20th IMEKO TC4 International Symposium and
18th International Workshop on ADC Modelling and Testing
Research on Electric and Electronic Measurement for the Economic Upturn
Benevento, Italy, September 15-17, 2014

Error Propagation in Software Measurement & Estimation

Luca Santillo[1]

[1] *GUFPI-ISMA (Gruppo Utenti Function Point Italia – Italian Software Metrics Association),*
luca.santillo@gmail.com

Abstract – **The real challenge in any activity is to minimize as much as possible the error between an estimate and an actual value, whatever the phenomenon to be evaluated. When dealing with software, the number of proxies can be quite high: the application of an algorithm including one or more independent variables (measures) is finalized to provide one or more output variables (estimates) for a series of measures typically about effort, cost, time, quality or other aspects of the software being developed. Recently ISO proposed also a specific standard on Measurement (ISO/IEC 15939), with a glossary aligned to the Metrology field and to the International Vocabulary of Metrology (VIM).**
Estimation could be seem as a "black art", but error is intrinsic in estimates and must be managed. Thus, regardless of the estimation model (algorithm) being used, practitioners must face the uncertainty aspects of such process: errors in initial measures do affect the derived metrics (or estimated values for indirect variables). Measurement theory does provide an accurate way to evaluate such error propagation for algorithmic derivation of variable values from direct measures, as in the GUM (Guide to the expression of Uncertainty in Measurement).
Although some software estimation models already propose confidence ranges on their results, the formal application of error propagation can yield some surprising results, depending on the mathematical functional form underlying the model being examined. Moving from a previous paper, this one will discuss the propagation of errors in software measurement and shows with some application and examples based on some of the most common software measurement methods and estimation models as Function Point Analysis (FPA) for product sizing issues and COCOMO (Cost Construction Model) for effort and/or duration issues as well as other ones, updating also the discussion to new advancement in the Software Quality field, in particular about product NFRs (Non-Functional Requirements). Few cases and examples will be shown, in order to stimulate a critical analysis for methods and models being examined from a possibly new perspective, with regards to the accuracy they can offer in practice.

Keywords – *Measurement, estimation, accuracy, propagation of uncertainty, error analysis.*

I. INTRODUCTION

In science, the terms uncertainties or errors do not refer formally to mistakes, but rather to those uncertainties that are inherent in all measurements and can never be completely eliminated. A large part of a measurer's effort should be devoted to understanding these uncertainties (error analysis) so that appropriate conclusions can de drawn from variable observations. Measurements can be quite meaningless without any knowledge of their associated errors.

In science and engineering, numbers without accompanying error estimates are suspect and possibly useless. This holds true also in software engineering – for every measurement, one should record the uncertainty in the measured quantity.

Beyond the well-known and easily understood definition of systematic errors and random errors (which is referred to any single quantity being measured, and is no further examined hereby), we point our attention to the consequences of using measures with unavoidable errors within the application of some formula (algorithm) to derive further quantities, as is the case of total size from components' size for a software system, or of effort, time or cost estimation from the total size of the system being examined. In science, this analysis is usually denoted as error propagation (or propagation of uncertainty).

This paper highlights the general phenomenon, for which it always holds true that generic uncertainties from two or more deriving quantities always sum up to provide a larger uncertainty on the derived quantity. This also results in a formal approach to risk analysis, in terms of what-if scenarios where input quantities change with respect to their initial, planned, or desired values.

II. ERROR PROPAGATION THEORY – A SUMMARY

When the quantity to be determined is derived from several measured quantities, and *if* the measured quantities are *independent* of each other (that is, contributions are *not correlated*), the uncertainty "δ" of each contribution may be considered separately [1]. For example, suppose that we have measured the quantities

time $t \pm \delta t$ and height $h \pm \delta h$ for a falling mass (the δ's refer to the relatively small uncertainties in t and h). We have determined that

$$h = 5.32 \pm 0.02 \text{ cm}$$
$$t = 0.103 \pm 0.001 \text{ s}$$

From physics, we can find the acceleration g from the relation:

