PERFORMANCE CRITERIA FOR SOFTWARE METRICS MODEL REFINEMENT

Adrian VISOIU[1]

PhD Candidate, Assistant Lecturer, Economic Informatics Department,
Academy of Economic Studies, Bucharest, Romania

E-mail: adrian.visoiu@csie.ase.ro

Abstract: *In this article, the refinement process is presented with respect to model list building using model generators. Performance criteria for built models are used to order the model lists according to the needs of modelling. Models are classified by means of performance and complexity. An aggregated indicator based on the two factors is proposed and analysed in model list ordering. A software structure for model refinement is presented. A case study shows practical aspects of using the aggregated indicator in model refinement.*

Key words: software; metrics; modelling; performance criteria

1. Model design concepts

The software metrics refinement is the process of building models for software metrics estimation and choosing among them those who explain the studied phenomenon. Choosing a certain model has to underlie on objective criteria such as statistical performance and expression complexity.

A set P containing the programs P_1, P_2 ,..., P_n is considered. A software quality system has the characteristics C_1, C_2 ..., C_m. For a certain characteristic C_i, several models, denoted by M_{i1}, M_{i2} ..., M_{ik}, are built in order to estimate its level.

These models have the following form:

M_{ih}: $y_i=f_{ih}(X_1, X_2, ..., X_r)$.

When building the analytical form of $f_{ih}()$, variables, coefficients and operators are taken into account.

A large series of analytical expressions associated to software quality characteristics assessing processes are built. The model for estimating software developing effort E in man hours, for implementing a software product with $KLOC$ thousands of source lines, has the expression:

$E(KLOC)=a+b\ KLOC^c$.

As the feasibility of automated recording of influence factors levels for a certain characteristic C_i increases, premises arise for building more complex analytical expressions containing large number of variables and a large number of operators. That is why the differences between models should be studied and quantified.

The objective of software metrics estimation is prognosis for the values of the studied variables and using that information as underlying for management decisions at software project management level.

2. Model refinement process

In the context of model generation, the refinement process is described as follows. Consider the dataset *S* and a model generator *G*. The set of models generated by *G* to fit the dataset *S* is L_M. The model set L_M is characterized by the following:

- the number of models is large
- models have a variety of values for the performance criterion ranging from the worst fitting model to the best fitting model
- models have a variety of values for the complexity, ranging from the most simple expressions to very complex ones
- the best fitting models are not compulsory the most complex ones, and also, the simplest models are not the worst in explaining the studied phenomenon.

The refinement process takes the set L_M of generated models and produces a new model set L_M' containing less models from the initial set. The models are chosen to meet certain criteria. The main criterion a model must meet is its capacity to explain the studied phenomenon. This is achieved using statistical performance indicators such as the sum of squared differences or R^2. Model complexity is also taken into account as the expressions used in practice need to be simple and easy to explain and interpret.

The sum of squared differences is an absolute measure of the quality of a model. Consider that the studied variable, *Y*, has n entries in the dataset, denoted by y_i. The estimated values using a model are denoted by $\hat{y}_i$. The sum of squared differences is obtained by:

$$SS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

This statistical performance indicator shows the unexplained variance. Its value is always positive and grows with the size of the data series. In the context of model generation it can be used for model comparison as long as the length of the data series is the same for all models. To remove this deficiency the standard error of differences *SE* is computed as

$$SE = \sqrt{\frac{SS}{n-2}}$$

In the following sections, the sum of squared differences *SS* is used as the dataset used in estimation is the same for all models and comparison can be done using this indicator. For the same dataset, if a model M_1 has a lower *SS* value than another model M_2, the M_1 model explains better the studied phenomenon.

An analytical expression for a model *M* contains different elements whose number of apparitions are used in measuring its complexity using a Halstead software metric:

$C(M)=n_1 \log n_1+n_2 \log n_2$, where

n_1 - number of operands, variables and coefficients;

n_2- number of operators.

