"Dimitrie Cantemir" Christian University
Knowledge Horizons - Economics
Volume 11, No. 1, pp. 82 – 92
P-ISSN: 2069-0932, E-ISSN: 2066-1061
© 2019 Pro Universitaria
www.orizonturi.ucdc.ro

THE REENGINEERING PROCESS OF THE ECONOMIC SOFTWARE RELIABILITY

Marian Pompiliu CRISTESCU

"Lucian Blaga" University of Sibiu, Romania, Email: marian.cristescu@ulbsibiu.ro

Abstract *In the study of software, reliability is important to know the difference between error, defect and defection. This three terms are often confounds and, are often used with the same sense but they have different meanings. Usually, a defect occurs when system behavior moves off from the demands. The defect can be studied also by prism of failure operations, so is defined as an incomplete operation. The defect is the cause of lot of defections because the variety of inputs makes that the programs system behave in a different way. Defect cause the defection. Because software depreciates in other way than hardware, the software reliability stay constant in time if we don't operate changes in the source code or at the environment conditions level, inclusive at the user behavior. However, if every time when a defection was rectified, and the defect that was the main cause is eliminate, then reliability will increase in ratio with time. This is a usual situation for the probation phase.*

Key words:
software reliability, software cost, error, failure
JEL Codes:
O31
O32
O33

1. INTRODUCTION

The software quality contains a set of different properties. One of the most important is represent by the reliability. A straight approach of the reliability supposes its orientation rather to the user than to the software development process. This approach derives from the users' point of view, which determinates an easier understanding of the reliability by the clients. Also, refers more at the execution than to the design, which makes it more dynamic than statically. One advantage is the fact that reliability is being calculate for the both components: hardware and software. We can also calculate the reliability to the entire system.

The software reliability engineering is a part of the software engineering that concentrates on the quality property named reliability. It implies all the steps from the development process in order to identify the demand regarding to the reliability and in order to control if we find them in the final product.

The software reliability management is definite as the software reliability improvement process that accentuate on warning, detecting and elimination of the software errors and on using the metrics in order to maximize the reliability by its project compulsions resources, the cost and the performances. For reliability maximization we must do:

- we need to prevent errors;

- to detect and eliminate errors;
- measurements for increasing reliability, for define and for using specific metrics to sustain first two activities.

Reliability is the quality product and quality can be measured. In order to be effective, the measure must be a part of the management activity. The measures help in order to achieve the fundamental objectives of the management, concerning prevision, progress, and processes improvement.

2. THE SOFTWARE RELIABILITY - PRELIMINARY CONSIDERATIONS

If the objective is to calculate global reliability, we try to prevent and to investigate errors since the first step of software development.

The error represent an omission of the programmer wherefrom results the defection. Errors are classified as:

- *Impermanent errors* - are errors that manifest by a temporary bad function of a component- directive, module, or program- but not by a permanent defection; most of the errors from the programs system are impermanent;
- *Permanent errors* - occur at a certain moment and maintains until the damages are repaired.

Because software depreciates in other way than hardware, the software reliability stay constant in time if we don't operate changes in the source code or at the environment conditions level, inclusive at the user behavior. However, if every time when a defection was rectified, and the defect that was the main cause is eliminate, then reliability will increase in ratio with time. This is a usual situation for the probation phase.

The hazard rate of a component is noted with $\lambda(t)$. Recent researches in reliability domain talk about the risk rate as failure rate $r(t)$ or failure intensity $\lambda(t)$. The fact that the risk rate of the program $z(\Delta t/ti-1)$ is equal with the failure intensity at the $[ti-1+\Delta t)$ for the reliability increasing models of Poisson type, contributes at this confusion.

To increase software reliability by preventing errors, we must analyze the demands and the probation plans, in order to assure the demands probation. In addition, we must pay attention to the software maintenance during the utilization time. This activity implies assistance from the programmer's side.

The fundamental factors that influence the software reliability are the redundancy, the time, the input space and the operational profile:

Redundancy - is used to build reliable programs system; there are two types: spatial and temporal;

- *the special redundancy* - use more components than is necessary , to implement a programs system; when redundancy is big, there are detected and tolerated more errors;

- *temporal redundancy* - consists in the same instructions set, in order to make the same thing in repetitive way; after that the results are compared between them.

