A Comparative Analysis of Software Refinement Techniques

Ion IVAN
Economic Informatics Department, Academy of Economic Studies,
Bucharest, Romania

Adrian VIŞOIU
Economic Informatics Department, Academy of Economic Studies,
Bucharest, Romania

ABSTRACT

The concept of refinement is defined for models used in estimating and measuring software quality. Both classical and genetic algorithms based refinement methods are presented. Specialized software is used for developing model refinement processes. The comparative analysis of refinement techniques emphases the situations in which a certain technique should be used and defines the limits in which of the refinement result leads to efficient models. For a homogenous set of C++ programs, initial models are built and then refined, the resulting models being validated.

Keywords: software metrics, model refinement, analysis

1. MODEL STRUCTURES

Processes and phenomena are modeled in order to estimate their dynamic behavior. There are numerous criteria to classify models.

- by the number of equations , there are models with a single equation and models with many equations
- by the nature of the variables, models are deterministic or stochastic
- by the size criterion, there are small size models, including few exogenous variables and big size models with many exogenous variables
- by linearity criterion, there are linear models and nonlinear models
- by deriving criterion, there are base models and derived models
- by the nature of the solutions, there are approximated solutions models, fuzzy, neural network based, genetic algorithm based and exact solution models
- by the aimed objective, models are for prognosis, optimization simulation, heuristic and evolution.

When a phenomenon or a process is modeled, the nature of the variables and factor interaction are analyzed and a set of models is identified, models which accurately represent the structure, the dynamics and the behavior of the phenomenon or the process. A criterion for model ordering is defined, data is recorded, model coefficients are estimated and model quality assessed. From the ordered list of models, only a few are chosen and those are taken into account for validation. After validating models, only a model is left and this is subject for refinement.
A technology for doing that is defined distinguishing the convergent - iterative nature of the model obtaining process as shown in figure 1.

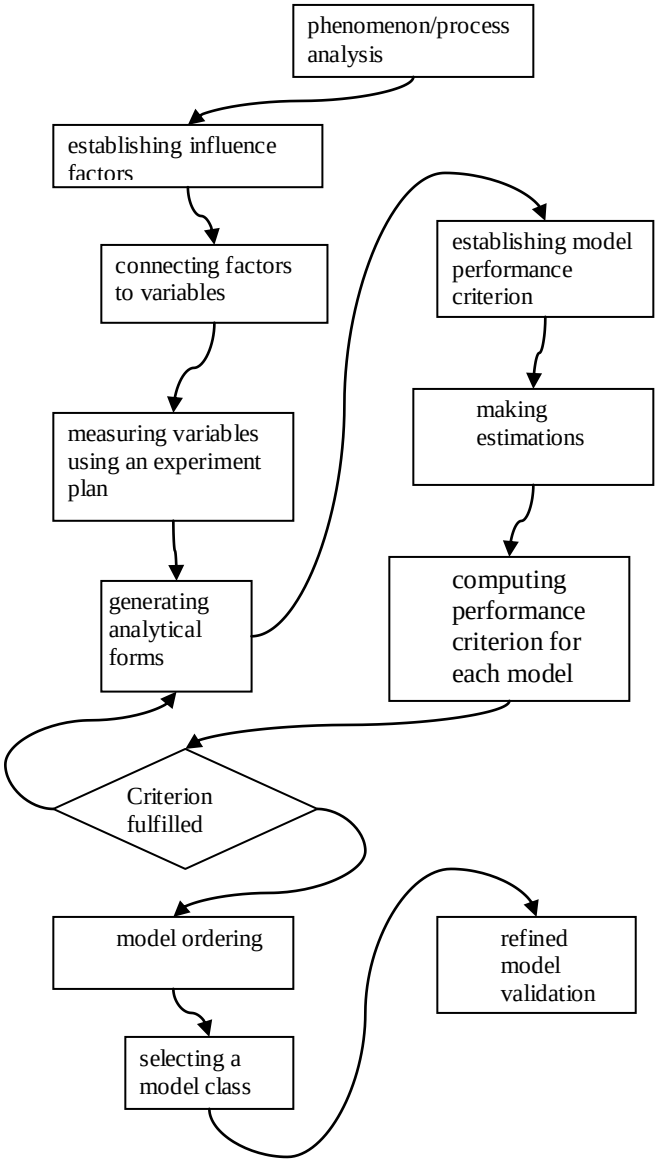

Figure no. 1 Convergent – iterative process for building process/ phenomenon estimation models

Considering the peculiarities of real processes and phenomena, a new cybernetics branch is developed, **model design**.
To developing models from a single model class whose properties are well known by the designer and there is software available for implementation, the model building technology adapts the selected model set structure to the input datasets.