For the model:

$y = a\,x + b\,x^3 + c\,u^3 + d$

the table 1 is built.

Table 1. Number of apparitions for operands and operators

Operands	**Frequency**
x	2
u	1
a	1
b	1
c	1
d	1
y	1
3	2
n_1	10
Operators	
*	3
+	3
$()^{\alpha}$	2
n_2	8

The complexity C is given by the relation:

$C = n_1 \log_2 n_1 + n_2 \log_2 n_2 = 10 \log_2 10 + 8 \log_2 8 = 57.21928095$

It is desired, that models used in practice to be simple and easy to interpret, leading to low complexity expressions.

The refinement process using model generators is described by the following diagram shown in figure 1.

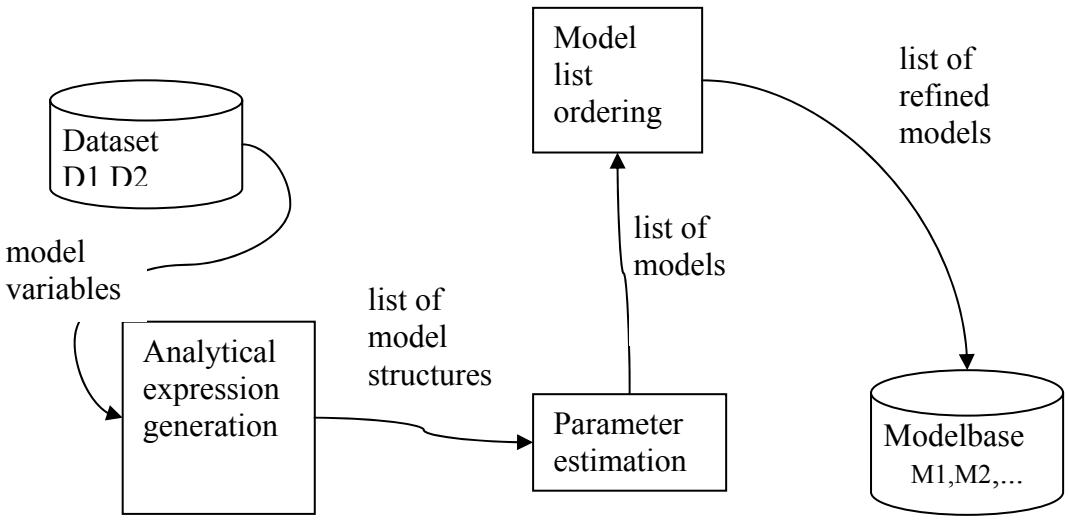

Figure 1. The model refinement process

The refinement process takes the following steps:

- the dataset is built; it contains data series for the dependent variable and independent variables

- using variable names found in the dataset, model structures are generated; the list of generated model structures is denoted by L_G and contains a large number of models, also depending on the constrains or type of the generation algorithm
- for each model structure in L_G, coefficients are estimated, along with statistical performance indicators, obtaining the list of estimated models, L_E
- for each model found in L_E, an aggregated performance indicator is computed; the L_E list is ordered by this performance indicator and an arbitrary number of models is chosen, forming L_R list, the refined list of models;
- the refined list of models is saved into a modelbase and then used by the human analyst to choose one or more models to be used in estimating the studied phenomenon.

3. Performance criteria

When using model generators, a large number of models is produced. From statistical point of view ordering the generated list of models by the performance indicator chooses the models that best represent the studied phenomenon. Ordering the list by the complexity of the model pays attention to simple models, but with significant loss of information.

Let *SS* denote the sum of squared differences as performance indicator for statistical performance, and *C* denote the complexity.

To classify generated models in two categories by means of complexity, an arbitrary value C_C is chosen. Models with complexity less than C_C are considered simple. Models with greater complexity than C_C are considered complex. To classify generated models in two categories by means of performance, an arbitrary value SS_C is considered. It is considered that models with performance indicator less that SS_C show little error in explaining the phenomenon, as models with greater performance indicators than SS_C do not represent correctly the phenomenon.