Time- from the point of view of the quantitative expression, the reliability is described

often with some time intervals. There are different types of the time units:

- *calendar time* : is useful in order to express reliability congenial with the calendar time, because it offers to the managers and to persons that develop software the chance to see the date when the system attains the objectives of reliability;

- *execution time*: is used by models to evaluate reliability; the reason consists in the fact that these models are higher than the models that use the calendar time. To be able to express reliability by calendar time, the models convert the execution time in calendar time in an afterwards stage. The models used to calculate the reliability are based on the defects which appears in a chance way. Because a defect appears only during the execution process, is important to use the calendar time in the reliability calculations.

The software reliability as a quality attribute: the measures and the concepts used to study hardware reliability apply fractionally at software, because of the differences that exist between these two components.

The nature and the errors comportment in programs systems – when a defection appears at a component, these stop constrained its execution. Also now, an exit message that contains the reason of stop execution is transmitted to all the interlinked components.

If the system is fault tolerance, exist the possibility that its execution to continue when the fundamental functions are not affected. In this case, the system must inform the user about the error occurred and about the way the execution spreader.

Table 1. The errors of the software systems

Appearance conditions	*In normal conditions/in abnormal conditions*
Origin	From design, execution, exploitation
The possibility to eliminate the defection cause	Eliminable, not eliminable
The possibility to use afterwards	Total, partial
Mode to eliminate the defections	The conversion of the defect component, the corrective maintenance
Frequency of defection appearance	Unique, systematic
The intercession complexity in order to eliminate defections	Simple, complex
Consequences	Dangerous, major, not dangerous, minor
Mode of fault localization	Visible, secretly
The defection level dependence	Dependent, independent
Properties modification	Sudden, lent

Is very semnificative an analyze on the comportment of the hardware defects and of the software errors. The software errors depend by the input errors. The fact that the errors occurs at a certain moment, in an unique input sequence elaborated in that moment, confirm their dependence on input dates. Hardware defects depends by time because all devices have a medium endurance, thereupon the defection intensity increases very fast.

Causes of the errors appearance – the cause of the errors that appears in a programs system are the mistakes made in different translations which takes place in its achievement

process; the mistakes are human actions, which generate undesirable results presented in every stage from the achievement process, since the define of demands and until the functional probation of the programs system.

3. SIMULATION THE RELIABILITY OF THE ECONOMIC SYSTEMS

A financial and accounting system offers a variety of functions; therefore, it contains a big amount of components. Evaluating the reliability of the system is done by analyzing the reliability of it's` components. In this process, the structure scheme must be taken into consideration. The components of the system are represented in figures 1 and 3. In a structural reliability scheme, these are connected in series or parallel.

Figure 1. A series structural scheme from a financial and accounting system

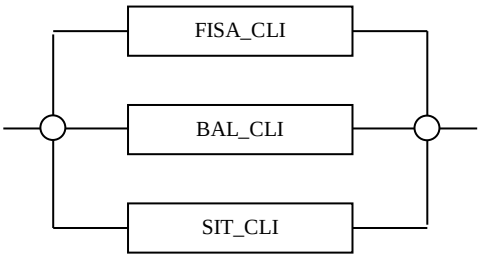


Figure 2. A parallel structural scheme from a financial and accounting system

In order to assure a normal functioning of a performing step, in the series scheme all components have to be working, but in the case of the parallel scheme only one component must be working.

The numerical simulation of the reliability of the financial and accounting system has been achieved through the following algorithm:

- for n in series connected components, each of them having the reliability R, n evenly distributed numbers between 0 and 1 are generated. If all n numbers are smaller or equal to R, the system is functioning properly;
- for n in parallel connected components, each with the reliability R, n evenly distributed numbers between 0 and 1 numbers are generated. If only one of the n numbers is bigger or equal to R, the system is functioning properly.

Estimating the global reliability of the system is made by repeating these numerical simulations for a number of times equal to the number of performing steps allowed. Because in the case of the financial and accounting system this number is high, the problem has been simplified and only 500 simulations have been performed.

a). The numerical simulating programe of performing components connected in series

```
n = [150,300]; % number of simulations
R = 0,8; % reliability of the components
m = 3; % number of series connected components
for j = 1 : length(n)
        k = 0;
                for I = 1 : n(j)
                        x = row(1,m);
                        if all(x<=R)
                                k = k + 1;
                        else
                end
    end
        f(1,j) = k / n(1,j); % reliability
        end
```

b). The numerical simulating programe of performing components connected in parallel

```
n = [150,300]; % number of simulations
R = 0,8; % reliability of the components
m = 3; % number of parallel connected components

for j = 1 : length(n)
        k = 0;
                for I = 1 : n(j)
                        x = row(1,m);
                        if any(x<=R)
                                k = k + 1;
                        else
                end
    end
        f(1,j) = k / n(1,j); % reliability
        end
```

c). The global simulating programe of 500 performed simulations

```
    n = 500; R1 = 0,8;R2 = 0,92;
F1 = row(n,1);  F2 = row(n,1);
N = length(F); % number of functions
fprintf(The reliability of the system is %3.2f\n',N/n)
```

