Linear Prediction Model for Effort in Programming based on User Acceptance and Revised use Case Point Method

Fahad H. Alshammari
College of Computing and Information Technology
Shaqra University
KSA

***Abstract*—As long as most of the processes of verification and validation of software to grant acceptance by the customer/user, are subjective type, it is aimed to design a standard mathematical model with empirical to perform an appointment with areas or stages where development teams most fail involving large-scale software projects. This model will be based on a survey that the user must fill as going testing and validating the software, and which response curve must be linear with respect to the software development process. This paper aims to discuss the aspects surrounding the estimation of mathematical model in the validation and acceptance by a user through the revised Use Case Point Method. First, an assessment of the most recent techniques of application of the method are done, and then a simulation of the process of acceptance and validation by a standard user (Beta Test) will be taken as a practical example. For purposes of this paper, revised use case point method (Re-UCP) must have a specific weight, based on the prerequisites for the development of large-scale software. Once obtained this weighting, the user shall assess the finished product and then an approximation function will be to determine the coefficients of the final model approach, and indicating that is the efficient trend of the development team.**

***Keywords*—Function; point; software; engineering; mathematical; model; large-scale; programming; acceptance; validation**

I. Introduction

The evaluation of design and usability, centered on user, has become a common practice in many organizations, however, in most software development companies is still in its infancy and is not used as often. Development cycles of typical software engineering do not use these practices because they may represent additional costs, especially those related to governmental or educational institutions [1]. Currently, usability, understood as an effective means for obtaining acceptability and validation by the user, is an area that takes strength worldwide in software engineering. Countries that are traditionally powers in software development are concerned more by the satisfaction and comfort that produce their products on their customers, taking it as their top priority, while one hand on the performance of the product. Starting with software for personal computers, cellular and even system for cars, they have adapted better to the type of people who require, avoiding with this, that people take too long to use and optimally understand the software of the equipment [2]. Lately, there have been significant changes in the computing revolution; changes covering all aspects of its main function: 'To serve mankind' changes both quantitative in nature, as some more fundamental emerging diversity located in the global context. It is known, moreover, that computers are included in a wide range of aspects of our daily life, to the point that directly influence our lifestyle. The human-computer relationship is intensifying on a global scale that result in even governmental, cultural and / or social tensions, issues that fall outside the scope of this study; however, the importance of thoroughly analyze these characteristics lies in the fact that this relationship (human-computer) is the spearhead to generate a vast multidisciplinary field, if willing, that is just beginning and whose growth is exponential [3]. In conceptual terms of human-computer relationship, the validation and acceptance of usability refers to the process with which the interaction is designed with a computer program. The term is also often used in the context of products like consumer electronics or in areas of communication. It can also refer to the efficient design of mechanical objects such as, for example, a handle or a hammer. As rules adopted worldwide and given the growing importance of ensuring the proper functioning of computer systems, emerged the need to establish parameters and standards governing the acceptance and usability of computer systems. This paper is based on the recommendations of the ISO/IEC 25010 standard which establishes regulations about quality requirements and evaluation of large-scale software development. It is well known that the acceptance and validation of a software depends purely on human behaviour and preferences, based, of course, in their ability to interact with the technology that is being presented, this is why in this paper is established a mathematical model with linear trend and empirical basis, as a reference between the work of a development team, estimated effort of development by the Rev-UCP method and preferences of a user. Here, will be used the experiences of software development of operational management of a private hospital, which is considered large-scale and small modules will break down in this way to be able to use this proposed model. As a case study, this article will discuss the acceptance and validation of an operational management software in a private hospital, as it can give a tangible perspective of the advantages or disadvantages of the software to analyze the acceptance by the user.

II. Theoretical Basis

A. Requirements Engineering, Assuring the Product Quality

Properly Application of Requirements Engineering increases the chances of producing software that meets the needs of users, many errors in the requirements stage are rooted in the ambiguity presented between end users and developers.

On his book, Pohl [4] suggest that the ’vision’ defines the change the reality of any system. In other words, a vision states the goal of making any change on any system. When a client have a vision we have to mind on the goal to change the reality of that client. However, Pohl also talks that the vision must be supplemented with the context of the system, in fact, they are taken as the two main inputs of the engineering requirement.

Much of the problems that arise during the process of software development, due to the lack of a proper process of definition and understanding of the requirements and the problem to solve, and the unclear interpretation of customer needs. That is why requirements management, in software engineering, is one of the main strategies to ensure the quality of applications from the earliest stages of software development [5].

B. Understanding the Software Complexity and Its Relation with Acceptance

The complexity of the software is, by tradition, a linearly direct indicator of software quality and, especially, the cost. While the complexity is greater, so the cost will be. In recent years they have invested large amounts of money and effort in the development of techniques and metrics to "measure" the complexity of software modules all dimensions. Obviously, many of these measures are correlated with each other. Understanding these relationships is important metrics to assess themselves and ultimately reducing software development efforts and maintenance [6].

Jay et al. [6] found a statistical method to establish the linearity between the lines of program code and McCabe cyclomatic complexity of using empirical inferences and refuting earlier studies had conflicting results. It also suggests that there is some instability in the predictions based on empirical collinear factors, in any case, dependent on language and the complexity inherent in it. In principle, to establish a relationship between the complexity of software and acceptance by the customer is quite difficult as acceptance depends solely on human behavior which can not be modeled linearly as suggested by this study, is why it is done necessary to have statistical tools with the same type and customer/user feedback to find satisfactory results of our interest.

Following the scheme proposed by Bentley [7] in which the software should follow three basic stages:

Verification: where it is confirmed that the software meets all technical specifications.

Validation: that software should meet all business requirements.

Find Defects: Any variation between the output of software and expected.

The true value of software testing go beyond pure test the code. It also examines the behavior of the software from the premise that the code is not necessarily bad if the behavior is too [7].

Meanwhile, Cristia [8] raises two questions regarding verification and validation.