$$g = g(h,t) = 2h/t^2$$

which yields $g = 10.029\ldots$ m/s^2. (the well-known value for g from physics is actually g = 9.80665... m/s^2). To find the uncertainty in the derived value of g caused by the uncertainties in h and t, we consider separately the contribution due to the uncertainty in h and the contribution due to the uncertainty in t, combined in quadrature:

$$\delta_g = \sqrt{\delta_{g_t}^2 + \delta_{g_h}^2}$$

where for instance we denote the contribution due to the uncertainty in t by the symbol δ_{g_t} (read as "delta-g-t" or "uncertainty in g due to t"). The basis of the quadrature addition is an assumption that the measured quantities have a normal, or Gaussian, distribution about their mean values.

A typical method by which one may calculate the contributions of direct measured independent variables in the dependent variable being derived is the *derivative method*. The basic idea is to determine by how much the derived quantity would change if any of the independent variables were changed by its uncertainty.

Let the function $f = f(x_1, \ldots, x_N)$ be the formula used to derive the dependent variable f from the N independent variables $x_i, \ldots, x_N$, and let δx_i be the estimated error on the value measured for each variable x_i. That is, the true value of x_i belongs to the interval $(x_i \pm \delta x_i)$ for each i from 1 to N. Then the true value of f belongs to the interval $[f(x_1, \ldots, x_N) \pm \delta f]$, where δf is given from the derivatives formula:

$$\delta f = \sqrt{\sum_{i=1}^{N} \left(\left| \frac{\partial f}{\partial x_i} \right| \delta x_i \right)^2} = \sqrt{\left(\left| \frac{\partial f}{\partial x_1} \right| \delta x_1 \right)^2 + \ldots + \left(\left| \frac{\partial f}{\partial x_N} \right| \delta x_N \right)^2}$$

The derivative $(\partial f/\partial x_i)$ is a partial derivative (simply put, when taking a partial derivative with respect to one variable, treat any other variable as constant). Since each uncertain variable will increase, not decrease the final uncertainty, we put the uncertainty in f due to the uncertainty in x_i as the absolute value. In the cited example for acceleration $g = g(h,t)$ we have:

$$\delta_{g_t} = \left| \partial g / \partial t \right| \delta_t = \left| -4h/t^3 \right| \delta_t$$
$$\delta_{g_h} = \left| \partial g / \partial h \right| \delta_h = \left| 2/t^2 \right| \delta_h$$

so that:

$$\delta_g = \sqrt{\delta_{g_t}^2 + \delta_{g_h}^2} = \sqrt{\left(\frac{4h}{t^3} \delta_t \right)^2 + \left(\frac{2}{t^2} \delta_h \right)^2} = 19.8 \text{ cm/s}^2 \approx 0.2 \text{ m/s}^2$$

For simple formulas, as addiction, subtraction, multiplication and division of two variables, one can easily find that the uncertainty is calculated as in the following.

If $f = f(x,y) = x + y$, then $\delta_f = \sqrt{\delta_x^2 + \delta_y^2}$.

If $f = f(x,y) = x - y$, then also $\delta_f = \sqrt{\delta_x^2 + \delta_y^2}$ (uncertainties always add).

If $f = f(x,y) = x \cdot y$, then $\delta_f = \sqrt{y^2 \delta_x^2 + x^2 \delta_y^2}$.

If $f = f(x,y) = x / y$, then $\delta_f = \sqrt{\left(\frac{x}{y^2} \right)^2 \delta_y^2 + \left(\frac{1}{y} \right)^2 \delta_x^2}$.

Notice how the variables values are mixed together in the uncertainty calculation for multiplication and division. It's also worth pointing out that *fractional* or *percentage uncertainties* in multiplication and division behave much like *absolute uncertainties* in addition. In other words, $f = f(x,y) = x \cdot y$:

$$\frac{\delta_f}{f} = \frac{\sqrt{y^2 \delta_x^2 + x^2 \delta_y^2}}{xy} = \sqrt{\left(\frac{\delta_x}{x} \right)^2 + \left(\frac{\delta_y}{y} \right)^2}$$

with the same result holding if $f = f(x,y) = x / y$.

If either x or y is a *constant* or has a relatively small fractional uncertainty, then it can be ignored and the total uncertainty is just due to the remaining term.

