzy Set" Theory to create a gradual classification that may work better with uncertainty.

There are two important aspects which should never be overlooked regarding the employment of use cases as a size measure: the differences in size and level of detail. These occur because different developers write use cases using different modalities. For instance, Anda [12] stated that the use of Use Case Points is affected by different aspects of the structure model of use cases -for example, the use of actor generalizations, the use of included and extended use cases, and the level of detail in use case descriptions-. To improve the estimation models based on use cases, Anda [21] recommended a balanced level of detail to be used in the writing of use cases when the use case model must be used as a basis for estimation. He suggested using good examples of use cases or defining specific guidelines for the use case modeling process. In fact, the CEIS-UFRO method solves the variation of the dimension of the use cases and the level of detail with which each use case is decomposed by defining a set of artifacts.

Finally, there is a study [24] that used the two variants of UCP, i.e., with or without unadjusted actors weights, which provided similar prediction accuracy. The author also

notes that the number of adjustment factors (21) could be reduced to 6 (2 EF and 4 TF), an aspect that leads us to consider the possibility of simplifying the method in future work, without losing precision in the estimate.

To conclude, we may state that by applying the CEIS-UFRO method we could obtain results similar to the ones in previous publications. Additionally, the decomposition of uses cases into artifacts has several advantages: a standardized web size measure may be found, the use case granularity problem may be solved and the measure process may be automated. CEIS-UFRO did introduce some modifications to the original UCP method, but our method may be simplified in the future.

5 Conclusion

This article has introduced a cost calculation method which is based on the size of a product. In fact, the application of this method involves collecting historical data that can feed the process of estimating the costs of a product by adjusting the value of the working team actual productivity.

It is a method that is adaptable to the particular characteristics of any development process -as it defines the percentage of effort to be assigned to each workflow of the development process lifecycle-, to any development team -their historical productivity is used or estimated-, to any product, since each use case size is measured, and to any company, as particular cost policies may be incorporated.

The application of this method generates a feedback model which improves the estimation error, which –in turn- has a positive effect on the commercial and production activities of the software company.

The method introduced in this paper has been implemented in Web software projects involving over 1,000 MH, performed by teams of 3-5 full time workers. If the CEIS-UFRO method is applied to develop bigger products, it is recommendable to divide them into sub-products of such size, when estimating their costs. The practitioners who participated in this quasi-experiment were able to learn and partially apply the method, as well as understand the importance of using techniques and methods of software engineering in industrial activities.

This quasi-experiment has verified the reproducibility of the proposed method and it should be considered as the first effort to validate the method. The quasi experiment has made it evident that effort estimation is sensitive to the definition of a given productivity and that the subjectivity introduced by estimators does not invalidate the method.

A full validation has not been performed yet, but different types of Web applications, a greater variety of estimators, and the calculation of the product cost will be added in future replications. It is also necessary to compare it to the application of the UCP effort estimation method [5] and to COSMIC [4] in future studies, in order to assess the value of the contribution of these authors.

Appendix

Table 6. Technical factors

Technical factors	Weight	Description
Distributed System	1	Product Architecture may be centralized or distributed. Interoperability with other simple or complex architectures may be included.
Response Time	1	Faster system responsiveness for users is a nontrivial aspect which depends on the product architecture and the level of concurrency. For a low response time, a major factor value has to be fixed.
Efficiency from user viewpoint	1	The functionality represents an optimization and improves the efficiency of the current processing. Actual product processing has to be improved.
Complex processing	1	The product included complex algorithms, high levels of analysis or verification, and the use of complex databases, using OLAP or cubes or Data Warehouse.
Reuse source code	2	Product reuses code such as library or components. A lower level of code reuse defines a lower value factor. For example: for 100% reuse, the influence value is 5, for 80% reuse it is 4, for 60% it is 3, for 40% it is 2, and for 20% it is 1. If there is no reuse, the value is 0.
Installation Ease	0.5	Product needs to implement tools that make the installation by end users easy. If the product works with interfaces, and a special manual has to be written, the value is greater. If the product is installed by the technical team, the influence value is lower.
Usability	0.5	Level of usability or UI validation. If usability for non-expert users or a strong UI validation is required, the value is higher.
Portability	2	The product must work on multiple operating systems and / or Web browsers. Higher portability and compatibility defined higher values. Example: one browser portability, value = 1, two browsers, the influence value is 2, an if it works on all browsers, the influence value is 5.
Easy to change	1	The product must allow customization or change in the future. Higher changeability or components flexibility define higher values.
Concurrency	1	The product will be used by many users simultaneously. The higher the concurrency, the higher the value. For example, an application with an open access has an influence value of 5.

Safety features	1	Data encryption or electronic signature defines a higher value of safety. If the product requires high levels of safety, the value will be 5.
Accessible by third parties	1	The API definition thought which external libraries, components and services require the access to the product. The higher the amount of accessibility elements required, the higher the value of the factor must be.
User training	1	The more user training and support required in the deployment phase, the higher the value will be.

Table 7. Environmental factors

Environmental factors	Weight	Description
Familiar with the project domain	1.5	Level of experience and knowledge of the development team about the problem domain. If the development team has high knowledge and experience, the influence value to be assigned will be 5 and if it has no experience and knowledge, the influence value will be 0.
Application Experience	0.5	Level of experience and knowledge of the development team regarding the tools used in product development, for example .NET, JAVA, PHP, ORACLE, etc.
Experience in OOP, AOO, DOO	1	Level of experience and knowledge of the development team about object-oriented analysis, design and programming. Development groups which are in a learning stage will be assigned lower values.
Analysis Capacity	0.5	Level of experience, ability and knowledge of the team leader. If the team leader has little experience, a lower influence value is assigned, on the contrary, if the project leader has wide capacity and experience, a high influence value is assigned.
Motivation	1	Level of motivation and stability of the development team and organizational climate. The higher the level of team consolidation, stability and motivation are, the higher the value to be assigned.
Development process maturity	1	Identifies the level of development process consolidation, the use of best practices, change control, quality management, change management, configuration management, etc. A high maturity level defines a higher value.