Using the above partitioning, the generated models can be classified into 4 categories as shown in figure 2.

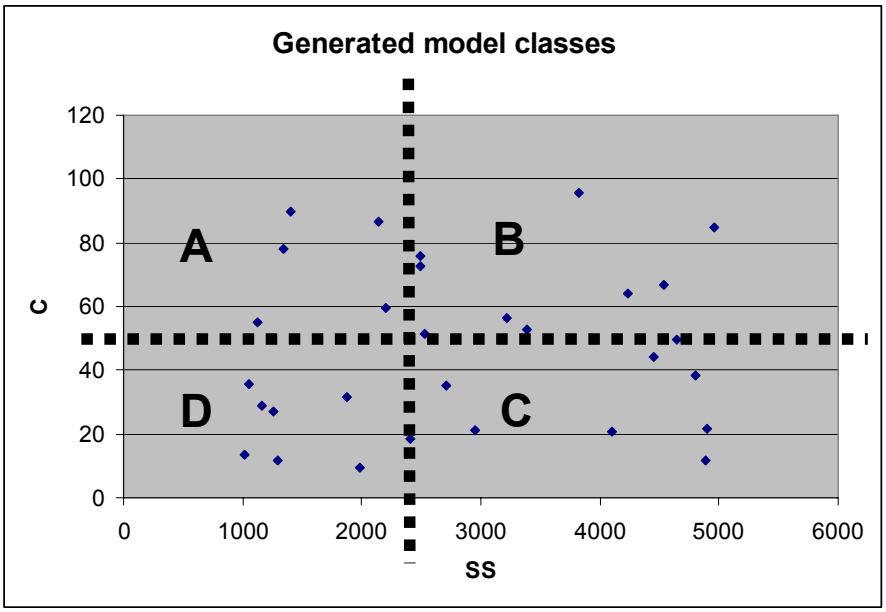

Figure 2. Model classification by performance and complexity

The models shown in figure 1 have certain characteristics:

A - complex models that explain the phenomenon; they are usually models with very good statistical performance

B - complex models but weak in explaining the phenomenon; they usually include many factors that do not have influence, and analytical expression include many operands and functions that do not express correctly the connection between factors

C - simple models but weak in explaining the phenomenon; usually, they do not include enough factors, and use simple operands and functions

D - simple models that explain the phenomenon; it is desired that the models used to be from this category.

It is obvious that an aggregated indicator is required to take into account both considered aspects of a model, the statistical performance and expression complexity. This is achieved by using an utility function associated to the model *f(c,s),* a two variable function, where *c* denotes the complexity of the model's analytical expression and *s* denotes the statistical performance indicator.

Consider that the value of the aggregated performance indicator must be minimized, e.g. the sum of squared differences. This leads to ascending ordered model lists by this indicator. The main properties of the function are:

- the function increases with the growth of complexity; the more operands and operators an expression has, its complexity grows; model generators that use only statistical performance to order the generated model lists, usually create long analytical expressions; the complexity is a criterion that has to be minimized;
- if the statistical performance indicator is to be minimized (e.g. the sum of squared differences), the function that computes the aggregated indicator value must increase while the factor raises.

The aggregated performance indicator is used only in the refinement of models. It is not intended to replace the statistical performance indicators. Its purpose is to help the analyst to order model lists accordingly to his needs.

Such an indicator is the weighted performance indicator P_M given by:

$P_M = SS_M^p \cdot C_M^q$

where

SS_M – *SS* indicator for model *M*;

C_M – complexity of *M*.

p – coefficient of importance for statistical indicator

q - coefficients of importance for expression complexity

This indicator has the properties exposed above. For example, for $p=1$ and $q=0$, the indicator becomes the same with *SS*. Setting different values for *p* and *q*, the importance of the two factors is modified.