Furthermore, if one of the measured quantities is raised to a *power*, the fractional or percentage uncertainty due to that quantity is merely *multiplied by that power* before adding the result in quadrature. For the original example:

$$\frac{\delta_g}{g} = \frac{\sqrt{\left(4h\delta_t/t^3 \right)^2 + \left(2\delta_h/t^2 \right)^2}}{2h/t^2} = \sqrt{\left(\frac{2\delta_t}{t} \right)^2 + \left(\frac{\delta_y}{y} \right)^2}$$

For the values in our example, $\delta_h/h = 0.5\%$ and $\delta_t/t = 1\%$ (so 2 $\delta_t/t = 2\%$), so we can see that the contribution from the uncertainty in h is negligible compared to the contribution from t. We can therefore conclude (as confirmed by previous calculation) that the fractional uncertainty in our measured result for g is about 2%:

$$g = 10.0 \pm 0.2 \text{ m/s}^2$$

As a reference, table 1 shows the uncertainty of simple functions, resulting from independent variables A, B, C, with uncertainties ΔA, ΔB, ΔC, and a precisely-known constant c.

Table 1. Example formulas.

Function	Function uncertainty
X = A ± B	$(\Delta X)^2 = (\Delta A)^2 + (\Delta B)^2$
X = cA	$\Delta X = c\,\Delta A$
X = c(A×B) or X = c(A/B)	$(\Delta X/X)^2 = (\Delta A/A)^2 + (\Delta B/B)^2$
X = cAn	$\Delta X/X = |n|\,(\Delta A/A)$
X = ln (cA)	$\Delta X = \Delta A/A$
X = exp(A)	$\Delta X/X = \Delta A$

III. APPLICATIONS TO SOFTWARE MEASUREMENT – A SIMPLE EXAMPLE

Generally, when measuring the functional size the measurer has to:

- identify (classify & count) the base functional components;
- assign a size value to each component;
- combine the information above.

For instance, in the IFPUG method [2] one has to identify logical files and elementary processes. Logical files are classified as "internal" (ILF) or "external" (EIF) with respect to the boundary of the software being measured, with different size assignments on a low – average – high complexity scale: 7/10/15 and 5/7/10 unadjusted function points, respectively. Elementary processes are classified as "input" (EI), "output" (EO), and "enquiry" (EQ) with respect to the processing logic being performed, with different size assignments on a low/average/high complexity scale: 3/4/6 for inputs and enquiries, 4/5/7 for outputs.

The (unadjusted) functional size formula for IFPUG could be expressed as:

$SIZE_{(UFP)} = 7 \times N_{ILF\text{-}LOW} + 10 \times N_{ILF\text{-}AVERAGE} + 15 \times N_{ILF\text{-}HIGH} +$
$5 \times N_{EIF\text{-}LOW} + 7 \times N_{EIF\text{-}AVERAGE} + 10 \times N_{EIF\text{-}HIGH} +$
$3 \times N_{EI\text{-}LOW} + 4 \times N_{EI\text{-}AVERAGE} + 6 \times N_{EI\text{-}HIGH} +$
$4 \times N_{EO\text{-}LOW} + 5 \times N_{EO\text{-}AVERAGE} + 7 \times N_{EO\text{-}HIGH} +$
$3 \times N_{EQ\text{-}LOW} + 4 \times N_{EQ\text{-}AVERAGE} + 6 \times N_{EQ\text{-}HIGH}$

With respect to such formula, if the measurer fails in identifying a function (file or process) and/or (s)he counts a function that should NOT be counted according to the measurement rules, (s)he is introducing an error in a single value for $N_{A\text{-}B}$ where "A" denotes the type of the function, and "B" the complexity rating. If the measurer is correct in identifying a function, but (s)he assign the wrong complexity value to it, (s)he introducing a double error, since the function is NOT counted under the correct figure AND it is wrongly counted under another figure.