The first programe takes figure 2 into consideration and uses components with the reliability *R=0,8.* The second treats the case of figure 3 and also uses components with the reliability *R=0,8.* In order to compare the calculated reliability, a number of 150 and 300 simulations have been conducted. The third programe takes into consideration a series structure made out of two components, the first reliability *R1=0,8,* and the second reliability *R2=0,92*.

After the first programe performed, the reliability obtained was:

[0,5200 0,5000].

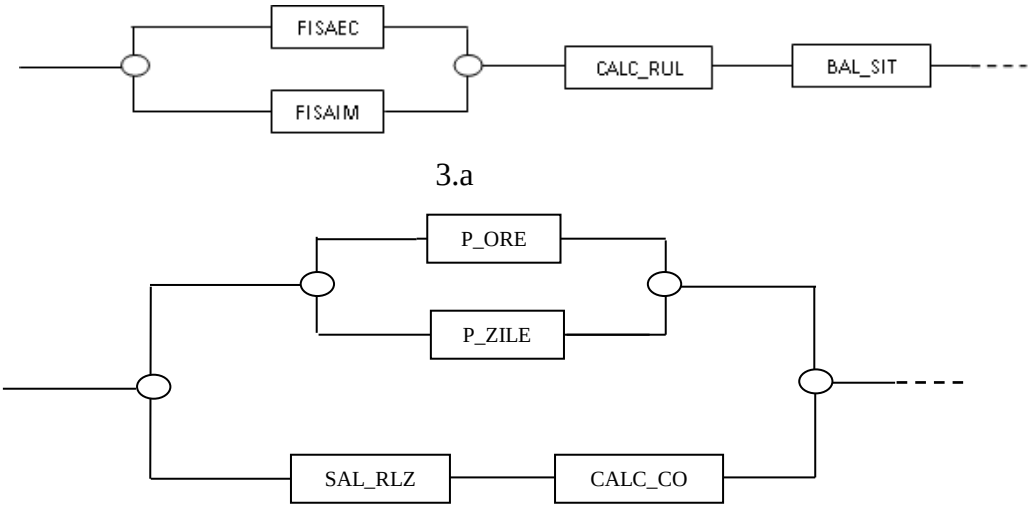
After the second programe performed, these values of the reliability were offered:

[0,9920 0,9960].

After the third programe performed, this value was obtained:

The reliability of the system is 0,74.

In practice, (Cristescu, 2003) it was been discovered that for financial and accounting programs which contain a big number of components, using the series and parallel scheme does not assure a high level of reliability. Therefore, a mixed structure that combines the advantages of both types is used. In figures 3.a and 3.b two specific cases of such mixed structures are presented. These are frequently used for financial and accounting evidences.



3.a

3.b

Figure 3.a, b Mixed structural schemes

The following reliability calculations are used in both cases:

- in the first case, from figure 3.a, the reliability is given by the relationship:

$$R_m = \prod_{i=1}^{n} R_i - \prod_{i=1}^{n} R_i (1 - R_i) \quad (1)$$

- for figure 3.b the reliability is given by the relationship:

$$R_m = 1 - \prod_{i=1}^{n} (1 - R_i) + \prod_{i=1}^{n} R_i (1 - R_i) \quad (2)$$

Because the complexity level of these schemes is very high, the very difficult necessity of simplification the structure function appears. In specialized literature (Szyperski, 2002), (Pham, 2000), (Davis, 2003) different methods of reducing the structure of the function and calculating the reliability of mixed structural schemes are presented. According to the method presented in (Goron, 1997), the components of a financial and accounting system must be grouped regarding to the way they are

situated in the serial or parallel graph and so, we get a primary level of a programming group.

This group is formed by components that are connected in series or parallel. A new group on the next hierarchical scale follows and this procedure is continued until a single series or parallel structure of n levels is formed, where levels of n-1 components are displayed. These methods have a low applicability rate due to a set of assumptions on which it relies and too many calculations.

