Available online at www.sciencedirect.com

ScienceDirect

Procedia Economics and Finance 27 (2015) 669 – 675

Procedia Economics and Finance

www.elsevier.com/locate/procedia

22nd International Economic Conference – IECS 2015 “Economic Prospects in the Context of Growing Global and Regional Interdependencies”, IECS 2015

The Comparison of Software Reliability Assessment Models

Marian Pompiliu Cristescu[a,*], Eduard Alexandru Stoica[a], Laurențiu Vasile Ciovică[b]

[a]*Lucian Blaga University of Sibiu, Faculty of Economic Sciences, 17 Dumbrăvii Street, Sibiu 550324, Romania*
[b]*Alma Mater University of Sibiu, Faculty of Economic Sciences, 22-24 Dumbrăvii Street,, Sibiu, 550324, Romania*

Abstract

The reliability of the software represents one of the most important attributes of software quality, and the estimation of the reliability of the software is a problem hard to solve with accuracy. Nevertheless, in order to manage the quality of the software and of the standard practices in an organization, it is important to achieve an estimation of the reliability as accurate as possible. In the present work there are described the principles and techniques which underlie the estimation of the reliability of the software, starting from the definition of the concepts which express the attributes of software quality. It is taken into account the issue of the estimation of a software part. The presumed objective of the estimation of the reliability consists in the analysis of the risk and of the reliability of the software-based systems. Supposedly, a documented opinion of the expert exists regarding the reliability of the software and an update of the defined estimation of the reliability is tried with the information contained in the records of the operational data.

Keywords: software reliability, software model, estimation, bug/fault, error

1. Introduction

If the failure of a hardware system is owing to the alteration in time of material properties, the wear doesn’t act in the software domain, the failure consisting here in the highlight of a latent error contained in the program (Mihalache, 1995). The failures of the software are treated as random events which are produced throughout a time axis, due to the lack of certainty regarding the exact time of their manifestation, reason for which we refer to the reliability, not

* Corresponding author.
E-mail address: marian.cristescu@ulbsibiu.ro (M.P. Cristescu), eduard.stoica@ulbsibiu.ro (E.A. Stoica), laurentiu.ciovica@gmail.com (L.Ciovică)

doi:10.1016/S2212-5671(15)01047-3

the quality, of the software.

A software system is thus similar from the point of view of the failure process with a hardware system with restoration and it is described by the same ensemble of reliability indicators as the latter.

According to IEEE (ANSI) 982.2 from 1988,(IEEE 1988) standard, the essential elements in defining the software are the following:

- The error. It is a human mistake which has as a result an incorrect program. E.g., the omission of a crucial requirement(task);
- The disorder (fault or bug). This is the result of a human error and represents the internal accident which causes the system not to work as expected.

In the current expression (language), the term error is used both for the act of making a mistake (error), and also for its direct manifestation in the program (fault or bug).

2. Criteria for analysis of software reliability models

Most of the existing models of software reliability enhancement treat time as a continuous variable (either calendar time, hour time or execution time). Nonetheless, there are systems, like stock transaction processing systems, in which reliability should be treated in terms of transactions handled successfully. More than that, there are other systems, like the command software of missiles (space shuttles), in which it is more natural to measure reliability according with the number of launches of missiles (space shuttles) successfully completed. Systems like these require a discreet conception of time, in terms of the number of runs of the software.

The “code coverage” indicator (the actual proportion of code covered by testing) within testing can significantly affect the software reliability estimation: testing may reach saturation, in which case new parts of code don’t actually get to be tested. Thus, the reliability estimations which rely completely on execution time/execution can overestimate the reliability of the program.

There are several ways that can be used to evaluate the value of a model (Imol 1983):

- Predictive validity. This one represents the capacity of a model to make predictions regarding the future failure behavior, throughout either the testing phase or the operation phase, predictions obtained from the current failure behavior and from the one in the past in the respective phase. This aspect can be, further on, divided in:
    - Accuracy (precision level), measured by prudential likelihood
    - Inclination, measured by the U-diagram
    - Tendency, or the systematic change of the inclination, from small values towards big values of the failure time, attribute measured by the Y-graphic
    - Noise, measured by the relative change occurred in the forecast rate of failure
- Capability. The capacity of the model to estimate with sizes of a satisfactory precision that the managers, engineers and the users of the software need in planning and administrating the software development projects, or in the control of the changes occurred in the operational software systems. These sizes include, e.g., the current reliability, the anticipated date for achieving the reliability objective, and also the necessary cost to achieve that objective.
- The quality of the hypotheses. If a hypothesis made by the model can be tested, then this attribute refers to the degree to which the model is accepted by the actual data; however, of the testing of a hypothesis is not possible, the attribute refers to the plausibility from the point of view of the logical consistency and of the experience gained in software engineering. Also, the clarity and the explanation of a model hypothesis need to be judged.
- Applicability implies the usefulness of the model within the various software products (size, structure, functions), the diverse development environment, the different operational environments and the different phases of the life cycle.
- Simplicity. Consists in the simplicity and ease of data collection, which is necessary in view of the customization of the model. Also, it refers to the conceptual simplicity, in that the audience of the model (software engineers, project managers, reliability specialists, officials) can understand the nature of the model and its hypotheses, so as to determine its applicability to a specific problem, and also the extent to which the model deviates from reality, in the intended application. More than that, the attribute also refers to the

simplicity of implementation, in such a way that the model can become a practical tool of management and software engineering.