4. Software for model refinement

Model refinement is achieved using dedicated software. Modelbases create the context for model refinement offering specific instruments for modeling and integrating aspects of modeling activity.

Modelbases are complex software constructions offering functions for:

- defining, retrieving and updating models
- modeling applications management
- estimation and validation of coefficients
- automated model generation from existing datasets
- dataset management

A representation of the modelbase structure detailing the model generator branch is presented in figure 3.

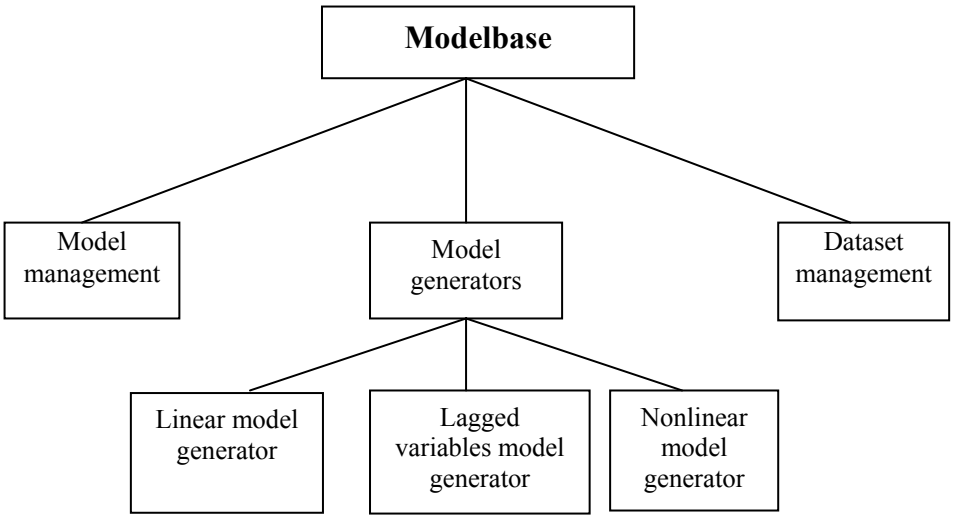

Figure 3. Modelbase structure with emphasis on model generators

Model generators are modelbase instruments that build model structures from a given class using variables found in the dataset given as input. Model classes group models with the same structure, e.g. linear models, linear models with lagged variables, nonlinear models. For each class a model generator is developed. Each dataset contains data series for the recorded variables. The endogenous variable is specified and the generator builds analytical expressions using influence factors. For each model structure, coefficients are estimated and a performance indicator is computed. The resulting model list is ordered by the performance indicator. The analyst chooses between the best models an appropriate form that later will be used in estimating the studied characteristic.

The nonlinear model generator is intended for building nonlinear models for software metrics estimation. The algorithm is based on generating expressions in polish form using a combinatorial method.

An expression is built up of operands and operators. In this case, operands are represented by independent variables and coefficients. Operators consist of elementary operations and other functions.

Consider the multiplication operator, *, and two operands. *a* and *b*. The operation *a*b* can be viewed as the result of a function, *multiply*, that has the form:

*multiply(a ,b) = a*b*, where *a ,b* are real numbers

In the same manner other operations are treated, building

plus(a, b) = a +b

$power(a, b) = a^{b}$

$log(a ,b) = log_{a} b$

This representations permits rephrasing of expressions, for example, given the expression:

$a + b * c + d^e$, it is equivalent with

plus (plus (a , multiply (b , c)) , power (d , e)).

Simplifying, it can be written as:

*+ + a * b c ^ d e*, where ^ denotes rising to power operation.

It is observed that this form corresponds to the polish notation, and has several advantages:

- it is easy to evaluate such an expression using a stack and pushing element by element in reverse order as long as that element is an operand or popping and operating a number of operands when the element is an operator and then pushing the result back on the stack; the value that remains on the stack is the value of the expression
- it is easy to build an expression directly in polish notation and when needed to be reconstructed an ordinary form.