2. MODEL GENERATORS

Model generators are software instruments for obtaining models from a certain model class given the list of variables, the model structure, existence restrictions and datasets.

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 as a software module. Each dataset contains data series for the Linear model generators take as input a dataset containing a number of independent variables and a dependent variable and produce linear models combining influence factors.

The practice conducted, in general, to the elaboration of linear models because: the studied phenomena aim a linear dependence, the parameters estimation methods are customary for this type of models, the results interpretation is lightened if the linearity hypotheses are taken into account.

The linear generators take as input: the list of independent variables, the dependent variable, the dataset, restrictions about the dimension and the complexity of the model, performance criterion for all generated models. The output consists of: the list of generated models ordered by the performance criterion.

In [8] nonlinear generators are described. Standard nonlinear model generators use predefined analytical forms for generating models. General nonlinear model generators build automatically analytical expressions containing influence factors. Analytical expressions of models are generated directly in polish form using a backtracking based algorithm with severe restrictions. The parameters for this process are: the operand set, the coefficient set, the operator set, the maximum length of the stack, the maximum complexity of the generated expression. The nonlinear model generator is suitable for modeling as the phenomena do not always follow linear laws.

The linear models generators with delayed arguments allow the elaboration of constructions which permit the modeling of the multiple stimulation effects which are found on short term in influences from all the sets. The phenomenon evolution shows that the factors differently influence the resultative variable. More, the variation at a moment *t* of a factor spread them with a delay abroad the evolution of the resultative variable. The delayed arguments model generator takes the same inputs, as the linear generator, but it also does not only combinations of variables, but also combinations of delays for the variables included in a certain model. As a new parameter for this algorithm, the maximum allowed delay is taken.

Model generators are important instruments for the different refinement methods, but also generally for model design.

3. MODEL COMPLEXITY

In [6] complexity classes for models estimating software quality characteristics are presented.

The complexity in Halstead sense for a model is given by the relation:

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

where

n_1 - number of operands (variables and coefficients);

n_2- number of operators.

The general model for complexity in Halstead sense is:

$$C_2 = (\sum_{i=1}^{m} f_i p_i)\log_2(\sum_{i=1}^{m} f_i p_i) + (\sum_{j=1}^{n} h_j q_j)\log_2(\sum_{j=1}^{n} h_j q_j) \quad (1)$$

where

m- number of operands

f_i – frequency of appearance of operand *i*

p_i – weight of the i[th] operand

recorded variables. The endogenous variable is specified and the generator builds analytical expressions using influence factors, coefficients, simple operators and functions. 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.

n – number of operators

h_j – frequency of appearance for j[th] operator

q_j- weight of j[th] operator, built from operator precedence table

The C_2 indicator is connected to an indicator normalized on the [0,1] range permitting a good assessing of model complexity.

Operand and operator diversity leads to an increased complexity of the model.

4. REFINEMENT THROUGH VARIABLE ELIMINATION

The independent variables X1, X2, .., Xn and the dependent variable Y are considered. For the given dataset, the complete model is built, containing all the variables:

$$y = \sum_{i=1}^{n} a_i \cdot x_i \quad (2)$$

Coefficients are estimated and the model is assessed using as performance criterion SS, the sum of squared differences between estimated values for the dependent variable and the real values. Also, the coefficient of determination is fit to be used as performance criterion.

Variants of models are built by removing one by one an independent variable. For each model variant built, coefficients are estimated. The SS indicator is computed and the model list is ordered by it. The first model is chosen its SS indicator value being the smallest among all. This is the result of the 1-refinement process.