4. SOFTWARE RELIABILITY ASSESSMENT METHODS

In order to determinate, the reliability increasing ways is necessarily to know the factors, which influences it, to analyze the errors causes and to evaluate quantitative the reliability. The reliability parameter it is a measure which expresses quantitative the reliability or one of its properties (Gandy, 2004). Considering the static character of the reliability result that the reliability parameters are static measurements.

The reliability parameters for program systems estimate its properties. This submit:

- to effectuate reliability computation;
- to fundament the reliability demands;
- to compare reliability for different programs;
- to analyze the influence of different factors on the software systems reliability;
- to choose and to fundament the reliability increasing ways.

The preliminary reliability of the systems it determinates in the design stage when it obtains their architecture-components, interfaces and interactions. To estimate the preliminary reliability we apply the analyses techniques of the systems (Huang, 2000).

4.1. THE SOFTWARE RELIABILITY MODELING

A model is a simple representation of the system or real process comportment or structure. The goal of the modeling process is to reproduce the fundamental relationships in order to realize their clear understanding (Jeske, 2005).

A good model presents next properties:

- it offers good prediction about the errors future comportment;
- it calculates the necessary measures;
- is simple;
- has a high applicability level;
- has as fundaments solid hypothesizes.

The software reliability modelling is a functional representation of the debated system and the better model offers a viable mechanism for reliability estimation.

4.2. CLASSIFICATION OF THE SOFTWARE RELIABILITY MODELS

Most of the specialists (Kapur et al., 2004), which designed and realized models, had validate them theories on the base of actuality dates. The studied mentioned previously are not convincing's because:

- they are based on a small number of experiments;
- the validity of a model and its evaluation criterions are not defined in an unitary way.

These reasons determines an absence of conviction, and this thing represents an important limitation to the users. Therefore, is necessarily to develop some procedures that admit:

- time between the errors;
- the count of the errors;
- the solving of the errors;
- inputs on the main domain.

In the next pages, are presented the most used models for the evaluation of the programs systems in accountancy:

4.2.1. PREDICTION MODELS

These models are applied in the first stages of the systems designing cycle. This models uses collect dates in the development process and product metrics in order to obtain a preliminary level for the defection density. The representative models are:

- RADC model
- the model based on phases.

4.2.2. ASSESSMENT MODELS

The estimative models included in the date's domain use the probation dates for the chosen programs systems according to the probability distributions of the anticipated operational uses. The models included in the time domain use as the fundamental element time, in these variants: the occur time of the errors, the execution time between successive errors, the probation time between successive errors.

The most representative models are:

- the logarithmic model Musa- Okumoto
- the Littlewood- Verrall model.

4.2.3. RELIABILITY INCREASING MODELS

For this type of models, the previous models predict new versions programs system reliability.

The increasing models realize predictions based on the observed number of errors during the test, and the parameters are expressed by the maxim likelihood method or the smaller squares method. They product different results for the same input dates.

The models used frequently in estimation of software increasing reliability are:

- the Goel-Okumoto model;
- the generalized Goel- Okumoto model;
- the model with S profile ;
- the Yamada-Ohba-Osaki model;
- the lately S model;
- the log-logistic model.

5. THE MODERN PROGRAMMING TECHNIQUES TO INCREASE THE RELIABILITY OF ECONOMIC SOFTWARE

The technique of object oriented modelling is a methodology used to develop financial and accounting software by using a collection of predefined techniques and noting conventions. It follows the entire life cycle which contains: analyzing, designing, implementation and testing.

These are followed by the stage in which it is used, when the maintenance and improvements on the system are done, to ensure the reliability needs imposed by the client.

For the development of accounting programming objects, two approaches are practiced: quick prototyping and the development of the entire life cycle. In the quick prototyping a small part of the system is initially developed, after this it is improved through gradual improvements of the specification and implementation, until it becomes robust.

The development methodology of software designed for financials and accountings is firstly characterized by the analyzing and projection steps, whereas the implementation and testing steps rely on the first. The analysis process has as a result a formed model which contains three essential aspects of the system: the objects and the relationships that exist between them, the dynamic flow between the orders and the functional transformation of data, using certain restrictions. Therefore the OMT methodology is bases on three models directed towards the object:

- the object oriented model – describes the static structure of data;
- the dynamic model – describes the temporal relationships of orders;
- the functional model – describes the functional relationships between values.

The programming technique frequently used is chosen on criteria such as error and performance tolerance. In (Kim et al., 2000) it is told that the extension of the traditional library of stopping points is easy to do, so as this one is able to notice more directions from the same process. A multidirectional set library of stopping points, which works at a processing level, must save all directions for a verification point and to restore each of them when it is restarted.