Verification: Are we building the right product?

Validation: Are we building the product correctly?

In this sense, verification is an activity carried out by engineers having at hand a model of the program, while validation is carried by the user and must make taking into account what is expected by the program. Cristia, at his work proposes the existence of various techniques for validation and verification, ranging from the most informal and empirical to the formal involving calculation refinement, etc. [8].

Jones in his work of 2012 [9], gives an economic to the third stage of the previously proposed scheme approach. He mentions that the industry spends about 50% of the cost of development, finding and fixing software defects. It indicates, moreover, that a synergistic combination of defect prevention, removal of defects in prototyping and formal test can dramatically reduce costs by more than 50% compared with the results of 2012 [9].

C. Objects Oriented Programming and use Case Points

As Glasser [10] suggests, object-oriented programming makes programs organized as a collection of interactive objects with their own data and functions. One of the advantages of this paradigm is that objects can be reusable and configurable. Separating concerns and focusing on each object separately makes oriented objects very attractive, especially for large-scale software programming.

This facilitates, in the best, identification and classification of the use cases.

Wirfs [11] on her presentation describes the action to determine the use cases as a full script, and makes it an art, calling it ’The art of writing use cases’. There she mentions, step by step, philosophy of establishing a use case ranging from understanding the case models, including actors, diagrams and glossaries, to a detailed and accurate description of the prototype to develop. There is highlighted the fact that each use case consists of a reference and a different perspective that involves, of course, the actors considered in the step. The mention of the requirements is a ‘point of honor’ in her presentation because it is repeatedly diagram as an essential basis for all work of lifting use cases.

On his book, *Software Engineering,* Marsic [12] indicates that projects with many complicated requirements take more effort to design and implement than projects with few simple requirements. In addition, the effort depends specially on what tools the developers employ and how skilled the developers are. The factors that determine the time to complete a project include:

- Functional requirements: The complexity of use cases, in turn, depends on the number and complexity of the actors and the number of steps (transactions) to execute each use case.

- Nonfunctional requirements: These describe the system nonfunctional properties, known as FURPS+, such as security, usability, and performance. These are also known as the technical complexity factors.
- Environmental factors: Various factors such as the experience and knowledge of the development team, and how sophisticated tools they will be using for the development.

An estimation method that took into account the above factors early in a project life cycle, and produced a reasonable accurate estimate, say within 20% of the actual completion time, would be very helpful for project scheduling, cost, and resource allocation [12].

However, Jones said that more than 80% of software applications are not new because they were developed in past. Because of this, most applications today are replacements for older and obsolete applications. Because these applications are obsolete and also in spite of the lack of information documents, the older applications contain hundreds or thousands of business rules and algorithms that need to be transferred to the new application. This is a different paradigm in the development of the list of requirements for large-scale software [13].

Jones also refers to the vital importance of the intervention of software engineer in raising the requirements for the application, because it is a serious mistake to think that the user, who is not a software engineer, is able to express, optimally, 100% of these requirements, and this lies in the fact that precisely these requirements represent the state of the art of engineering applicable to software. He mentioned, in any case, that one of the ways in which we can base this symbiotic relationship is data mining for business rules and appropriate algorithms. And while this happens, data mining is also used for sizing through function points and lines of code [13].

D. *Revised use Case Point Method as Extension of Function Point Method*

In addition to his work of 2012, Jones emphasizes, among other things, that there are two very useful metrics to show both the economic value and the quality of software. These metrics are:

- Function points for normalization of results.
- Defect removal efficiency [14].

In the work of Manzoor et.al [15], is indicated that the UCP method is originated, in principle, from the method of Function Point except that the UCP makes an analysis of requirements in the object-oriented process. It begins with the system functionality measurement based on the Use Case Model on a count called Unadjusted Use Case Point (UUCP). The technical factors in which UCP is based are equal to those of function points. The UCP estimates the total size of the system that leads to the goals of acceptability and user validation [15]. In other work, Mazoor et.al, suggests that the validation process involving Re-UCP should be carried out for different large scale software projects in order to increase and perform a better acceptability. Future research should be conducted to enhance the benefits of Re-UCP for large-scale software projects through vertical and horizontals [16].

- Transactions of Re-UCP:

The calculation process involved in UCP need case diagrams and descriptions. To understand the logic of UCP utilization, it must be known that there are several steps for implementing a use case. These steps are so called 'transactions' [17].

- Steps for an effective Re-UCP:

Step 1: Classification of Actors trough calculation of its weights UAW (Unadjusted Actor Weight) [18]. Table 1 referes to classification of actors:

TABLE I. CLASSIFICATION OF ACTOR

Actor Category	Description	Actor Weight
Simple	Actor use API's	1
Medium	Actor use Protocol	2
Complex	Actor use GUI's	3

UAW has an equation:

$$UAW = \sum_{i=1}^{n} AW_i \quad (1)$$

Where:

n= Number of Actor.

AW= Weight of each Actor Category (Table 1).

Step 2: Classification of Unadjusted Use Case Weight (UUCW) through its calculation. Table 2 represents the number of transactions in a use case.

TABLE II. CLASSIFICATION OF USE CASE

Use Case Category	Description	Use Case Weight
Simple	A use case has 3 or less transactions	5
Medium	A use case has 3 to 7 transactions	10
Complex	A use case has more than 7 transactions	15

From this classification, the study can synthesize the equation that allows the study to calculate the UUCW:

$$UUCW = \sum_{i=1}^{n} UCW_i \quad (2)$$

Where:

n= Number of Use Case.

UCW= Weight of each Use Case Category (Table 2).

Step 3: Calculating Unadjusted Use Case Point (UUCP).

$$UUCP = UUCW + UAW \quad (3)$$

Step 4: Calculating Technical Complexity Factor (TCF). TCF is involved with the software size, considering the technical aspects of the system. This is ranged from 0 (nonrelevant) to 5 (important factor) [18]. Table 3 summerizes the technical factor weight.

