Mathematica
Balkanica

New Series Vol. 18, 2004, Fasc. 1-2

Software Engineering Models and Their Evaluation

Nelly Maneva

In any field of interest the researcher can create a simplified version of reality so as to identify all significant objects and their relationships. Software engineering (SE) follows the same approach. There are several models for any type of SE objects, involved in software development and use - products, processes and resources. Different models meet different goals and it is no easy for common software people to select the most appropriate model for their particular theoretical or practical tasks. That is why any attempt to develop a technique for evaluation and comparative analysis of SE models should be appreciated.

The paper describes the role of SE models and some possible benefits from their use. The basic characteristics and requirements for models are identified. The main part of the work presents the proposed constructive procedure for model evaluation. It is based on a general scheme for modeling, describing three main objects (Reality, Model, User) and their relationship: Applicability (between Reality and Model), Utility (between Model and User) and Validity (characterizing the Model itself). We develop a hierarchical quality model representing quality factors, criteria and metrics for their evaluation. A feasible procedure for total quality evaluation has been described. In conclusion the generality of the procedure is discussed and some ideas for further research are shared.

AMS Subj. Classification: 68N30

Key Words: modeling, SE models, model quality evaluation

1. Modeling and models

Modeling is a very powerful technique from a cognitive point of view. When we study a piece of reality we create a piece of "virtual" reality, called a model. Usually the model allows us to reveal and understand the essence and the underlying features of the real system or process so as to achieve some goals, defined in advance (to evaluate, to make a decision, trying to keep things under control, to assess a risk involved by the evolution and use of the system, etc.).

The word model has a variety of everyday meanings, but further on we are going to stick to the following general definition:

Model is a simplified representation of a real entity.

There are three key ideas in this definition:

a) The correspondence between the real objects and their "images" in the model is obligatory;

b) The model is an abstraction. It is not an identical copy, but a simplified version of it, i.e. neither a precise nor a complete version. Once created, a model has its own existence, which can be somewhat different from those of the original entity;

c) The model is a representation and therefore we need a special language for it. To the same pair (real entity, model) we can apply different descriptions - some, which are true both for the real entity and its image and others, which are true only for the real entity or only for its model.

There is a great number of models and the common user needs a classification scheme helping him to be oriented when he/she tries to understand and/or to apply the models.

In [4] three kinds of models have been distinguished:

- Iconic - reproducing the appearance of the real entity;
- Analogic - reproducing the content and structure of the real entity;
- Analytic - a description that expresses the researcher's analysis of the relevant part of reality.

We suggest a few other criteria for model classification: the application area they represent, the scope (complete or partial), purpose of model creation (to simulate, to study, to assess) and the level of abstraction used for their representation (formal or informal models).

2. Software engineering models and their evaluation

According to the IEEE definition, Software Engineering (SE) is the application of a systematic, disciplined and quantitative approach to the development, operation and maintenance of software. Regardless of the application domain and peculiarities of the software to be engineered, we have to clarify what we want to create and how we can do it in an effective and efficient (with less resources) way. Therefore we need models for any type of SE objects: products, processes and resources. But for none of them a unified, standard and accepted by all SE model exists. The main reason is that each model is created to meet a specific goal. It is constructed in a specific environment and uses different resources. For example, there are more than 20 models of Software Life Cycle, both chronological and functional. Even for a qualified researcher it is a difficult task to decide which model will be most appropriate for a given software company. It is necessary to perform a comparative analysis on a set of similar models so as to select the best one according to some preliminary stated criteria.

There is another, cognitive reason for the great number of models. Both products and processes are very sophisticated and it is not easy to create a perfect model in one step. Usually we create a sequence of models, following a step-wise procedure of refinement. In [2] this process has been described in the following way: "Year by year we devise more precise instruments with which to observe nature with more fineness. And when we look at the observations, we are discomfited to see that they are still fuzzy, and we feel that they are as uncertain as ever. We seem to be running after a goal which lurches away from us to infinity every time we come within sight of it." Thus it is necessary to accomplish a continuous process of assessing the consecutive SE models so as to achieve the proactive search for improvement.

3. Our approach to model evaluation

There are several difficulties in assessing model quality. First, we need a precise definition of model quality as well as a way to derive quantitative measurements of models for objective analysis. Second, the model is an abstraction with no physical presence. Users must be able to define their own quality requirements, bearing in mind the goal of evaluation, application area and available resources. So we need an evaluation procedure, satisfying the following requirements:

a) To be general enough to be applied for any SE model;

b) To be adjustable in order to meet different user demands;

c) To allow at least two levels of complexity and cost;

d) To be feasible and well-defined so as to be automated.

