Proceedings of the 33rd International Conference on Information Technologies (InfoTech-2019)
19-20 September 2019, Bulgaria

TRANSIENT IN THE SOFTWARE SYSTEMS

***Digest of paper*[1]**

Dmitry Maevsky, Elena Maevskaya, Andriy Bojko, Oleksandr Besarab

Educational and Scientific Institute of Electromechanics and Energy Management
Odessa National Polytechnic University
e-mails: {Dmitry.A.Maevsky, E.J.Maevskaya, Boyko, Besarab}@opu.ua
Ukraine

Abstract: The article shows that a software system that is in the process of testing can be considered as a non-equilibrium system in the transient mode. The simulation of this transitional process was carried out using an electric equivalent circuit. Modeling allowed to conclude that in the process of testing real and ideal systems that interact with each other. The report draws an analogy between the process of identifying defects in a program and the transition process in an equivalent electrical circuit. This made it possible to explain some of the features of the defect detection process.
Key words: defect flow, dynamic of program systems, failure, growth reliability models, software reliability.

1. INTRODUCTION

Software reliability theory is a relatively young science. Historically, work [1] should be considered, most likely, the first one. During this time, which came with the release of this work, numerous authors have developed dozens of different models of reliability [2], the so-called Software Reliability Growth Models (SRGM). Today, the contradictions in the modern theory of reliability are clearly visible.

The first contradiction is that a single model of reliability is not built. All of these models attempt to describe the evolution of the software defect detection process over time.

The second contradiction is the problem of choosing a single model from a huge number of them for practical application. There are many models, but there are no clear recommendations when and how should be used.

[1] The full paper is proposed for including in the IEEE Xplore Digital Library

The third contradiction is the impossibility of taking into account existing SRGM secondary defects. Secondary are understood as such defects which are entered the software at correction of the found defects.

The fact that in nearly sixty years of existence of the theory of software reliability, these contradictions have not been overcome, suggests that the traditional theory of software reliability has reached an impasse. Exit from this deadlock is offered by the theory of Software System Dynamics [3]. Let's cover in more detail the bases of the new theory.

2. ABOUT SOFTWARE SYSTEM DYNAMICS

In the theory of software system dynamics (SSD) two streams of defects are considered. The first stream is the defects revealed in the software system and eliminated during testing. This stream is directed from the software system to the environment. Defects of this stream are a subject of consideration of all existing SRGM. The second stream of defects (secondary defects) is directed to the software system.

The number of defects that are currently in the software system is determined by the total effect of these two defect streams. It is assumed that the transfer of defects by both streams is subject to the generalized transfer law [4]. Based on this law in [3] it is shown that the process of defect transfer in the software system is subject to the equations (1):

$$\begin{cases} \dfrac{df_1}{dt} = -A_1 f_1 + A_2 f_2 \\ \dfrac{df_2}{dt} = A_2 f_1 - A_1 f_2 \end{cases}, \tag{1}$$

where A_1 and A_2 are the constants.

Solutions of the system (1) concerning defects of f_1 and f_2 are dependences:

$$f_1 = F_0 \cdot e^{-A_1 t} \cdot \cosh(A_2 t), \tag{2}$$

$$f_2 = F_0 \cdot e^{-A_1 t} \cdot \sinh(A_2 t). \tag{3}$$

In these formulas, F_0 is the initial number of defects in the software system at the time of test-beginning.

To verify the dependencies (3) and (3), the time series of defect detection in 21 software systems experimentally determined in [5] were used. As a result of verification, as shown in [3], simulation using SSD theory showed significantly higher simulation accuracy (six times) for all the software systems studied. It allows us to conclude that on the basis of SSD theory it is possible to build a universal model of software reliability. The SSD has revealed some unknown patterns in the process of identifying defects. So, for example, in Fig. 1 shows the phase portrait of the system (1) constructed for $F_0 = 100$.

As can be seen from the phase trajectories for f_1 and f_2 streams, there is an internal equilibrium phenomenon in the software system.