TABLE III. TECHNICAL FACTOR WEIGHT

Ti	Technical Factor	Weight
T1	Required Distributed Systems	2
T2	Response Time Is Important	1
T3	End User Efficiency	1
T4	Required Complex Internal Processing	1
T5	Reusable code to Focus	1
T6	Installation Easy	0.5
T7	Usability	0.5
T8	Cross-Platform Support	2
T9	Easy To Change	1
T10	Highly Concurrent	1
T11	Custom Security	1
T12	Dependence On Third-Part Code	1
T13	User Training	1

TF is obtained as the sum of multiplying score and weight nad the following is the equation [18]

$$TF = \sum_{1}^{13} Score_i * Weight_i \quad (4)$$

And TCF is obtained using TF

$$TCF = 0.6 + (0.01) * TF \quad (5)$$

Step 5: Environmental Complexity Factor. It is determined through a score of between 0 (no experience) to 5 (expert) for each of the 8 environmental factors [17], as referred in Table 4.

TABLE IV. ENVIRONMENTAL FACTOR WEIGHT

Ei	Environmental Factor	Weight
E1	Knowledge of the Project	1.5
E2	Application Experience	0.5
E3	OO Programming Experience	1
E4	Lead Analyst Capability	0.5
E5	Motivation	1
E6	Stable Requirements	2
E7	Part Time Staff	-1
E8	Difficulty Programming Language	-1

First, the study should calculate the prevous EF (Environmental Factor) and then calculate the ECF.

$$EF = \sum_{1}^{8} Score_i * Weight_i \quad (6)$$

and ECF:

$$ECF = 1.4 + (-0.03 * EF) \quad (7)$$

Step 6: UCP (Use Case Point) [17].

$$UCP = UUCP * TCF * ECF \quad (8)$$

Step 7: Calculating the effort. For the purpose of this investigation, the latter factor Effort is used to determine how much time/hours/staff was invested in the development of the application and thereby effectively determine quantitatively if the expectations and requirements of users are met and if the development team is working efficiently. The value of effort is obtained by multiplying the value of UCP and the constant ER in staff hours/UCP. Sholiq suggest that a value of ER equal to 20 staff hours/UCP can be used. Following this proposal, for small and medium-scale business applications the ER can be 8.2 or 4.4 in the case of development of websites using a template or component [17].

$$Effort = UCP * ER \quad (9)$$

E. Linear Model for Mathematical Characterization, using Least Square Model

When a linear pattern is formed from a graph of scattered data, the relationship between the two variables is often modeled by a straight line [20].

The Statistical Models traditionally are used to predict the response of a dependent variable on the observed values of the independent variables. The independent variables are known better as predictor variables. Using linear models as predictor for phenomena can be very straightforward [19].

Because of the relation between the score and production effort of software, obtained via the method of Re-UCP, and the score given by the user, upon completion of the software, is linear where the desired value is a slope equal 1, was chosen for purposes of this research a model scheme with quadratic approximation or least squares.

Van der Geer in 2005 associated with the statistical approach, behavioral science through the use of least squares. The method of least squares is about estimating parameters by Minimizing the squared discrepancies Between Observed data, on the one hand, and Their expected values on the other [21].

Schmidt, in his 2005 project [22] proposes a parameter estimation based on linear regression of least squares with an L1 penalty in the regression coefficients. Indicating the special interest in this issue given the appeal that may be able to create fairly accurate prediction models with the simplicity of a well-known mathematical methodology.

The main work of Schmidt, beyond focusing directly on the properties of the model was the assessment of a variety of previous approaches to the estimation of these parameters.

- The Regression Problem

The most frequent use of LS was linear regression, which corresponds to the problem of finding a line (or curve) that best fits a set of data points. In the standard formulation, a set of N pairs of observations (Yi,Xi) is used to find a function relating the value of the dependent variable (Y) to the values of an independent variable (X)[23].

The prediction is given by:

$$\hat{Y} = a + bX \quad (10)$$

Where:

a: Intercept with Y axis.

b: the slope of the function.

The least square method involve the estimate of these last parameters as the values which minimize the sum of the squares between the real measurements and the theoretical model [23].

The minimizing expression is:

$$\varepsilon = \sum_i (Y_i - \hat{Y}_i)^2 = \sum_i [Y_i - (a + bX_i)]^2 \qquad (11)$$

Where:

ε : Error to be minimized

Using the property that derivating a quadratic expression the study can achieve its minimum value. Calculating the derivative of ε with respect to **a** and **b** and making them to zero, gives the following set of equations:

Derivative respect to **a**

$$\frac{\partial \varepsilon}{\partial a} = 2Na + 2b\sum X_i - 2\sum Y_i = 0 \qquad (12)$$

Derivative respect to **b**

$$\frac{\partial \varepsilon}{\partial b} = 2b\sum X_i^2 + 2a\sum X_i - 2\sum Y_i X_i = 0 \qquad (13)$$

Solving these equations results the following least square estimates of a and b as:

$$a = M_Y - bM_X \qquad (14)$$

Where:

M_Y : Mean of Y

M_X : Mean of X

And:

$$b = \frac{\sum (Y_i - M_Y)(X_i - M_X)}{\sum (X_i - M_X)^2} \qquad (15)$$

[24].

F. Method for Linear Adjustment using Average Line

Based on the well known Line Equation [25]

$$y = mx + b \qquad (16)$$

The study would have two lines with m_1, m_2, b_1 and b_2, these are the parameters which define the straight average, if the study does the semi-sum of the coefficients m_1, m_2, b_1 and b_2, the study obtains m and b of the average line. While semi-difference give us the range of uncertainty, Δm and Δb.