This work describes our attempt to create such a procedure.

3.1. A framework for model quality measurement

Our suggestion is to use the structure of the framework for the measurement of the software quality [2] and to redefine its content to make it appropriate to model evaluation.

At the top of the hierarchy is the total quality. The first level comprises some user-oriented attributes, called factors. The second level describes a number of criteria, which are model-oriented attributes, providing quality. The third level represents metrics, which gives quantitative measures of the criteria (see Figure 2).

To derive a general formula for evaluating quality, we apply a simplification of the MECCA (Multi-Element Component Comparison and Analysis) method described in [1]. We will describe briefly this method. Let's use the following notation:

F - number of factors

f_j - the weight (coefficient of importance) of the j factor

k - total number of criteria

k_j - number of criteria affecting factor j

Next we define

$\delta_{ij} = 1$, if criterion i affects factor j

$\delta_{ij} = 0$, if criterion i does not affect factor j

Let's $a_{ij} = \frac{\delta_{ij}.f_j}{k_j}$. Then

$$\sum_{j=1}^{F} f_j = 1; \sum_{i=1}^{k} a_{ij} = f_j; \sum_{j=1}^{F}\sum_{i=1}^{k} a_{ij} = 1$$

hold.

In case we manage to obtain a measure m_i for each criterion, then the total quality will be

$$Q = \sum_{j=1}^{F}\sum_{i=1}^{k} m_i a_{ij}$$

.

In order to apply the framework, we have to accomplish the following procedure:

Step 1. Define the goal of model assessment. The goal can be:

a) To characterize - to gain understanding;

b) To evaluate - to determine status with respect to expectations;

c) To predict so that we can plan what to do with the model;

d) To improve the model. When we obtain quantitative information, we can identify some drawbacks and model characteristics, which we can try to improve.

Step 2. According to the goal, to define the corresponding hierarchical structure - which elements will be considered at the different levels and what are the relationships between them. This should be done top-down, ensuring at least one element at each level. In order to have two different levels of evaluation complexity, we can skip the criteria level and apply directly some metrics to obtain factor measures. This evaluation will be more simple and cheaper.

Step 3. Apply the defined set of metrics in order to obtain a measure for each criterion (or for each factor in the simple version).

Step 4. Calculate the total model quality, applying the MECCA method.

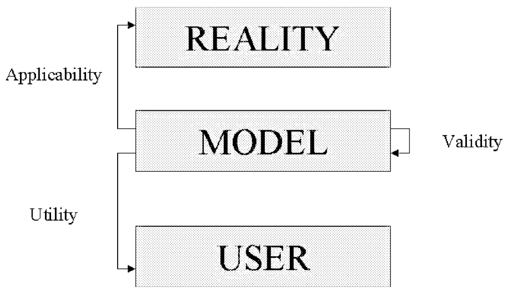

Figure 1:

3.2. Factors and criteria in the framework for model quality measurement

To illustrate the feasibility of the proposed framework, we will define the content of a general framework, which can be further adjusted.

To define the factors, we consider the general scheme for modeling. It represents three main objects: Reality, Model and User, and their relationships: Applicability (between Reality and Model), Utility (between Model and User) and Validity (characterizing the Model itself):

We will define their meaning as factors and will try to define a number of criteria for each of them i.e. model-oriented attributes. In order to do this we select and reformulate some software quality characteristics, proposed in [3,7]. The example tree structure can be seen in the following Figure 2.

Utility concerns the relation between model and user and addresses the question of the possibility to derive conclusions from model's results.

Some possible criteria, affecting this factor:

- Content
  - What does the model represent?
  - Is it detailed enough?
- Fidelity
  - Will different users get similar results?
- Constructiveness