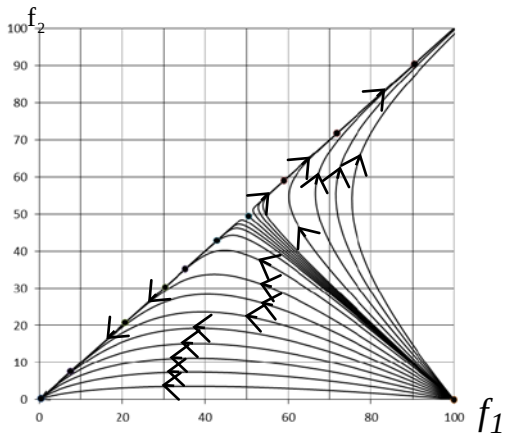

Fig. 1. Phase trajectories of the process of defects detection

This phenomenon is that over time the number of defects detected (f_1 stream) becomes equal to the number of secondary defects introduced (f_2 stream). After establishing the stream equilibrium, the number of defects either tends to zero (at $A_1 > A_2$) or tends to infinity at $A_1 < A_2$. Physical explanation of this phenomenon in previous works on SSD is not given.

Besides, equations (1) actually describe the transient phenomenon in software systems. This transient process lies in the transition of the system from the first steady state, in which the system has the maximum number of defects F_0 in the second steady state, in which the system defects are completely absent. This article examines the patterns of occurrence and development of such a transient.

3. TRANSIENT IN THE PROGRAM SYSTEMS

By simple transformations from a system (1) it is possible to obtain the differential equations of rather direct and reverse stream of defects. For direct stream of the primary defects, the differential equation is:

$$\frac{d^2 f_1}{dt^2} + 2A_1 \cdot \frac{df_1}{dt} + \left(A_1^2 - A_2^2\right) \cdot f_1 = 0, \tag{4}$$

and for reverse stream of the secondary ones:

$$\frac{d^2 f_2}{dt^2} + 2A_1 \cdot \frac{df_2}{dt} + \left(A_1^2 - A_2^2\right) \cdot f_2 = 0. \tag{5}$$

As follows from the theory of transients in non-equilibrium systems, all parts of the system are under the influence of a single transition process [6]. These equations differ only in their values and initial conditions $f_1(0) = F_0$, $f_2(0) = 0$.

The solution of equations (4) and (5) taking into account initial conditions has the form:

$$f_1(t) = \frac{F_0}{2} e^{(-A_1+A_2)t} + \frac{F_0}{2} e^{(-A_1-A_2)t}, \tag{6}$$

$$f_1(t) = \frac{F_0}{2} e^{(-A_1+A_2)t} - \frac{F_0}{2} e^{(-A_1-A_2)t}. \quad (7)$$

It is easy to see that expression (6) after conversions turns into expression (4), and expression (7) – into expression (5), which are obtained in the theory of dynamics of software systems.

It is known from the theory of transients that the process occurs inequality of energy reserves accumulated in different parts of the system. And if the process of detecting defects in software is a transient process, then the question of the nature of this energy is interesting. The answer to this question can give a representation of the software system and the environment in the form of an electrical equivalent circuit.

4. ELECTRICAL MODELLING OF TRANSIENT IN THE SOFTWARE SYSTEMS

Transients are observed in all real systems. But they are the most studied for electrical circuits. Therefore, it is necessary to try to find an analogy between the software system and the electrical circuit. To do this, we construct an equivalent electrical circuit, the transition process in which will be similar to the process of detecting defects in a software system.

It can be shown that such an equivalent circuit is the circuit shown in fig. 2.