So, the semi-sum:

$$m = (m_1 + m_2)/2; b = (b_1 + b_2)/2 \qquad (17)$$

and the semi-difference:

$$\Delta m = (m_1 - m_2)/2; \Delta b = (b_1 - b_2)/2 \qquad (18)$$

Considering that $m_1 > m_2$ Therefore the best set of lines that inform us within what range the study expects to drop a new measure is given by the expression:

$$y = (m \pm \Delta m)x + (b \pm \Delta b) \qquad (19)$$

To graph the lines of maximum and minimum slope should mark the centroid, in other words, the P($\bar{x}, \bar{y}$) point that emerges from the average of the coordinates from the data:

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}; \bar{y} = \frac{\sum_{i=1}^{N} y_i}{N} \qquad (20)$$

Where:

N is the number of data.

(x_i, y_i) are the experimental data.

Once located the centroid, draw the line with maximum and minimum slope passing through this point P($\bar{x}, \bar{y}$) [26].

G. Human Behaviour in Software Technology

Today the changes generated by technological advances, affect the behavior and actions of the individual, leading to approach new rules or disciplines to address and provide answers to the problems generated by the Information and Communication Technologies.

Kusumari et al. [28] said that capability of using software development and collaboration tools would increase the quality of resulting software and, this way, may incide in acceptance and validation. Some calculated tools and/or models would make the development phase easier to do.

Humans are an integral part of a more complex systems. If the study wants to describe a system of this type with good accuracy, it is necessary to model the human components with the same precision as the technical components. Human behavior is structurally very complex. As human behavior is influenced by physical, emotional, cognitive and social factors, it is very intricate [29].

Ghezzi et al. [30] in their work of 2014 emphasizes the importance of knowing and predicting the different behaviors of users for successful software application, which, the fact, dismiss these factors can lead almost always failures of type techniques and even non-technical that in the end entail significant loss of economic order. However, it takes into account when the number of users grows as it is clear that the behavior will vary greatly; in any case, a population of users of the same application can be handled uniformly, in both, the respective corrective training and knowledge in the application itself is true.

Part of these corrections should be made in the software development stage involving users in the same [30].

III. Calculations Development

To start the calculations, the paper proposes the following block diagram which will guide the reader step by step analysis of the study as shown in Fig. 1:

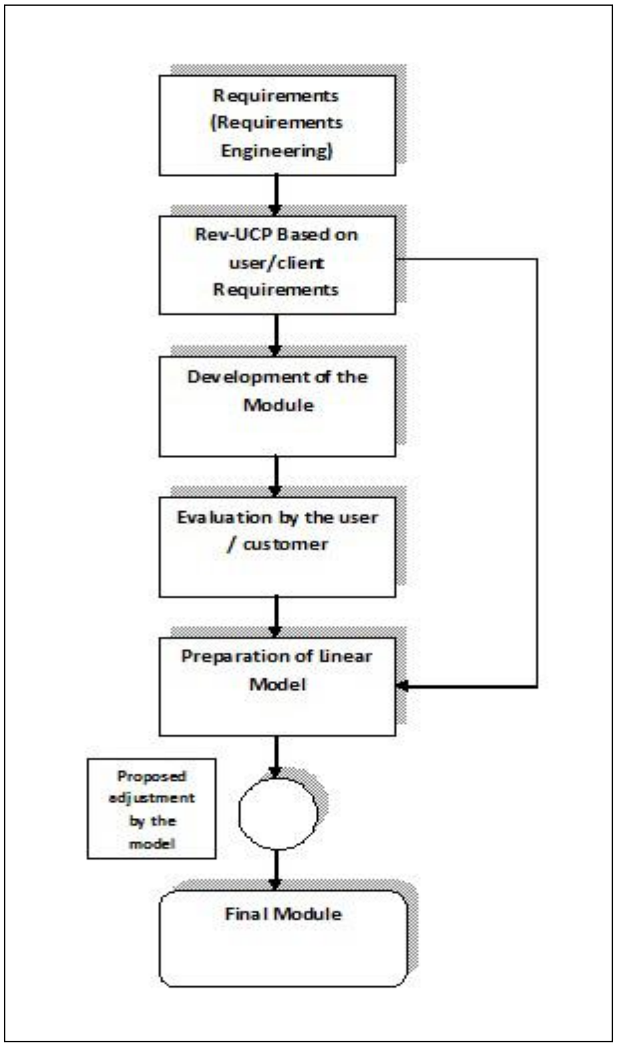
Requirements (Requirements Engineering)
Rev-UCP Based on user/client Requirements
Development of the Module
Evaluation by the user / customer
Preparation of linear Model
Proposed adjustment by the model
Final Module

Fig. 1. Block diagram of the steps in the development of study.

This algorithm is repeated with each module or use case, and would be adjusted as software development advances.

This linear algorithm was chosen because its scalability is better to those with feedback, ie, the response times are much better and more tailored to customer needs.

In this study, it has three possible scenarios:

- Existing conditions given by labor and programming resources.
- Ideal conditions given by the Rev-UCP and requirements analysis
- The final conditions of the product are given by acceptance testing by the customer / user.

Being the analysis of software programming of large scale, it was decided to segment the application in modules which will show to the user for evaluation through a form which will have a weighting on each stage of software development. Evaluating the results provided by the user on the results obtained by the programming team can establish a linear relationship which comes from the method of average line adjustment.