- Ease of measuring the parameters (Lyni 1992). This attribute takes into account the number of parameters required by the model and the degree of difficulty encountered in their estimation.
- Insensitivity to noise (Lyni 1992). Attribute that refers to the capacity of a model to carry out accurate predictions, even when the failure data are incomplete or contain uncertainties.

Goel (Goel 1985) proposes that, before adopting a model, the errors should be studied in their evolution, depending on calendar time, execution time or number of runs.

Goel (Goel 1985) proposes that, before adapting a model, its evolution errors have to be studied, according to the calendar time, execution time or the running number. Thus, significant variables can be selected and an orientation is realized towards a particular class of models. After estimating the model parameters selected in the first phase, an accordance test is performed according to the adopted data model (Mihalache 1995) (figure 1). If the model is accepted, the predictive measures will be calculated and decisions regarding the completion or continuation of testing will be taken. According to figure 1, the essential step regarding the model adopting is the accordance and not the comparison between the predicted and observed values.

The techniques for evaluating a model, revised in this section, like the U-chart, Bayes factor and the prudential likelihood report, work only when (positive) event instances exist. In case the software defects, then at least some of the software failures should occur and be observed. In the case of reliable programmable devices that are in operation, this assumption is often unrealistic. Therefore, these methods are better suited to assessing the software's reliability growth models, which addresses primarily the development step of the software's life cycle.

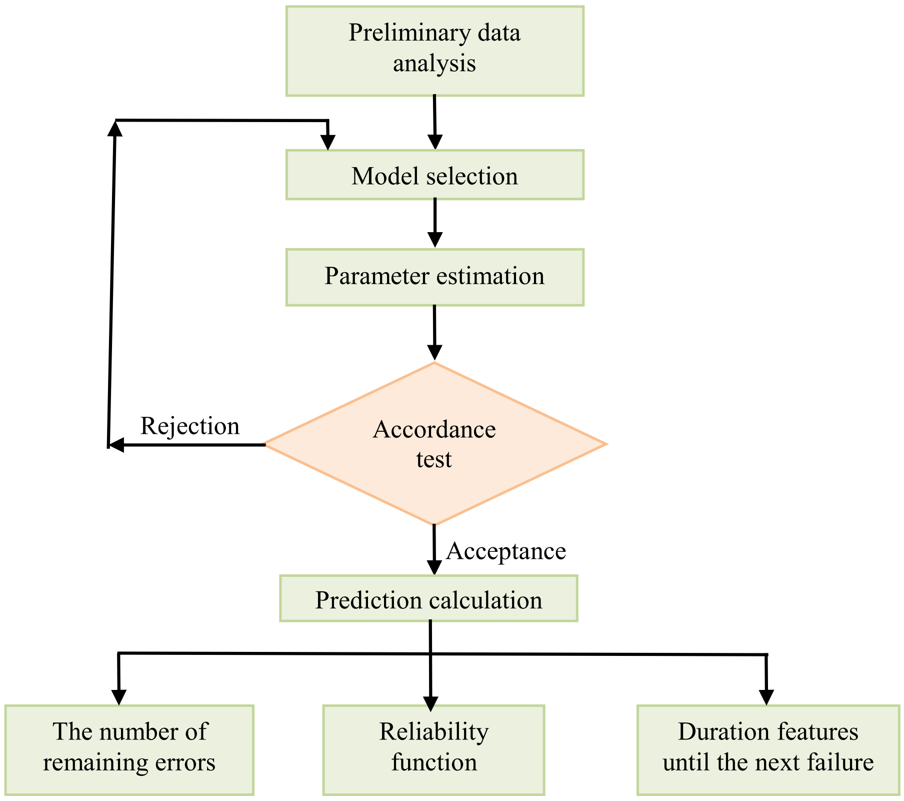

Fig. 1. Adopting a model of reliability programs

There are other methods for model validation and comparison, such as the cross-validated likelihood (Baeb 1998).

2.1 U-chart

The U-chart (Brocklehurst 1992) is used to determine whether the postulated cumulative distribution function, is close to the true distribution, (given by the observations). It is known that the random variable has a uniform distribution on the interval. Thus, if the achievements (e.g., time of failure) are observed, and are calculated, then must be an achievement of a uniform random variable. Any deviation from uniformity indicates the deviation of from.