In (Teodorescu et al., 2001) it has been demonstrated that this mechanism of stopping points increases the flexibility and efficiency of the error tolerance schemes. Due to these characteristics it is used in the development of financial and accounting systems, in order to increase the efficiency of the tests and to raise the reliability level.

To exemplify the way this mechanism is used, an accounting programming system which, for error tolerance, uses the distributing algorithm of coming and going – present in (Kim et al., 2000), is taken into consideration.

As a consequence of the existing relationships, the functions of the programming system become interdependent. If one of them fails, the algorithm determines which of the functions is dependent on the one that failed, and these must be performed backwards from the last stopping point. This solution is suboptimal when every function is multidirectional. In practice, it has been observed that only the paths dependent on the failing function must be performed backwards and the others remain unchanged. When establishing stopping points and backward points the following aspects are taken into consideration:

- the minimum frequency of performance for registering the dependencies and other

information about the performance of the programe;

- the procedure for establishing selective testing points by using the information and the guiding points so as to develop the restore algorithm;
- the selective backwards algorithm based on guiding points.

To demonstrate how to use the stopping point and backward point technique in order to increase the reliability of financial and accounting software based on object oriented modelling, two arguments are taken into account:

- investigating the way group stopping points, for isolated groups of objects and communication ways of the programe, are established;
- investigating the way in which certain performing ways from a programe can be performed backwards in a selective way and others continue to be performed; during this period the general well-being of the programming system is preserved.

Developing error tolerance schemes at a high level involves the usage of selective algorithms. In (Porter, 2003) it is said that the conventional models, that coordinate the process of restoring, after errors of interacting components are detected, must be implemented at the top of selective algorithms.

By using these techniques, the results obtained due to the growth in error tolerance and, therefore, of the reliability, indicate the fact that using selective schemes at processing levels is better than using techniques based on checkpoints and using recovery schemes when the number of current functions or error numbers are high.

6. CONCLUSIONS

Models are classified by the method the model is generated. The first type is the part of the theoretical models that are based on the existence of some relationships between variables. The second type is based on extracting dates and its fundamental is statistic analyze. The third type is a models combination where the intuitive factor is used to determine the models constantly. In practice, is adopted the combined model.

The metrics models are classified in: process model metrics and product model metrics. The first group improve the software development process and the second group concentrates on measurement of the software products intern attributes and to establish a connection between them and the extern products quality.

REFERENCES

- Cristescu M.P., *Modele de evaluare a fiabiltăţii sistemelor de programe*, Teză de doctorat, A.S.E. Bucureşti, 2003;
- Davis A.M., "*Software Requirements: Objects, Functions, and States*", Prentice-Hall, Saddle River, New Jersey, 2003;
- Gandy A., Jensen U., *A non-parametric approach to software reliability: Research Articles*, Applied Stochastic Models in Business and Industry, v.20 n.1, p.3-15, January 2004;

- Goron S., "*Fiabilitatea softului*", RISOPRINT Publishing House, Cluj-Napoca, 1997;
- Huang C.Y., Kuo S.Y., Lo J.H., Lyu M.R., *Quantitative Software Reliability Modeling from Testing to Operation*, Proceedings of the 11th International Symposium on Software Reliability Engineering (ISSRE'00), pp. 72, October 2000 ;
- Jeske D.R., Zhang X., *Some successful approaches to software reliability modeling in industry*, Journal of Systems and Software, v.74 n.1, p.85-99, January 2005;
- Kapur P.K., Goswani D.N., & Gupta A., *A software reliability growth model with testing effort dependent learning function for distributed systems,* International Journal of Reliability, Quality and Software Engineering, 11(4), 2004, pp. 365-378;
- Kim S., Clark J.A., and McDermid J. A., "*Class mutation: mutation testing for object-oriented programs*", in Proceedings of the NetObjectDays - Conference on Object-Oriented Software Systems, 2000;
- Pham H., "*Software Reliability*", Springer, 2000;
- Porter A.A., "*Developing and analyzing classification rules for predicting faulty software components*", in Proceedings of the 15th International Conference on Software Engineering and Knowledge Engineering, San Francisco, California, June 2003, p.453-461;
- Szyperski C., "*Component Software Beyond Object-Oriented Programming*", Addison-Wesley and ACM Press, 2002;
- Teodorescu L., Ivan I., "*Managementul calităţii software*", INFOREC Publishing House, Bucharest, 2001;