In order to build an expression directly in polish notation, expression elements are separated into sets.

Let the set containing the influence factors symbolic names be denoted by *V*. Let the set containing the operators be denoted by *O*. Let *CO* denotes the set {*q*} where *q* marks the apparition of a coefficient in the expression. The set of expression elements is the reunion *E=V* U *CO* U *O*. Consider the model:

$TIME\_DEV = a * CC^b * PR^c + d$

where

- studied variable: *TIME_DEV*, the time needed to develop a software module
- influence factors: *CC* – the cyclomatic complexity of module assessed at design time, *PR* – the rating of the programmer assigned to this activity; *V*={*CC , PR*}
- operators: *O* = {+, *, ^}
- coefficients : {*a, b, c, d*}

The equivalent polish notation is:

*+ * a * ^ CC b ^ PR c d.*

The nonlinear model generator uses internally the above representation of models. Using a combinatorial algorithm expressions are built directly in polish form. The parameters for this process are:

- the operand set, *V*
- the coefficient set, *CO*
- the operator set, *O*
- the maximum length of the stack
- the maximum complexity of the generated expression

The model generation is an important step in the refinement process. The nonlinear model generator is suitable for modeling as the phenomena do not always follow linear laws.

5. Experimental results

Consider the model that estimates the time required for fixing a defect from a program module taking into account its severity and the number of code lines estimated to be modified or added:

FIX_HOURS = f(LOC_COUNT, SEVERITY)

where

FIX_HOURS – the endogenous variable to be estimated, measured by man hours

LOC_COUNT – estimated number of lines of code involved in the fix

SEVERITY – the severity of the defect, between 1 and 5

The sets taken into account are:

- *S = {plus, multiply, power}*
- *V = { LOC_COUNT, SEVERITY }*

It observed that only addition, multiplication and raising to power operations are permitted.

Using a nonlinear model generator with specific constraints the list of generated models is filtered and reduced in size. Constraints limit consecutive raising to power operations, consecutive operations on coefficients and a maximum stack dimension of 13. Consider the dataset D described in table 2.

Table 2. Dataset used for model generation

No.	FIX_HOURS	SEVERITY	SLOC_COUNT
1	1	2	4
2	1	2	10
3	1	3	1
4	1	3	2
5	2	2	15
6	3	3	15
7	40	5	200
....			
102	8	3	2
103	20	2	265
104	30	5	20

The list of generated models is presented in table 3.

Table 3. List of generated models for the selected dataset

ID	Expression
M_1	$2.86*SEVERITY+0.284*LOC\_COUNT-3.394$
M_2	$1.115*10^{-4}*SEVERITY^{7.781}$
M_3	$1.196*LOC\_COUNT^{0.753}-0.134$
M_4	$1.125*(SEVERITY*LOC\_COUNT)^{0.601}-0.898$
M_5	$1.081*SEVERITY^{0.036}*LOC\_COUNT^{0.763}+0.082$
M_6	$0.321*SEVERITY^{2.204}+0.501*LOC\_COUNT^{0.891}+0.253$
M_7	$(2.877*SEVERITY+0.833*LOC\_COUNT)^{0.814}-2.827$

For $C_C = 23$ and $SS_C = 2.00$ the generated models classify as presented in figure 4.