For sake of readability, we rewrite the formula in abbreviated form as:

$$S = S(N_{11}, N_{12}, N_{13},\ldots,N_{51}) =$$
$$7N_{11} + 10N_{12} + 15N_{13} + 7N_{21} + 10N_{22} + 15N_{23} +$$
$$7N_{31} + 10N_{32} + 15N_{33} + 7N_{41} + 10N_{42} + 15N_{43}$$
$$+7N_{51} + 10N_{52} + 15N_{53}$$

This formula is easily recognized as a combination of additions and multiplications, where the multiplication coefficients are fixed by definition of the measurement methods, and the measures has to determine fifteen quantities N_{AB}.

If we recall that for addition: $f = f(x,y) = x + y \Rightarrow \delta_f = \sqrt{\delta_x^2 + \delta_y^2}$, and for multiplication (by a constant): $f = f(x) = k \cdot x \Rightarrow \delta_f = \sqrt{k^2 \delta_x^2} = k\delta_x$, we easily obtain the uncertainty formula for unadjusted size in the IFPUG measurement method:

$$\delta S = \sqrt{(7\delta N_{11})^2 + (10\delta N_{12})^2 + (15\delta N_{13})^2 + \ldots\ldots + (7\delta N_{51})^2 + (10\delta N_{52})^2 + (15\delta N_{53})^2}$$

Although long, this formula is quite trivial, since it derives from a linear combination of simple terms. Still, it proves that "an error in identifying logical files is much more impacting than an error in identifying elementary processes", as any experienced measurer can state by experience, since the conventional multipliers for logical files have higher values than for elementary processes.

For linear combination of simple terms, the reader with no experience in derivatives can still easily derive uncertainty propagation for other measurement methods. In the IFPUG case, a more complex case comes from applying the Value Adjustment Factor as currently proposed by the method (non-ISO case). The Value Adjustment Factor (VAF) is derived from fourteen General System Characteristics (GSC's) by the following formula:

$$VAF = \frac{1}{100}\sum_{i=1}^{14} DI_{GSC_i} + 0.65$$

where *DI* denotes the degree of influence of each GSC and can only have integer values from 0 to 5. The final function point count according to the IFPUG method is given by the general formula:

$$SIZE_{(FP)} = SIZE_{(UFP)} \times VAF = SIZE_{(UFP)} \times \frac{1}{100}\sum_{i=1}^{14} DI_{GSC_i} + 0.65$$

What is the impact on such formula if the measurer introduce an error on one or more of the fourteen general system characteristics? Recalling the fractional or

percentage formula for multiplication, the percentage uncertainty in this case is:

$$\frac{\delta S_{(FP)}}{S_{(FP)}} = \sqrt{\left(\frac{\delta S_{(UFP)}}{S_{(UFP)}}\right)^2 + \left(\frac{\delta VAF}{VAF}\right)^2}$$

where the uncertainty on VAF is given by:

$$\delta VAF = \frac{1}{100}\sqrt{\sum_{i=1}^{14}\left(\delta DI_{GSC_i}\right)^2} = \frac{1}{100}\sqrt{\delta DI_{GSC_1}{}^2 + \delta DI_{GSC_2}{}^2 + \ldots + \delta DI_{GSC_{14}}{}^2}$$

As an example, assume that the unadjusted size measures 100 unadjusted function points, with a relative error of ±5%, and that the degree of influence of any GSC is assumed to be equal to 3, with an uncertainty of ±1 per GSC. Thus:

$$S_{(UFP)} = 100 \text{ UFP}, \qquad S_{(UFP)} = 5 \text{ UFP},$$
$$\delta S/S_{(UFP)} = 5\%$$
$$VAF = 1.07, \quad \delta VAF = \frac{1}{100}\sqrt{14} \approx 0.04,$$
$$\delta VAF/VAF = 3.5\%$$

and finally:

$$\frac{\delta S_{(FP)}}{S_{(FP)}} = \sqrt{\left(\frac{\delta S_{(UFP)}}{S_{(UFP)}}\right)^2 + \left(\frac{\delta VAF}{VAF}\right)^2} = \sqrt{0.05^2 + 0.035^2} \approx 0.06 = 6\%$$

Depending on the uncertainty on both terms (size and value adjustment factors), the uncertainty on the final count S is greater than each of those by a certain amount. While in a statistical (generic) sense, we would state that errors tend to compensate, uncertainty is always growing.