Clear and stable Requirements	2	Level of clarity of the requirements and stability over time. It also quantifies the level of collaboration of the project partners (customers and users). Low clarity and / or changes in the requirements, or if there is poor partners' collaboration, a low value is defined.
Part Time Professional	-1	External professionals working part time in the development team. Higher external support represents higher risk, therefore, the weight is negative. Higher external collaboration defines a higher value.
Rigid Planning	-1	Projects that have a rigid schedule with fixed delivery date. The more rigid and upcoming the dates are, the higher the influence value.
Difficulty of the building tool	-1	Frameworks or languages require different capacities and experience, for example, the use of JAVA requires more knowledge and experience than PHP. The higher the difficulty is, the higher the risk, the higher the value to be assigned.

Acknowledgements. We are grateful to Universidad de la Frontera and Universidad Austral for their support.

References

1. Agrawal M. and Chari, K.: Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects. IEEE Transactions on Software Engineering, Vol. 33, N. 3, pp. 145-156, March (2007)
2. Fenton, N. and Lawrence Pfleeger, S.: Software Metrics. PWS Publishing Company, (1997)
3. ISO/IEC 20926: 2009, Software engineering – IFPUG 4.1Unadjusted functional size Measurement method – Counting Practices Manual, International Organization for Standardization, Geneva, (2009)
4. ISO/IEC19761:2011, Software Engineering -- COSMICFFP– A Functional Size Measurement Method, ISO and IEC, (2011)
5. Karner, G.: Metrics for Objectory. Diploma thesis, University of Linköping (1993)
6. Boehm, B.W., Horowitz, E., Madachy, R., Reifer, D., Clark, B.K., Steece, B., Winsor A., Brown, Chulani, S. and. Abts, C.: Software Cost Estimation with Cocomo II. Prentice Hall, (2000)
7. Diaz Villegas, J.E. y Robiolo, G.: Método de estimación de costos de un producto de software Web, in proceedings of 15 Simposio Argentino de Ingeniería de Software, JAIIO 2014, Buenos Aires, Argentina (2014)
8. Jørgensen, M., Indahl, U. and Sjøberg, D.: Software effort estimation by analogy and regression toward the mean. Journal of Systems and Software, 68(3), pp. 253-262, (2003)
9. Jedlitschka, A., Ciolkowski, M.and Pfahl, D.: Reporting Experiments in Software Engineering, Guide to Advanced Empirical Software Engineering, pp 201-228, (2008)

10. Braz M, Vergilio S.: Effort Estimation Based on Use Cases. 30th Annual International Computer Software and Applications Conference (COMPSAC'06), 221-228, (2006)
11. Anda, B.C.D., Benestad, H.C. and Hove, S.E.: A multiple-Case study of effort estimation based on use case point. In Fourth International Symposium on Empirical Software Engineering (Australia, November 17-18, 2005) ISESE'2005, IEEE Computer Society, 407–416 (2005)
12. Anda, B., Angelvik, E. and Ribu, K.: Improving Estimation Practices by Applying Use Case Models. Lecture Notes In Computer Science, Vol. 2559, 383-397, (2002)
13. Mohagheghi P., Anda B. and Conradi R.: Effort estimation of use cases for incremental Large-scale Software development. Proceedings of the 27th international conference on Software engineering**,** 303 – 311 (2005)
14. Montgomery, D. and Runger, G.: Probabilidad y estadísticas. McGraw-Hill (1996)
15. Sparks, S. and Kaspcynski, K. The Art of Sizing Projects, Sun World. 1999.
16. Reifer, D.J., http://www.crosstalkonline.org/storage/issue-archives/2002/200206/200206-Reifer.pdf
17. Reifer, D.J. Web development: Estimating quick-to-market software. IEEE Software, Nov - Dec, pp. 57-64, (2000).
18. Ruhe, M., Jeffery, R., & Wieczorek, I.: Cost estimation for Web applications.Proceedings ICSE 2003, pp. 285-294, (2003)
19. Mendes, E., & Mosley, N.: Web Cost Estimation. Web engineering: principles and techniques, 182, (2005)
20. Diev, S.: Use cases modeling and software estimation: applying use case points. CAN Software Engineering Notes, Vol. 31, N. 6, 1-4 (2006).
21. Anda, B., Dreiem, H., Sjøberg, D. and Jørgensen, M.: Estimating Software Development Effort Based on Use Cases-Experiences from Industry. Lecture Notes In Computer Science, Vol. 2185, 487 – 502, (2001)
22. Jacobson, I., Booch, G. and Rumbaugh, J.: The Unified Software Development Process. Addisin Wesley, (1999)
23. Yavari, Y., Afsharchi, M. and Karami, M.: Software Complexity Level Determination Using Software Effort Estimation Use Case Points Metrics. 5th Malaysian Conference in Software Engineering (MySEC 2011), pp. 257- 262, (2011)
24. M. Ochodek, J. Nawrocki, K. Kwarciak: Simplifying effort estimation based on Use Case Points, Information and Software Technology, Volume 53, Issue 3, pp. 200-213, ISSN 0950-5849, http://dx.doi.org/10.1016/j.infsof.2010.10.005, March (2011)