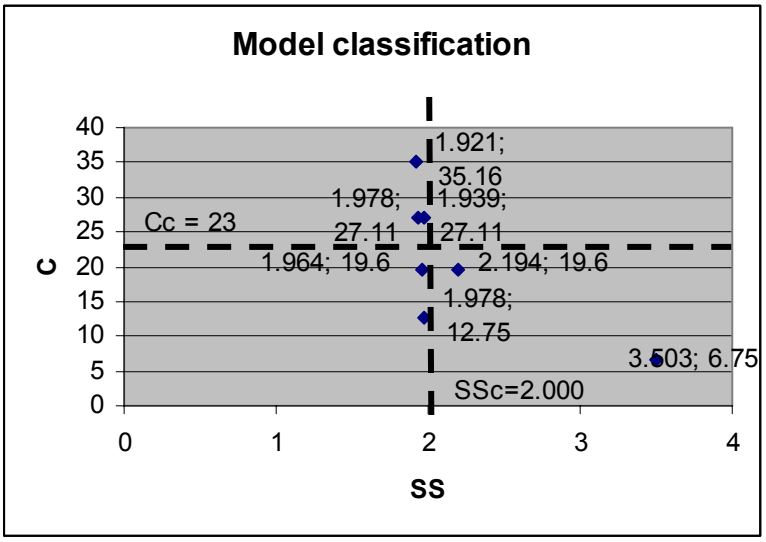

Figure 4. Generated models classification

Ordering model lists by statistical performance indicator is done when model performance is the main objective of the analysis. The list of models, ordered by statistical performance indicator *SS*, given as output is shown in table 4.

Table 4. Models ordered by statistical performance indicator *SS*

ID	Expression	SS (10^4 units)	C
M_6	$0.321*SEVERITY^{2.204}+0.501*LOC\_COUNT^{0.891}+0.253$	1.921	35.16
M_7	$(2.877*SEVERITY+0.833*LOC\_COUNT)^{0.814}-2.827$	1.939	27.11
M_1	$2.86*SEVERITY+0.284*LOC\_COUNT-3.394$	1.964	19.60
M_3	$1.196*LOC\_COUNT^{0.753}-0.134$	1.978	12.75
M_5	$1.081*SEVERITY^{0.036}*LOC\_COUNT^{0.763}+0.082$	1.978	27.11
M_4	$1.125*(SEVERITY*LOC\_COUNT)^{0.601}-0.898$	2.194	19.60
M_2	$1.115*10^{-4}*SEVERITY^{7.781}$	3.503	6.75

As seen in table 4 the best model that estimates *FIX_HOURS* is M_6, taking into account only the quality of estimation. However, it has the greatest complexity among all the rest, and uses five coefficients and two variables. Other models with resembling performance are M_7, M_1, M_3, M_5.

Ordering the list by the weighted indicator P_M with parameters p=3 and q=1, M_3 becomes eligible. It has few parameters and a single variable that is easy to obtain data for. It uses only a variable to asses the phenomenon. The results are shown in table 5.

Table 5. Models ordered by aggregated indicator with p=3 and q=1

ID	Expression	SS (10^4 units)	C	PM p=3 q=1
M_3	$1.196*LOC\_COUNT^{0.753}-0.134$	1.978	12.75	98.67089
M_1	$2.86*SEVERITY+0.284*LOC\_COUNT-3.394$	1.964	19.60	148.4843
M_7	$(2.877*SEVERITY+0.833*LOC\_COUNT)^{0.814}-2.827$	1.939	27.11	197.6346
M_4	$1.125*(SEVERITY*LOC\_COUNT)^{0.601}-0.898$	2.194	19.60	206.9979
M_5	$1.081*SEVERITY^{0.036}*LOC\_COUNT^{0.763}+0.082$	1.978	27.11	209.8014
M_6	$0.321*SEVERITY^{2.204}+0.501*LOC\_COUNT^{0.891}+0.253$	1.921	35.16	249.2476
M_2	$1.115*10^{-4}*SEVERITY^{7.781}$	3.503	6.75	290.1511

The parameter setting shown in table 5 pays more attention to model performance. It is recommended that analysts use such settings when the studied phenomenon do not offer clues for the form of the link between the variables.

The *p* and *q* parameters are used to increase or decrease the importance of factors. Different values for *p* and *q* offer different list orderings according to analyst's interest. For example, the list ordering for $p=1$ and $q=2$ is shown in table 6.