H. Calculating Rev-UCP for each use Case as Ideal Conditions

To start the investigation, the study takes as reference various methodologies which were tested with different scenarios associated with the same large-scale software development, which is a system for operational management of a clinical laboratory based in the cloud. Was chosen the method of Rev-UCP which throws the study with great accuracy which is the effort required to develop such software, the first step was to identify, from customer requirements, what are the use cases on each module as an integral part of the system. The following format was used to identify and classify each use case, as assigned in Table 5:

TABLE V. Use Case Format

	Catalog Information			
	Project	**Operative System for Private Hospitals**		
	Author	Fahad		
Version	0.1	Status	Development	
	Use Case Definition			
	Code	**Use Case 01**		
	Title	**Enter patient data**		
	Objective	Enter name, identity card number, date of birth, address, height, weight, medical history, photography.		
	Description	Entering via the keyboard, the data mentioned above.		
	Actors	Client/user		
	Prior Conditions	Client/user must be authorized for this action, database is able to accept this data.		
	Main Scenario	(A) The user opens the patients form. (B) Entering patient details. (C) Check that everything is correct before accepting. (D) Accept the entered data. (E)The system checks if exists previous data related to the identity card. (F) The system drops a message with satisfactory transaction.		
	Alternative Scenario	(A) The system tells the user that the identity card already exists and gives the possibility to modify any data if required. (B) The user modifies some data and accepts. (C) The user closes the form.		
	Exception Scenario	(A) The system tells the user that the identity card already exists and gives the possibility to modify any data if required. (B) The user delete all the information. (C) The user closes the form.		
	Success Condition	All data entered is saved and organized successfully and displayed through a flat pdf format.		
	Hypotheses	When you delete data from a patient, what happens with the clinical history done in the laboratory?		

Using this format, and meeting customer requirements, there were 85 cases of use identified and listed below with their respective calculations based on the Rev-UCP method, note the study decided to break down each use case and apply the methodology for a more accurate perspective of the curve of effort required by the software to develop, as assigned in Table 6:

- Use Case 1: Enter Patient Data

TABLE VI. USE CASE 1: ENTER PATIENT DATA

Enter Patient Data	
UAW	3
UUCW	5
UUCP	8
TF	37
TCF	0.97
EF	29.5
ECF	0.515
UCP	3.99
Effort	79.8

The above calculation yields the following segmentation in terms of timely dedication of the use case, as assigned in Table 7:

TABLE VII. DISTRIBUTION IN PROJECT STAGES OF USE CASE POINTS

Enter Patient Data		
Stage	**Percentage**	**H/M**
Analysis	10	7.98
Design	15	11.97
Programming	40	31.92
Tests (Functionality)	5	3.99
Tests (Errors)	5	3.99
Tests (Efficiency based in exec time)	5	3.99
Benchmarking (PC's Resources)	5	3.99
Tests (Database)	5	3.99
Tests (Overload and Tuning)	5	3.99
Tests (Running)	5	3.99
Total	100	79.8

I. *Existing Conditions of Labour Resources and Delivery Time*

The workforce of the group under study in this paper consists of one project leader, one quality expert, two programmers analysts, one GUI designer and two senior programmers. Clearly each module, section or segment of the software has its own characteristics and the team to develop them may vary over time. However organizational behavior can be modeled linearly under the requirements of effort given by the Rev-UCP method. The following Table 8 shows the time available for the project for each position:

TABLE VIII. WORK TEAM

Code	**Quantity**	**Description**	**Years of Experience**	**Hours/ Day**
PL1	1	Project Leader	15	2
QE1	1	Quality Expert	12	1
PA1	2	Programmer Analysts	3	4
GD1	1	GUI Designer	8	3
SP1	2	Senior Programmer	10	4

Using the following formula the study can obtain the total hours/man available for the project:

$$HM = \sum (Code_{Quantity} * HD) \tag{21}$$

Where:

HM= Hours/man a day.

Code: Type of staff.

Quantity: The number of people of a type available for the project.

HD: Hours a day.

For example, for a time span of 40 days for delivery of the product it has a total of:

$$TotalHM = 40 * 22 = 880\, Hours/man \tag{22}$$

The total work of the development team for Use Case 1 is reflected in the following Table 9:

TABLE IX. DISTRIBUTION IN PROJECT STAGES OF REAL PROGRAMMING PROGRESS

Enter Patient Data		
Stage	**Time per Stage**	$\sum$
Analysis	6	6
Design	9	15
Programming	55	70
Tests (Functionality)	5	75
Tests (Errors)	2	77
Tests (Efficiency based in exec time)	4	81
Benchmarking (PC's Resources)	8	89
Tests (Database)	6	95
Tests (Overload and Tuning)	12	107
Tests (Running)	1	108

Clearly, if the team requires more time than stipulated by the Rev-UCP method may impact on the real costs of software and should optimize this feature through a linear regression for a series of standardized points on a plot where the horizontal axis are values estimated by the Rev-UCP method and the vertical axis are the actual values provided by the development team.

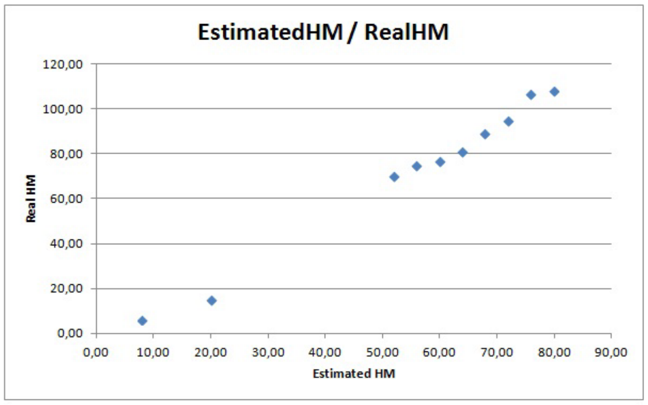

Fig. 2. Estimated HM Vs. Real HM in scattered form.