To find the deviations (if they exist) it is written the sampled distribution function of the transformed observations. The chart is a staircase function, consisting of numbers from the interval. Then it is written an increasing staircase function, every height step being written on every value from the abscissa.

The closer this graphic is to the unitary slope line, the closer is to. On the other hand, any systematic deviation from the unitary slope indicates a probability distribution haziness.

This thing can be developed in an operational measure, through finding the Kolmogorov distance (the absolute maximum vertical deviation) between the perfect prediction line of slope 1 and the effective diagram (Lyni 1992).

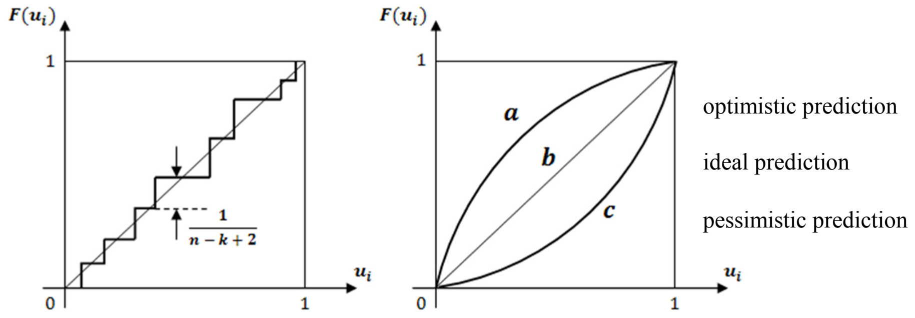

Fig. 2. The U-chart, discreet (leftside) and continuous (rightside)

2.2 Y-chart

The Y-chart measures the degree of coherence of the propensity's model; for example, a model can be at the beginning too pessimistic, and sometimes too optimistic, regarding the number of software defects.

This is the result of the $u_i = \hat{F}(t_i)$ transformation sequence, defined in the previous section, as follows:

$$x_i = -\ln(1-u_i) \tag{1}$$

and

$$y_i = \frac{\sum_{j=1}^{i} x_j}{\sum_{j=1}^{M} x_j}, \tag{2}$$

where $i \leq M$.

This thing can be done in an operational measure by calculating the Kolmogorov distance, $\max_i \left(| x_i - y_i | \right)$, between the variables that are defined above.

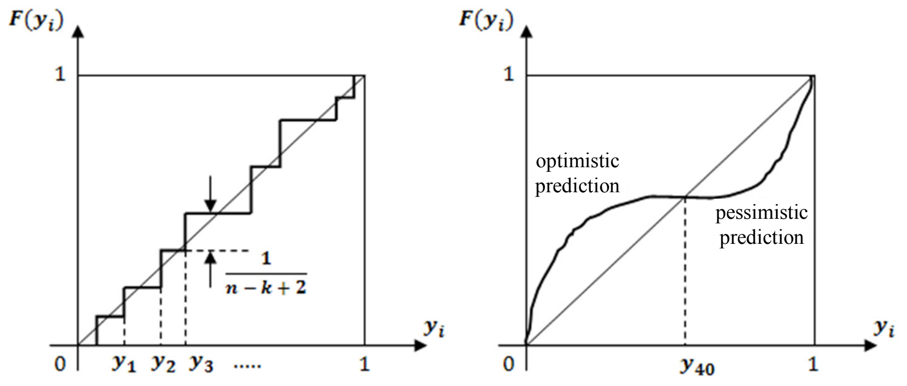

Fig. 3. The *Y*-chart, discreet (left side) and continuous (right-side)

2.3 Bayes factor

The Bayes factor (Gcsr 2000) represents the formal comparison criteria of the Bayesian models. There are two competing models H_1, respectively H_2. The Bayes factor consists of the report between the marginal probabilities of the two compared models, as follows:

$$BF(H_1,H_2)=\frac{p(t_1,...,t_m\mid H_1)}{p(t_1,...,t_m\mid H_2)}, \tag{3}$$

where $$p(t_1,...,t_m\mid H_i)=\int p(t_1,...,t_m\mid\theta_i,H_i)\cdot p(\theta_i\mid H_i)\cdot d\theta_i\ . \tag{4}$$

The Bayes factor calculation might require demanding calculation resources. For a revision regarding the marginal probability estimation based on the a posteriori distribution charts, please see the references (DiCiccio 1997).

2.4. The prudential likelihood and the prudential likelihood ratio

The prudential likelihood measures the accuracy degree of a model (Lyni 1992). It is marked with $f_A(t)$ the probability density function of data, provided by model *A*. Moreover, there are the $t_1,...,t_m$ observed manifestations of the defects (or of every phenomenon's we wish to shape/model).