Actually, even in the discussed "simple" examples, we must notice that some variable could result in being not independent (several general system characteristics in VAF can influence one each other), and that some uncertainties also could affect one each other, as in the case where an error on some function figure results in an error on another function figure, in the generic formula for size in the IFPUG model above. In such cases, the error propagation is complicated by mutual effects, and the concept of *covariance between variable pairs* should be taken into account, with a more complex form of the uncertainty formula (not discussed here).

IV. APPLICATION TO SOFTWARE ESTIMATION – MORE COMPLEX EXAMPLES

A. Constructive Cost Model (COCOMO)

The constructive cost model [3] for work effort estimation for software projects in its general form proposes some simple ranges for *optimistic* and *pessimistic* estimates as "half the original estimate" or "twice the original estimate". Such estimation ranges are not formally proved. A more reliable uncertainty range can be determined with the described approach of error propagation. To derive the development effort for a software program, the "zero-order" COCOMO formula is:

$$y = A \cdot x^B$$

where y represents the expected work effort expressed in person hours, and x the size in lines of code or functions points, with factors A and B accordingly determined by statistical regression on an historical sample [4]. Note that having fixed statistical values for some parameters in the model does not mean necessarily that these values are exact: their associated errors can be derived from the standard deviation of the statistical sample from the fitting function y. To evaluate the error on y, given the errors on the parameters A and B and on the independent variable x, we have to perform some partial derivatives, recalling that:

$$\frac{\partial}{\partial x}\left(A \cdot x^B\right) = A \cdot B \cdot x^{B-1}, \quad \frac{\partial}{\partial A}\left(A \cdot x^B\right) = x^B, \quad \frac{\partial}{\partial B}\left(A \cdot x^B\right) = A \cdot x^B \cdot \ln x$$

For instance, we perform a approximated measurement of a project in function points, and obtain for x the estimated value of 1,000 ± 200 FP, or a percentage uncertainty of 20%. Assume also, as an example, that A and B are equal to 10 ± 1 and to 1.10 ± 0.01 respectively, in the appropriate measurement units. Collecting all data and applying the error propagation for Δy, we obtain:

$$y = A \cdot x^B = 10 \cdot 1,000^{1.10} = 19,952.6 \text{ (person hours)}$$

$$\delta y = \sqrt{\left[\left(A \cdot B \cdot x^{B-1}\right)\delta x\right]^2 + \left[\left(x^B\right)\delta A\right]^2 + \left[\left(A \cdot x^B \cdot \ln x\right)\delta B\right]^2} =$$
$$= \sqrt{[21.948 \times 200]^2 + [1,995.262 \times 1]^2 + [137,827.838 \times 0.01]^2} = 5,015.9 \text{ (person hours)}$$

Taking only the first significant digit of the error, we obtain for y the estimated range 20,000 ± 5000, or a percentage uncertainty of 25%. Note that the percent error on y is not the simple sum of the percent errors on A, B, and x, because the function assumed in this example is not linear.

We consider now a further step in the COCOMO model, where the work effort derived by statistical parameters (nominal effort, y_{nom}) is to be adjusted by a series of (independent) cost drivers, c_i [3]. Although for simplicity the cost drivers assume discrete values in practical application, we can treat them as continuous variables. Here is a derivation of the uncertainty for the adjusted effort estimate:

$$y_{adj} = y_{nom} \cdot \prod_i c_i$$

$$\frac{\partial}{\partial y_{nom}}\left(y_{nom} \cdot \prod_i c_i\right) = \prod_i c_i, \quad \frac{\partial}{\partial c_j}\left(y_{nom} \cdot \prod_i c_i\right) = y_{nom} \cdot \frac{\prod_i c_i}{c_j}$$