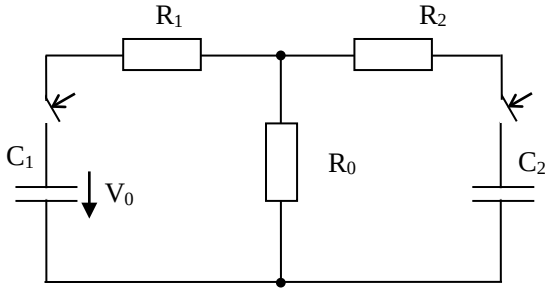

Fig. 2. Equivalent electric circuit

Based on Kirchhoff's laws, it is easy to verify that the law of change in time of voltages V_{C1} and V_{C2} on capacitors C_1 and C_2 is described by a system of two differential equations, which in their structure correspond to the system (1). Note that the voltage on the capacitor is connected with its charge directly proportional to the dependence $Q=CV$. Therefore, we can assume that the charge of capacitors C_1 and C_2 in the equivalent circuit corresponds to the number of defects f_1 and f_2 in the software system.

The characteristic equation in accordance with [7] has the form, similar to the form of equation for software system only if the relations $R_1=R_2=R$, $C_1=C_2=C$ are fulfilled. In this case, constant A1 and A2 can be calculated as:

$$A_1 = \frac{R + R_0}{\left(2R_0R + R^2\right)C}; \; A_2 = \frac{R_0}{\left(2R_0R + R^2\right)C},$$

and voltage variation V_{C1} and V_{C2} according to [7] is defined as:

$$V_{C1} = \frac{V_0}{2} e^{p_1 t} + \frac{V_0}{2} e^{p_2 t},\ V_{C2} = \frac{V_0}{2} e^{p_1 t} - \frac{V_0}{2} e^{p_2 t}.$$

It is not difficult to see that, taking into account these expressions are similar to the laws in the number change of defects in the software system (6) and (7).

Electricity approach to the consideration of the process of identifying defects allows us to explain the phenomenon of the emergence of the equilibrium flow of defects shown in fig. 1. Indeed, in the equivalent circuit, discharging capacitor C_1 will transfer its charge to capacitor C_2. This process will occur until the charges of the capacitors become equal to each other. In the software system, there is a balance between the number of primary and secondary defects. After the balance of charges, both capacitors will discharge, which corresponds to the downward branches of the phase trajectories on fig. 1.

5. CONCLUSION

Thus, the theory of transients in software systems allowed us to explain the phenomenon of equilibrium flows, which was not explained in the theory of the dynamics of software systems. The driving force behind transients in a software system is its energy reserve. Therefore, it is very interesting from the point of view of practice and philosophy to study the question of the origin and properties of this energy. Promising in the study of this issue is the resource approach described in [8] and [9].

REFERENCES

[1] R. Barlow, E. Scheuer (**1966**). Reliability Growth during a Development Testing Program. *Technometrics*, No. 8(1), pp. 53-60.

[2] H. Pham (**2006**). *System software reliability*. London: Springer.

[3] D. Maevsky (**2013**). A New Approach to Software Reliability. *Lecture Notes in Computer Science*, pp. 156-168.

[4] L. Onsager (**1931**). Reciprocal relations in irreversible processes. *Physical Review*, vol. 38, No. 12, pp 4005-4026

[5] M.R Lyu (**1996**). *Handbook of Software Reliability Engineering*, London: McGraw-Hill.

[6] Y.P. Kondratenko, D. Simon (**2018**). Structural and parametric optimization of fuzzy control and decision making systems. In: *Recent Developments and the New Direction in Soft-Computing Foundations and Applications. Studies in Fuzziness and Soft Computing*, vol. 361, pp. 273-289, doi: https://doi.org/10.1007/978-3-319-75408-6_22

[7] A. Robbins, W. Miller (**2004**). *Circuit analysis*, New York: Thomson/Delmar Learning.

[8] A. Drozd, M. Drozd, V. Antonyuk (**2015**). Features of Hidden Fault Detection in Pipeline Components of Safety-Related System, *CEUR Workshop Proceedings*, vol. 1356, pp. 476-485.

[9] A. Drozd, M. Drozd, O. Martynyuk, M. Kuznietsov (**2017**). Improving of a Circuit Checkability and Trustworthiness of Data Processing Results in LUT-based FPGA Components of Safety-Related Systems, *CEUR Workshop Proceedings*, vol. 1844, pp. 654-661