The same way, combinations of two independent variables are removed. The number of models obtained is combinations of n taken as 2. Coefficients are estimated and the SS indicator computed for each model. The model with the smallest SS indicator is chosen as the result of the 2-refinement process.

The process continues until the linear model has the form:

$y = a_i * x_i$, $_i$=1, 2, .., n

There is a number of 2^n-2 model variants.

5. REFINEMENT THROUGH COMPLEXITY DECREASE

Reducing nonlinearities refers to the situation in which power terms are replaced by variables, function calls are replaced by variables, function composition is replaced by directly aggregated functions.

The model:

$y=ax^2+bz^2+cu^2+e$ is replaced by $y = Ax+BZ+CU+E$

The model

$y=a$ tg $x+b$ ln $u+ e$ is replaced by $y=Ax+Bu+E$

The model $y = a \sin(\log x)+b\ e^{\sin u}+g\ \ by\ y = A \sin x*\log x+B\ e^{x}*\sin x +G$

Reducing nonlinearities simplifies the model and creates the context for easy to observe processings and allow term removal in following refinement steps.

6. REFINEMENT THROUGH GENETIC ALGORITHMS

For genetic algorithms, which implement the model of population evolution, one important application is symbolic regression. Symbolic regression evolved itself with the introduction of genetic programming and later with the gene expression programming. Starting from a dataset in which it is specified the dependent variable and the independent variables, an initial population of chromosomes is built and then it is subject to a replication process including specific rules.

These algorithms have a very specific way of representing analytical expressions of models. The chromosome is a linear structure of fixed length made up of genes. The role of the chromosome is to code an analytical expression. A gene structure is obtained from the syntax tree of the expression it represents. The nodes of the tree are numbered starting from the root and then level by level and from left to right obtaining a linear structure. Each node contains either an operator, a constant or a variable. Like the nonlinear generator do, the domain for the generated expressions depends on the set of accepted operators and the set of operands built up from variables and coefficients. To create the final expression from the chromosome, the subexpressions derived from genes are aggregated using a simple aggregation function like summation of multiplication.

An initial population of chromosomes is considered. The evolution is obtained through applying genetic operators:

- selection implies extracting a number of individuals from the population based on a measure for their fitting to the aimed objective
- mutation implies random changes of some positions in the chromosome; a certain position contains an operand or an operator and its change leads to a new analytical form, differing from the initial one, when the expression is rebuilt; this genetic operator is essential to introduce a degree of variability in the generation process
- transposition implies changing the position of sequences of elements from a gene inside a chromosome
- recombination implies pairing two parent chromosomes and obtaining a new chromosome inheriting contents from both parents

The genetic operators are applied for a number of generations leading to a best fitting model in the given hypothesis.

The main drawback in analytical expression generation using gene expression programming is the building of high complexity expressions in contrast with the objective of model refinement as shown in the next section.

7. EXPERIMENTAL RESULTS

The dataset considered for experimental results refers to software metrics collected for a project. The dataset contains data series for 40 software modules and is a sample extracted from the public dataset JM1 found at [14]].

ERR – The number of defects associated with a module

LOC – number of source code lines

RM – Branch count metrics showing the number of excedent arcs from decision nodes of he program

CC – The cyclomatic complexity of a module

Using the refinement through variable elimination it is desired to obtain a model for estimating the number of defects from a software module, ERR, using as independent variables the others recorded.

A linear model containing all the variables is built

ERR_EST = a*LOC+b*RM+c*CC+d

where

ERR_EST – estimated value of defects from a software module.

Estimating the coefficients through the least squares method, the initial model is:

M_0: ERR_EST = 0.0038*LOC+ -0.00025*RM+ 0.0098*CC+ 5.2707.

About the quality of the model, the sum of squared differences is SS = 175,70 and standard deviation is 2.15 and adjusted coefficient of determination is $R^2_{adjusted}$=0,7174.