For instance, if we consider $y_{nom} = 20,000 \pm 5000$, and a set of only 7 factors c_i, for each of which (for sake of simplicity) we assume the same value $c = 0.95 \pm 0.05$), then fro y_{adj} we obtain:

$$y_{adj} = y_{nom} \cdot \prod_i c_i = 20{,}000 \cdot \prod_1^7 0.95 = 20{,}000 \cdot 0.95^7 = 13{,}966.7$$

$$\delta y_{adj} = \sqrt{\left[\left(\prod_i c_i\right)\delta y_{nom}\right]^2 + \left[\left(y_{nom} \cdot \frac{\prod_i c_i}{c_1}\right)\delta c_1\right]^2 + \ldots + \left[\left(y_{nom} \cdot \frac{\prod_i c_i}{c_7}\right)\delta c_7\right]^2} =$$

$$= \left(\prod_i c_i\right)\sqrt{(\Delta y_{nom})^2 + (y_{nom})^2\left(\frac{\Delta c_1}{c_1} + \ldots + \frac{\Delta c_7}{c_7}\right)^2} =$$

$$= 0.95^7 \cdot \sqrt{(5{,}000)^2 + (20{,}000)^2\left(\frac{0.05}{0.95} + \ldots + \frac{0.05}{0.95}\right)^2} = 5{,}146.8 \quad 5000$$

We therefore see that *each additional factor* in the estimation process can apparently make the estimation *more accurate* (in the given example, for instance, the y_{adj} is reduced with respect to its nominal value because all the factors are smaller than 1), but its percent error is *increased* (it's now about 36%, versus the 25% of the nominal estimate).

We can then draw a general conclusion: the more we add information to refine an estimate, the more we're also adding uncertainty sources to the estimation process. Thus, the measurer should locate the right cut-off between accurate formulas and reasonable percentage errors applying to those formulas. For instance, in some cases one could decide between accepting the measured value of each cost driver in the model, refining it (if possible, reducing its error), or even avoiding completely some factor from the overall model because of any unacceptable impact on the overall uncertainty of the estimate.

B. Jensen/Putnam Model (Software Equation)

A general form of the systemic model by Jensen & Putnam is [5]:

$$Size = (\ Eff\ /\ \beta\ )^{1/3} \times Sched^{4/3} \times Prod$$

where:

Size is the size in *SLOC* ("or other measure of amount of function").

Eff (effort) is the amount of development effort in *person-years*.

β (beta) is a special skills factor that varies as a function of size "from 0.16 to 0.39" (this factor has the effect of "reducing process productivity, for estimating purposes, as the need for integration, testing, quality assurance, documentation, and management skills grows with increased complexity resulting from the increase in size" [5]).

Sched (schedule) is the development time in years.

Prod (process productivity parameter) is the productivity number used to tune the model to the capability of the organization and the difficulty of the application (by calibration – the "range of values seen in practice [for *Prod*] across all application types is 1,974 to 121,393" [5]).

The so-called *software equation* by Jensen and Putnam can be put in several equivalent forms, depending on which variables are considered as independent, and which variable(s) is to be derived, for estimation purposes. As a common scenario, assume that: *Size* is measured in advance according to an appropriate measurement method and unit, *Sched* is constrained by market needs (although this may present strong risks in real world applications), β and *Prod* are determined by regression and calibration from known projects. Therefore, the effort *Eff* is the dependent variable in this scenario, and its formula is:

$$Eff = \beta \times Size^3 \times Prod^{-3} \times Sched^{-4} = \frac{\beta \times Size^3}{Prod^3 \times Sched^4}$$

The partial derivatives of *Eff* with respect to the given variables are:

$$\frac{\partial Eff}{\partial \beta} = \frac{Size^3}{Prod^3 \times Sched^4}$$

$$\frac{\partial Eff}{\partial Size} = \frac{3\beta \times Size^2}{Prod^3 \times Sched^4}$$

$$\frac{\partial Eff}{\partial Prod} = \frac{-3\beta \times Size^3}{Prod^4 \times Sched^4}$$

$$\frac{\partial Eff}{\partial Sched} = \frac{-4\beta \times Size^3}{Prod^3 \times Sched^5}$$

To instantiate the example, we can consider some figures from [5] for a real time avionic system (uncertainties are put by the author for the example to follow):

Size = 40,000 SLOC in C++, with an uncertainty of ± 2,000, or 5%;

β = 0.34, with an uncertainty of ± 0.02, or approx. 5%;