A model's prudential likelihood is as follows:

$$PL_A=\prod_{j=1}^{M} f_A(t_j) \tag{5}$$

This factors product is, usually, close to zero, and a more obvious indicator is obtained by the prudential likelihood logarithm.

The prudential likelihood report (Brocklehurst 1992) compares the capacities of two models to predict a certain set of data. It is marked with $f_A(t)$ and $f_B(t)$ the probability density function of data, provided by model *A*, respectively *B*. The prudential likelihood report, PLR_i^{AB} is defined as:

$$PLR_i^{AB}=\prod_{j=1}^{i}\frac{f_A(t_j)}{f_B(t_j)} \tag{6}$$

This ratio must increase once with the observation number incensement, if model A is superior to B, and it must decrease otherwise.

It is easily to see that the prudential likelihood ratio represents a simplified version of the Bayes factor: if the observations are independent and the probability domain is a discreet one, then these ones coincide. Anyway, in most cases, the prudential likelihood ratio is easy to calculate.

3. Conclusions

In literature a multitude of models were proposed, but each one with their own disadvantages, some of them being common in most cases. One of the revised models problems is that none doesn't allow the analysis based on the non-existent defect data, *i.e.*, a history of software usage that has in its operational state a period of time with no detected failure / defect. Another common problem of these models is that they support only average reliability requirements. There is no solution to this problem.

The existing software reliability models don't take into consideration the application complexity or the test coverage degree (the proportion of every possible entries that were effectively tested) (Whittaker 2000).

Farther, things are complicated again, because the software under investigation doesn't ever run on its own, as it is a component of a system, composed of hardware, operating system, interfaces (*e.g.*, device drivers and communication interfaces), and, possibly, other programs (*e.g.*, virtual environments).

Most revised models share, likewise, the feature through they were developed to model the reliability increase attributes. This thing is adequate when the program was developed by a well-disciplined team and when reports regarding the development stage defects or, at least, statistics of defects are available. Nevertheless, from the software user's point of, it is more realistic to assume that only the operational data recordings are available.

Some recommendations can be made regarding the software reliability models application:

- To the models that take into consideration the software architecture, the software complexity, the test coverage degree, a validation and verification conduct as well as the structured opinion of experts, should be given priority.
- In the applications that require a high dependence degree, the software reliability models should be used only in relation with other methods that ensures a satisfactory quality level – otherwise, the testing dimensions will increase exaggeratedly. Among such methods, formal methods, software inspections and revisions, statistic-code analysis, software systematic testing etc. are also part of.
- It would seem that the Bayes models present higher perspectives than the traditional models. An advantage of the Bayes approaches is that it allows the integration of different types of information, including human (judgment) appreciation.
- Any software developer should not rely on a single model, but rather choose a set of models, whose results can be combined in a way or another.

Acknowledgment

This work was supported by the strategic grant POSDRU/159/1.5/S/133255, Project ID 133255 (2014), co-financed by the European Social Fund within the Sectorial Operational Program Human Resources Development 2007-2013.

References

Basu S. and Ebrahimi N., "Estimating the Number of Undetected Errors: Bayesian Model Selection" in Proceedings of the The Ninth International Symposium on Software Reliability Engineering, 1998, p. 22;

Brocklehurst S. and Littlewood B., "New Ways to Get Accurate Reliability Measures" IEEE Software Transaction, vol. 9, no. 4, pp. 34-42, 1992;

DiCiccio T. J., Kass R. E., Raftery A., and Wasserman L., "Computing Bayes Factors by Combining Simulation and Asymptotic Approximations" Journal of the American Statistical Association, vol. 92, no. 439, pp. 903-915, 1997;

Gelman A. B., Carlin J. S., Stern H. S., and Rubin D. B., „Bayesian data analysis", Boca Raton: Chapman and Hall, 2000;

Goel A. L., "Software Reliability Models: Assumptions, Limitations, and Applicability" IEEE Transactions on Software Engineering, vol. 11, no. 12, pp. 1411-1423, 1985;

Iannino A., Musa J. D., Okumoto K., and Littlewood B., "Criteria for software reliability model comparisons" ACM SIGSOFT Software Engineering Notes, vol. 8, no. 3, pp. 12-16, 1983;

Lyu M. R. and Nikora A., "Applying Reliability Models More Effectively" IEEE Software Transactions., vol. 9, no. 4, pp. 43-52, 1992;

Mihalache A., „Când calculatoarele greşesc: fiabilitatea sistemelor de programe (software)", Bucureşti: Editura didactică şi pedagogică, 1995;

Whittaker J. A. and Voas J., “Toward a More Reliable Theory of Software Reliability” Computer, vol. 33, no. 12, pp. 36-42, 2000;
IEEE (ANSI) Standard 982.2/1988. Software Reliability Terminology.