Table no.6 Models ordered by aggregated indicator with p=1 and q=2

ID	Expression	SS (10^4 units)	C	PM p=1 q=2
M_2	$1.115*10^{-4}*SEVERITY^{7.781}$	3.503	6.75	159.6054
M_3	$1.196*LOC\_COUNT^{0.753}-0.134$	1.978	12.75	321.5486
M_1	$2.86*SEVERITY+0.284*LOC\_COUNT-3.394$	1.964	19.60	754.4902
M_4	$1.125*(SEVERITY*LOC\_COUNT)^{0.601}-0.898$	2.194	19.60	842.8470
M_7	$(2.877*SEVERITY+0.833*LOC\_COUNT)^{0.814}-2.827$	1.939	27.11	1425.0721
M_5	$1.081*SEVERITY^{0.036}*LOC\_COUNT^{0.763}+0.082$	1.978	27.11	1453.7353
M_6	$0.321*SEVERITY^{2.204}+0.501*LOC\_COUNT^{0.891}+0.253$	1.921	35.16	2374.7894

The parameter setting shown above pays attention to very simple models. In this case simplicity is more valuated than performance. This kind of setting is recommended only when there is a clue that the link between the studied variables follows a simple expression.

Subsequent updates on the database require that periodical estimation of coefficients to track changes.

The best models obtained are stored and operated on using the modelbase.

5. Conclusions

The refinement process is guided by the analyst according to his needs. This process is complex and operates on model sets. Model sets are built by model generators using the datasets that are to by analyzed.

Performance criteria are needed to choose among the models. The analyst uses these criteria to order generated model lists.

The most important criterion for list ordering is the statistical performance. The model is used to explain a phenomenon and predict evolution when giving certain inputs. Ordering the model lists by this criterion gives access to the best models and it is recommended to use it when precision is a critical aspect of the modeling activity.

Model complexity is a criterion that orders the models according to their structure. It is incorrect to use the complexity alone when ordering model lists.

The aggregated performance indicator is a solution to combine both studied aspects of a model. The analyst can set parameter settings that give separate importance to performance and complexity. The purpose of using the aggregated performance indicator is to obtain simple models but with minimum loss of information in explaining the studied phenomenon.

On the other hand, the analyst is required to interpret results, to asses different variants. A group of models from the top of the ordered list are chosen and used. It is important to validate the models assessing their performance over time or using other datasets.

The aggregated performance indicator does not replace statistical performance indicators. It is intended to be used only in the context of model generation.

Bibliography

1. Boja, C., Ivan, I. **Metode statistice în analiza software,** Editura ASE, Bucharest, 2004
2. Ivan, I., Popescu, M. **Metrici software,** Editura Inforec, Bucharest, 1999
3. Ivan, I., Visoiu, A. **Baza de modele economice,** Editura ASE, Bucharest, 2005
4. Jalote, P. **Software Project Management in Practice,** Addison Wesley, 2002
5. Toma, C., Ivan, I., Popa, M., Boja, C. **Data Metrics Properties,** Proceedings of International Symposium October 22-23, Iasi, 2004, p. 45-56
6. Visoiu, A., Garais, G. **Nonlinear model structure generator for software metrics estimation,** The 37th International Scientific Symposium of METRA, Bucharest, May, 26th - 27th, 2006, Ministry of National Defence, published on CD
7. Visoiu, A., Ivan, I. **Rafinarea metricilor software,** Economistul, supliment Economie teoretica si aplicativa, nr.1947 (2973), 29 august 2005

---

[1] Adrian Visoiu graduated the Bucharest Academy of Economic Studies, the Faculty of Cybernetics, Statistics and Economic Informatics. He has a master degree in Project Management. He is a PhD student at Doctoral School of Bucharest Academy of Economic Studies in the field of Economic Informatics.
He is an assistant lecturer in the Economic Informatics Department of the Bucharest Academy of Economic Studies.
He published 7 articles and he is coauthor of "Baza de modele economice" book.