Sched = 2 years, with an uncertainty of ± 0.1, or approx. 5%;

Prod = 3,194, with an uncertainty of ± 160, or approx. 5%;

As correctly reported in [5], the expected value for effort *Eff* is then equal to 41.74 person-years (approx. 500 person-months). For the uncertainty on the computed *Eff*, we find:

$$\delta Eff = \sqrt{\left(\left|\frac{\partial Eff}{\partial \beta}\right|\delta\beta\right)^2 + \left(\left|\frac{\partial Eff}{\partial Size}\right|\delta Size\right)^2 + \left(\left|\frac{\partial Eff}{\partial Prod}\right|\delta Prod\right)^2 + \left(\left|\frac{\partial Eff}{\partial Sched}\right|\delta Sched\right)^2} = 12.42\,py$$

that is, $\delta Eff/Eff \approx 30\%$. We therefore see that mixing four parameters in the software equation, each with an uncertainty of approximately 5%, leads to an overall uncertainty of 30% on the effort final estimation. From a mathematical point of view, this is due to the highly non-linear form of the model.

If one wants to reduce the uncertainty of the effort estimation (provided that all figures are reliable), (s)he can only act on the uncertainties of each parameter. To evaluate which parameter is more influencing the error

propagation, we make the following assumption – we do reduce the uncertainty on only one parameter per try, leaving unchanged the uncertainties on the remaining parameters.

For instance, if we put an error of only 1% on *Size*, we obtain $\delta Eff/Eff \approx 26\%$.

If we put an error of only 1% on β, we obtain $\delta Eff/Eff \approx 29\%$.

If we put an error of only 1% on *Prod*, we obtain $\delta Eff/Eff \approx 26\%$.

If we put an error of only 1% on *Sched*, we obtain $\delta Eff/Eff \approx 22\%$.

So, given that it is realistic to refine the estimation on the input variables of the software equations, we should conclude that *Sched* is the most impacting factor in the Jensen & Putnam model, followed by *Size* and *Prod* in equal measure. Such evaluation is confirmed by the power value carried by such parameters in the software equation.

V.   CONCLUSIONS

In real cases, the scope of a software project is often not fully defined in early phases; the main cost driver, the size of the project, could thus be not accurately known in the beginning, when software estimates are more useful. Different values estimated for the size provide different estimates for the effort, of course, but this fact does not provide by itself any consideration of the precision of a single estimate. Any estimation model cannot be seriously performed without consideration of its possible deviations between estimates (expected values) and true values. Also, when deriving metrics from directly measured factors, we must consider the impact that a (small) error in any direct measure have on the derived metrics, depending on the algorithm or formula used to derive them.

Considering the formal error propagation theory can add a new perspective in the evaluation of software metrics, in the choice of a software measurement method when compared to others, as well as in the choice of a software estimation approach or model among the possible options. The quality and advantages claimed by any method or model can be enforced or diminished when an objective analysis of such method or model is performed in terms of error propagation and overall accuracy that it can offer. Error propagation theory does provide some useful insights into such topic, from both a theoretical (method's/model's form) and a practical point of view (application in real cases).

REFERENCES

[1] Boehm, B. et al., COCOMO II Model Definition Manual, Version 1.4, University of Southern California, 1997. URL: http://goo.gl/jkv1ZR

[2] IFPUG, Function Point Counting Practices Manual, Version 4.3.1, International Function Point Users Group, 2010. URL: www.ifpug.org

[3] ISBSG, Practical Project Estimation. A toolkit for estimating software development effort & duration, International Software Benchmarking Standards Group, 2010, URL: http://goo.gl/QYnxHi

[4] Jensen, RW, Putnam, LH, Roetzheim, W, Software Estimating Models: Three Viewpoints, Crosstalk, Feb. 2006. URL: http://goo.gl/84Zya4

[5] Santillo L., Error Propagation in Software Measurement & Estimation, Proceedings of IWSM 2006 (International Workshop on Software Measurement), November, 2-4, 2006, Potsdam (Germany)

[6] Taylor, J.R., An Introduction to Error Analysis, The Study of Uncertainties in Physical Measurements, University Science Books, 1982.