Because the data are scattered as shown in Fig. 2, must do a linear quadratic regression approach explained in the theoretical basis for finding the best line that fits the data. If properly apply linear regression equations, then yields the following result:

$$F_{real}(x) = 1.474x - 9.472 \quad (23)$$

And the plot is shown in Fig. 3:

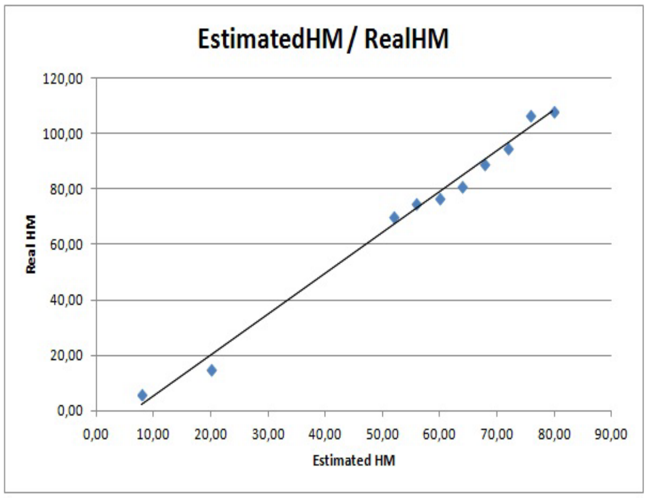

Fig. 3. Estimated HM Vs. Real HM With linear approximation.

J. *Conditions of Evaluation and Acceptance by the Customer/User*

While it is true that the calculation of the estimated effort through the Rev-UCP method is a fairly accurate approximation, the study finds that user or client preferences differ somewhat from those estimates, even may differ medium or largely from real effort applied to the development of the module to be evaluated. In the case study of this research, the study find this feature because it is obvious that is very difficult to model accurately and precisely the behavior, tastes and human preferences.

In this section, Fogg says that can be used to design technological channels influence the behavior of a user over the use of software, however, people do not understand what factors lead to change behavior and that's why some persuasive design fails [27].

In order to rate the acceptance and validation of client/user, the study used a table that automatically weighs each stage of the process, one by one, and thus can establish a linear relationship to the effort estimated by the Rev-UCP method.

As was discussed above in the introduction, to this applies an alpha test type by the customer in development site. The user naturally observing and recording errors and problems of use. This test was conducted in a controlled environment, as summarized in Table 10.

TABLE X. SURVEY THAT THE USER MUST FILL OUT WHEN TO ALPHA TEST

Description	**Value Given By User**
Is the prototype performing the agreed and expected functions correctly?	85
Is the prototype achieving specific goals?	93
Does the prototype has the appropriate set of functions for specified tasks?	95
Does the system showing actual transaction?	98
Does the user can interrupt an operation without affecting the normal operation of the prototype?	75
In case an error occurs, the prototype is still functioning normally?	85
Is the prototype able to return to a stable state after an error occurred?	65
Do users perform their tasks properly in the shortest possible time?	90
Is it appropriate the size of the text?	80
Does the physical space used is appropriate?	85
The amount of information is well distributed?	95
Does the application enables the user to feel comfortable?	100
Is there default values?	88
Do the actions can be performed simply in a few steps?	92
Is there clarity of the elements of the interface?	78
Are the messages properly notifies the action that the user is going to carry out?	66
Are the controls properly selected for each function?	79
Is it easy to recognize quickly and clearly what actions the user can perform on an interface?	91
Are there elements that show the progress of a transaction?	93
Are the controls interfaces, provide help or information from its use?	87
Do the data is displayed complete and easily?	100
Can You easily perform actions on the data?	100
Can you search and access data quickly?	99
Is it editable the content entered by the user?	78
Is the correction of errors in input data allowed?	86
Do actions can be canceled without detrimental effects to normal operation?	86
Is the prototype can be adapted to the needs of different users?	88
Does the design is consistent across all screens of the prototype?	92
Are the controls of the same type maintain the same behavior?	99
Are the controls always kept in the same position of the interface?	100
There are mechanisms for validating input data provided?	94
There are mechanisms that facilitate the user to input data provided?	95
Do error messages represent clearly and concisely the error occurred?	85
Do error messages suggest a solution to the problem occurred?	68
Do help messages are clear and concise?	79
Do background colors used in the elements of the user interfaces are always the same?	78
Can foreground elements (either text or images) easily distinguished background?	85
Are there non-aligned or disorderly elements?	86
Are the sections where the interface is divided, remain uniform throughout the application (prototype)?	87
Are the actions and tasks designed to perform as fast and intuitive as possible?	96

The value given by the user corresponds to the following classification, as demonstrated in Table 11:

TABLE XI. WEIGHTING GIVEN BY THE USER

Not fulfilled	Poorly Fulfilled	Mildly Fulfilled	Fulfilled	Totally Fulfilled
to 20	21 to 40	41 to 60	61 to 80	81 to 100

Each question has a direct impact based on percentage on some stages of development. Tables 12 and 13 below are shown an example of how the weighting is calculated, the study must add that these tables are entirely empirical and developed based on field experience of authors.

TABLE XII. WEIGHTING GIVEN BY THE USER PART 1

Question	Value Given By User	Analysis	Design	Programming
	85	17		34
	93	9,3		37,2
	95	9,5	9,5	19
	98	9,8	19,6	9,8

TABLE XIII. WEIGHTING GIVEN BY THE USER PART 2

Functionality	Errors	Efficiency	PC Resources	Database	Tuning	Running
25.5		8,5				
27.9		18,6				
38					9,5	9,5
29.4	9,8			9,8	9,8	

Once the survey is completed weight / total value of each stage of the process is calculated using the following formula:

$$ValueStage_i = \sum_{j}^{n} W_j \qquad (24)$$

Thus, the forty questions involved in the survey have their weightings in each stage of the process until the following total weight, as summarized in Table 14:

TABLE XIV. TOTAL WEIGHTING FOR STAGE OF THE PROCESS

	Values Obtained	Expected Values	Rate
Analysis	301,9	330	91,48%
Design	617,7	710	87,00%
Programming	425,8	480	88,71%
Functionality	448,7	510	87,98%
Errors	245,2	300	81,73%
Efficiency	236,3	270	87,52%
PC Resources	341,2	400	85,30%
Database	305,7	330	92,64%
Tuning	282,6	320	88,31%
Running	295,9	350	84,54%

Once the relationship between Obtained Values and Expected, proceed to establish a new relationship, now, between user perception and the real effort used for the development team in programming the module, using the following Table 15:

TABLE XV. RATE USER EVAL VS. ESTIMATED HM

	Real	User Perception %	Rate from real
Analysis	6	91,48%	5,49
Design	9	87,00%	7,83
Programming	55	88,71%	48,79
Tests (Functionality)	5	87,98%	4,40
Tests (Errors)	2	81,73%	1,63
Tests (Efficiency based in exec time)	4	87,52%	3,50
Benchmarking (PC's Resources)	8	85,30%	6,82
Tests (Database)	6	92,64%	5,56
Tests (Overload & Tuning)	12	88,31%	10,60
Tests (Running)	1	84,54%	0,85

And then the study proceedees to represent these values, by way of summation in a graph whose horizontal axis (X axis) is the values obtained through the use of Rev-UCP method, as is shown in Fig. 4:

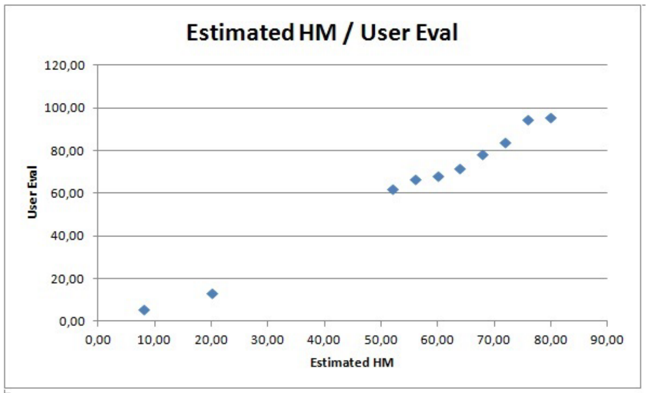

Fig. 4. Estimated HM Vs. User Perception.

And this data optimization depicted in a scattered way is also given by a linear regression based on the method of least squares, and whose equation is:

$$F_{user}(x) = 1.301x - 8.174 \qquad (25)$$

And the plot is shown in Fig. 5:

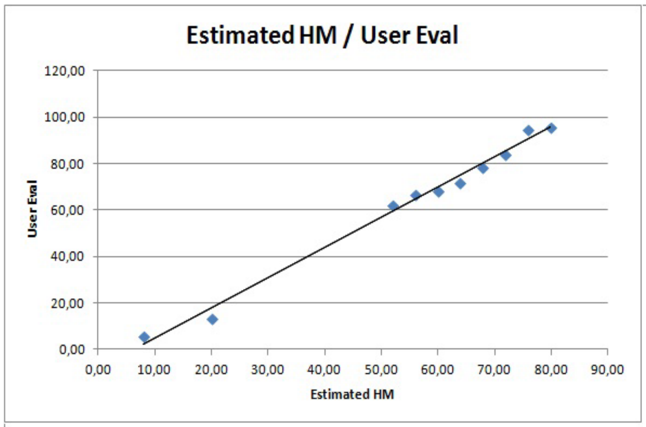

Fig. 5. Estimated HM Vs. User Perception.

K. Prediction of Required Programming Effort using a Linear Mathematical Model

The average linear equation (excluding errors) is as follows:

$$F_{average}(x) = \frac{(1.474 + 1.301)}{2} x + \frac{(-9.472 - 8.174)}{2} \quad (26)$$

However, the study must now consider the error produced by the semidifference:

$$F_{average\Delta}(x) = (1.3875 \pm 0.0865)\, x + (-8.823 \pm 0.649) \quad (27)$$

To graph the lines of maximum and minimum slope need to calculate the centroid using equation (20), Fig. 6 shows it.

$$Cent = (7.5069, 1.5925) \quad (28)$$

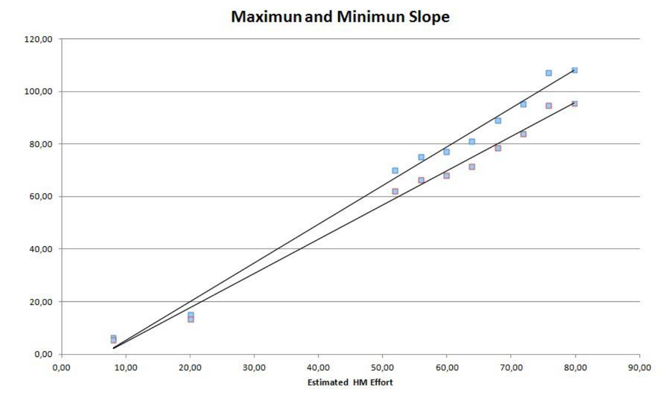

Fig. 6. Maximun and minimun slope.

IV. Discussion of Results, Conclusions and Recommendations

With the results obtained in this study, produced a model of linear approximation to determine the effort required in large-scale software programming, this was done thanks to the modularization and segmentation of all software and as they develop the different modules it can be applied and, in fact, adjusting the linear model very accurately.

The main reasons for the development of this model is to consider the validation and acceptance of the software by the user without impacting significantly on the costs of programming. As is known that, excessive application of hours/man in an activity directly affects the final cost of the software obtaining virtually the same result, thereby decreasing the efficiency of the development team.

While it is extremely difficult to model the tastes and preferences of a user regarding a software, the linear approximation even dependent requirements, fits quite well with the objectives of this study.

However it should be noted that in the development of software, especially large scale, both teams programming and users can vary greatly throughout the life of the project, which implies that adjustments must be made provided when necessary or at least the start of the programming of each module.

A way to future studies might include further aspects such as organizational behavior, psychology client/user and design persuasive way to minimize errors in the calculation of the linear model.