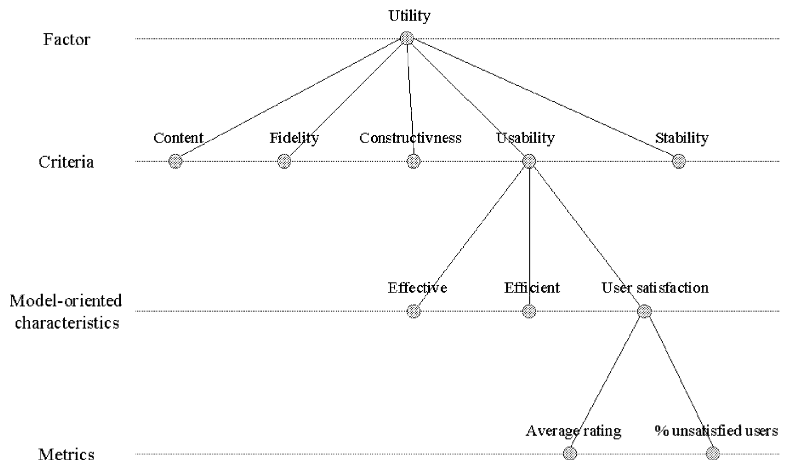

Figure 2:

- Does it help in understanding real entities?
- Are the derived results explainable?

• Stability
- Can the model be manipulated to obtain false results?

• Usability
- Is the model effective?
- Is the model efficient?
- What is the level of user satisfaction?

The second factor should be applicability.

Applicability concerns the relation between the model and the real system and addresses the question of the legality of the use of the model for a given system, running under specific conditions.

Some possible criteria:

• Adequacy
- Is the model correct?
- Is the model complete?
- Is the model consistent?

• Trackability
- Must the change in the real object be accompanied by a change in the model?

- Consistency

  - If there is an order of real objects, do the corresponding models have the same ordering?
- Accuracy

  - Is it possible to define an analog of Hamming's distance between the real object and its model and to apply it in a constructive way?

The third factor should be validity.

Validity concerns the inner capacity of the model to produce results corresponding to the expected ones.

Some possible criteria, affecting this factor:

- Reliability
  - Is the model adequate?
  - Is the model robust?
- Understandability
  - Is the model structured?
  - Is the model concise?
  - Is the model self-descriptive?
- Measurability
  - Is the model assessable?
  - Is the model quantifiable?
- Modifiability
  - Is the model simple?
  - Is the model visible?
  - Is the model modular?
  - Is the model consistent?

Each criterion has to be measured by a metric. The simplest metrics are in the form of checklists [6]. An expert can evaluate the degree of fulfillment of a given attribute, using a defined in advance scale (binary or k-valued). This approach is rather subjective, but it is the most appropriate because of the abstract nature of the evaluated model attributes.

4. Conclusions

This work describes our approach to Model Evaluation. The proposed evaluation procedure satisfies all stated requirements, namely:

a) The procedure is general and can be applied not only to SE models, but (with minor modifications) to any model;

b) The evaluation procedure is flexible. For each model the evaluator can define the structure and the content of the framework used and the significance of the attributes at all levels, ignoring some of them by setting the corresponding weights equal to 0. So the complexity and the cost of the evaluation can be under control.

c) The evaluation is constructive, because it provides a guideline for modelers, who are made aware of what qualities are considered important and therefore should be built in the model. Having a quantitative measure of the model quality, we can apply the comparative analysis to two consecutive versions of the model. So the modeler receives better direction how to proceed and how the modeling is progressing relative to the established goals.

d) The obtained quantitative measure of the model quality can be used for the selection of the best model among a set of models for the same real entity.

e) The procedure is algorithmic and can be easily automated.

Some possible directions for further research are:

- To try to develop a set of metrics for any defined criterion and to validate these metrics.
- To develop a tool automating the described evaluation procedure.
- To verify the approach in a real user environment.

References

[1] T. Bowen, G.B. Wigle, J. Tsai. Specification of software quality attributes, *Software quality evaluation guidebook*, RADC-TR-85-37, vol. **III**, 1985.

[2] J. Cavano, J. McCall. A framework for the measurement of Software Quality, *Proc. of the ACM software quality assurance workshop*, November 1978, 133-139.

[3] ISO/IEC 9126 Software quality characteristics.

[4] M. Jackson. *Software requirements and specification.* Addison-Wesley Publ. Company, 1995.

[5] N. Maneva. An approach to usability assurance. *Proc. of the CompSys-Tech'03*, Sofia, 2003, II.3-1, II.3-5.

[6] R. Pressman. *Software engineering - A practitioner's approach*, Fifth edition, McGraw Hill, 2001.

[7] G. Walters, J. McCale. Software quality metrics for life-cycle cost reduction. *IEEE Transactions*, vol. **R-28**, **3** (1979).

Institute of Mathematics and Informatics - BAS,
Sofia 1113, "Acad. G.Bonchev" St. Bl. 8,
e-mail: neman@gbg.bg, nmaneva@aubg.bg

Received 30.09.2003