The refinement through variable elimination implies building models from this initial structure by removing one variable at a time. The following model structures are obtained:

M1: ERR_EST = a*LOC+b*RM+c

M2: ERR_EST = a*LOC +b*CC+c

M3: ERR_EST = a*RM+b*CC+c

For these models, using the same dataset, the estimations are obtained:

M1: ERR_EST = 0.00323*LOC+ 0.00783*RM+ 5.2874

SS=178,91 $R^2_{adjusted}$ =0,7202

M2: ERR_EST = 0,0003797*LOC +0,00961*CC+5,27064

SS=175,70 $R^2_{adjusted}$ =0,725

M3: ERR_EST = 0,0209*RM-0,00301*CC+5,4132

SS=202,725 $R^2_{adjusted}$ =0,683

The process continues with removing combinations of two variables from the initial model, obtaining the model structures:

M4: ERR_EST = a*LOC + b

M5: ERR_EST = a*RM+ b

M6: ERR_EST = a*CC+b.

After estimating the coefficients the models are:

M4: ERR_EST = 0,005199*LOC + 5.3547 SS=189,19 $R^2_{adjusted}$ =0,7121

M5: ERR_EST = 0,01905*RM+ 5,4153 SS=203,1 $R^2_{adjusted}$ =0,691

M6: ERR_EST = 0,02844*CC+5,6704 SS=247,26 $R^2_{adjusted}$ =0,623

The data obtained from the models built by elimination of variables is presented in table 1. For comparison, the first line includes the initial model.

Table no. 1. Obtained model performance indicators

Model	SS	R
M0	175,70	0,7174
M1	178,91	0,7202
M2	175,70	0,725
M3	202,725	0, 683
M4	189,19	0,7121
M5	203,1	0,691
M6	247,26	0,623

Listing in ascending order by SS, the models appear as in table 2.

Table no.2 . Ordered model list ascending by SS

Model	SS	R
M0	175,70	0,7174
M2	175,70	0,725
M1	178,91	0,7202
M4	189,19	0,7121
M3	202,725	0, 683
M5	203,1	0,691
M6	247,26	0,623

The initial model is ignored and M_2 is chosen as refined model because it has the smallest SS, which is a criterion to be minimized.

It is a fact that the coefficient of determination that shows the quality of the regression has a greater significance than the SS

indicator. That is why the list of generated models is also ordered descending by this $R^2_{adjusted}$ indicator, as shown in table 3.

Table no. 3. Ordered model list descending by $R^2_{adjusted}$

Model	SS	R
M2	175,70	0,725
M1	178,91	0,7202
M0	175,70	0,7174
M4	189,19	0,7121
M5	203,1	0,691
M3	202,725	0,683
M6	247,26	0,623

It is observed that the model M_2 keeps being the first on the list. It is a proof that refinement is necessary. The smaller SS of model M_0 than the SS value of M_2 which has a greater $R2_{adjusted}$ value than M_0, is explained by a frequent phenomenon met in statistics. When adding many variables to a model, due to accidental correlations between the newly added independent variables and the dependent variable, the value of SS decreases, without raising the degree in which that model explains the studied phenomenon.

Using the complexity decrease refinement method, an initial model structure containing power terms is considered:

M_a: ERR_EST = a * LOC ^{b} + c * CC ^{d} + e

Through coefficient estimation using a least squares method, the model becomes:

M_a: ERR_EST = $-2.034 * LOC^{-1.38}+1.143*CC^{0.434}+1.8$

The coefficient of determination for this model is $R^2 = 0.8783$.

Proceeding to refinement, the power terms are replaced by variables, obtaining the model structure:

M_b: ERR_EST = a*LOC +b*CC+c

which is the same with M_2 with the same characteristics as described above. M_2 is given as refined model also by this method.

For the dataset JM1 presented in the next chapter the model structure obtained using gene expression programming is:

ERR = $(c\ (LOC^{1/2}+c^{1/2})^{-1}+(LOC+LOC*\ln LOC)^{1/2})^{1/2}$

where c is a constant.

The expression complexity is high. If starting hypothesis are large with respect to the number of variables the complexity of the generated formula is high, further refinement being necessary.

Model refinement through genetic algorithms needs special methods, of which, those based on artificial intelligence give edifying results in decreasing model complexity. For those there is the *Intelligent Model Refinement* specialized module in the software product used for refinement.