The software studied in this study is still in development stage approximately 70% complete. Below is Table 16 with the percentages of global acceptance by the user of some modules and estimated by the proposed model calculation:

TABLE XVI. Results of Apllied Model in Some Modules

	Inserting Clinical Analysis	**Daily performance report**	**Billing**
Analysis	92,00%	89,00%	97,00%
Design	93,00%	98,00%	94,00%
Programming	89,00%	98,00%	96,00%
Tests (Functionality)	95,00%	99,00%	95,00%
Tests (Errors)	94,00%	96,00%	89,00%
Tests (Efficiency based in exec time)	88,00%	97,00%	89,00%
Benchmarking (PC's Resources)	96,00%	85,00%	90,00%
Tests (Database)	97,00%	98,00%	93,00%
Tests (Overload & Tuning)	92,00%	93,00%	92,00%
Tests (Running)	95,00%	89,00%	95,00%

As its clear that the results are quite satisfactory considering that it is taking into account the same user that was used to calculate the model.

It is also necessary to analyze this result from the pragmatic point of view because for non-productive modules prototypes were used, however, they were basis for developing the final module required by the client.

Acknowledgements

My deepest acknowledgements to the University of Shaqra for the support received during the preparation of this study, and that made possible its successful completion. Thank you for promoting this study, which is part of my growth as a professional and that in turn, such research will contribute to the overall knowledge of software engineering.

References

[1] J. Scholtz, B. Shneiderman, *Introduction to Special Issue on Usability Engineering*, Boston, United States: Kluwer Academic Publishers, 1999.

[2] J. Cabrera and E. Contreras, Usabilidad: Factor importante para hacer atractivos y comprensibles los sitios Web. Caso de estudio: Sitio de la UTM, Oaxaca, Mexico: Universidad TecnolÃ³gica de la Mixteca, 2009.

[3] R. Harper, T. Rodden, Y. Rogers, A. Sellen, Being Human: Human-Computer Interaction in the year 2020, Cambridge, England: Microsoft Research Ltd, 2008.

[4] K. Pohl, Requirements Engineering, Berlin, Germany: Springer-Verlag Heidelberg, 2010.

[5] G. Espinoza, Metodo de validacion de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones RIA, Barquisimeto, Venezuela: Universidad Centro-Occidental Lisandro Alvarado, 2012.

[6] G. Jay, J. Hale, R. Smith, D. Hale, N. Kraft, C. Ward , Cyclomatic Complexity and Lines of Code: Empirical Evidence of a Stable Linear Relationship, Tuscaloosa, United States: Software Engineering & Applications, 2009.

[7] J. Bentley, Software Testing Fundamentals-Concepts, Roles, and Terminology, Charlotte, United States: Wachovia Bank, 2005.

[8] M. Cristia, Introduccion al Testing de Software, Rosario, Argentina: Universidad Nacional de Rosario, 2009.

[9] C. Jones, Software defect origins and removal methods, Sydney, Australia: Namcook Analytics LLC, 2012.

[10] M. Glasser, Open Verification Methodology Cookbook, Wilsonville, United States: Mentor Graphics Corporation, 2009.

[11] R Wirfs-Brock, The Art of Writing Use Cases, Wirfs-Brock Associates, 2001.

[12] I. Marsic, Software Engineering, 1em plus 0.5em minus 0.4em New Brunswick, New Jersey: Rutgers University, 2012.

[13] C. Jones, Software Engineering Best Practices, Australia: Mc Graw Hill, 2010.

[14] Capers Jones, Software Quality Metrics:Three Harmful Metrics and Two Helpful Metrics, Australia: Namcook Analytics LLC, 2012.

[15] M. Manzoor K. and A. Wahid, Revised Use Case Point (Re-UCP) Model for Software Effort Estimation, Hyderabad, India: International Journal of Advanced Computer Science and Applications, 2015.

[16] M. Manzoor K. and A. Wahid, Impact of Modification Made in Re-UCP on Software Effort Estimation, Hyderabad, India: International Journal of Advanced Computer Science and Applications, 2015.

[17] Sholiq, A.P. Widodo, T. Sutanto and A.P. Subriadi, A Model to determine cost estimation for software development projects of small and medium scales using Case Points, Surabaya, Indonesia: Journal of Theoretical and Applied Information Technology, 2016.

[18] M. Ochodek, J. Nawrocki, and K. Kwarciak, Simplifying effort estimation based on Use Case Points. Information and Software Technology, Poznan, Poland: Poznan University of Technology, 2010.

[19] O. Burke, Statistical Methods - Linear Models, Oxford, England: University of Oxford, 2013.

[20] A. Arnholt, Least Squares Regression, North Carolina, United States: Appalachian State University, 2008.

[21] S. van der Geer, Least Squares Estimation, Chichester, England: Encyclopedia of Statistics in Behavioral Science, 2005.

[22] M. Schmidt, Least Squares Optimization with L1-Norm Regularization, Vancouver, Canada: University of British Columbia, 2005.

[23] H. Abdi, The Method of Least Squares, Dallas, United States: Encyclopedia of Measurement and Statistics, 2007.

[24] S. Chartier and A. Faulkner, General Linear Models: An Integrated Approach to Statistics Ottawa, Canada: Tutorial in Quantitative Methods for Psychology, University of Ottawa, 2008.

[25] C. Lehmann, Analytic Geometry, New York, United States: John Wiley and Sons, 2005.

[26] G. Molina and M. Rodrigo, Estadistica Descriptiva en Psicologia, Valencia, Spain: University of Valencia, 2010.

[27] B.J. Fogg, A Behavior Model for Persuasive Design, Palo Alto, United States: Stanford University, 2009.

[28] T. Kusumari, K. Surendro, I. Supriana, Human Behavior Conceptual Model in Collaborative Software Development Product Quality, Bandung, Indonesia: ICACSIS, 2013.

[29] B. Schmidt, Human Factors in Complex Systems The Modelling of Human Behaviour, Riga, Latvia: ECMS, 2005.

[30] C. Ghezzi, M. Pezze, M. Sama and G. Tamburrelli, Mining Behavior Models from User-Intensive Web Applications, Hyderabad, India: ICSE 2014,2014.