The generalization of these results is based on the analysis of LTSCPP source code collection. Those sources are developed under imposed homogeneity conditions , applying all the refinement methods implemented in the software product SoftRefine with emphasis on advantages and disadvantages of these methods.

8. REFINED MODELS VALIDATION

Using the results obtained from the refinement process, refined indicators are computed for a new set of software modules, solving a problem from the same class. The estimated levels are compared with the effective ones.

Considering M_1, M_2, ..., M_n models used in the refinement process via variable elimination method. Estimated levels are computed and compared to the recorded real levels. If for the first V models, the differences between real values and the recorded values are acceptable for K of them, if the fraction $q=K/V \in [0.78;1]$ the method is validated.

The model obtained through a validated technique is further object to a validation process, using the new SD_1, SD_2, …, SD_h datasets. If there are small differences between the recorded values and those estimated, the refined model is validated.

For the M_2 model, a new dataset is used, a sample of 35 records extracted from CM1 public dataset also from [14]. After estimating the error count, the correlation between the recorded values and the estimated ones is 0,8204 which shows the model is usable outside the initial dataset.

Model validation corresponds to a process the refinement method is validated in the first hand.

9. CONCLUSIONS

The sets of studied programs should be extended and metrics for them should be automatically recorded The refinement process has a cyclic character. It must be reapplied periodically as the phenomena are dynamic and changes occur within the relations between dependent variables and independent variables, new factors influencing the phenomenon appear, other factors cease to affect the phenomenon.

The software instruments used for refinement such as refinement modules and model generators should be included into a modelbase. The modelbase is a complex structure grouping datasets, models and modeling instruments such as modules for refinement and model generators.

Refined models should be simple, easy to use and interpret, the variables have automatically recorded values and must be connected to computed ranges of quality indicators. If the indicator ranges from 0 to 1, and the value of the indicator for the current model is below a certain value, the model must be abandoned.

Refinement methods are chosen according to their fitness for the solved problem. First the refinement method is validated, next, the refined model resulting from the method is also validated.

Software solutions help the automation of refinement processes and model generators.

REFERENCES

[1] Candida Ferreira, **Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence**, Springer May 2006, 2nd Edition, ISBN 3540327967

[2] Adrian Vişoiu - **Performance Criteria for Software Metrics Model Refinement**, Journal of Applied Quantitative Methods, Volume 2, Issue 1, March 30, 2007

[3] Ion Ivan, Adrian Vişoiu - **IT Project metrics**, Journal of Applied Quantitative Methods, Volume 2, Issue 3 - September 30, 2007

[4] Ion Ivan, Gheorghe Nosca, Marius Popa - **Managementul calitatii aplicatiilor informatice**. Editura ASE, Bucuresti, 2006.

[5] Ion Ivan, Cătălin Boja - **Managementul calităţii proiectelor** *TIC*, Editura ASE, Bucureşti, 2005, ISBN 973-594-558-4

[6] Ion IVAN, Adrian Vişoiu - **Baza de modele economice**, Editura ASE, Bucureşti, 2005, ISBN 973-594-571-1

[7] Adrian Vişoiu, Ion Ivan - **Rafinarea metricilor software**, Economistul, supliment Economie teoretică si aplicativă, nr.1947(2973), 29 august 2005

[8] Adrian Vişoiu, Gabriel Garais - **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

[9] Cătălin Boja, Ion Ivan - **Metode statistice în analiza software**, Editura ASE, Bucureşti, 2004

[10] Pankaj JALOTE: **Software Project Management in Practice**, Addison Wesley, 2002
[11] Ion Ivan, Doru UNGUREANU – **Project Complexity**, INFOREC Publishing House, Bucharest 2002
[12] Cristian Toma, Ion Ivan, Marius Popa, Cătălin Boja: **Data Metrics Properties**, Proceedings of International Symposium October 22-23, 2004, Iaşi, Romania, pg. 45-56
[13] Ion Ivan, Mihai Popescu: **Metrici software**, Editura Inforec, Bucureşti, 1999
[14] http://mdp.ivv.nasa.gov/ **Metrics Data Program**