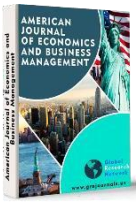


American Journal of Economics and Business Management

Vol. 9 Issue 4 | **pp.** 638-644 | **ISSN:** 2576-5973
Available online @ https://www.globalresearchnetwork.us/index.php/ajebm

Article

Experience in Detecting Software Errors in Real Time

Abidov Abdujabbar Abduxamidovich

1. TSUE, Doctor of Economics Sciences, Professor
* Correspondence: AbidovAbdujabbar11@gmail.com

Abstract: Development of a successful IT product is a complex multi-stage process with a number of mandatory stages, some of which can take place in parallel. It is important to note that different outsourcing companies define the stages of product development differently, and it is important that the process is completely transparent for the development of client applications. There are various approaches to defining the stages of software development. In this study, much attention is paid to error detection methods and the results of their application are presented.

Keywords: application failures, error detection methods, algorithm design, procedure development, application interaction.

1. Introduction

The architecture of a communication node consists of physical, logical, and software-based components. Physical elements of the communication node: multiplexers, modems, adapters, computer hardware (RAM, processor, input- output channels, etc.). Logical and software structures (elements of these two structures are considered together here and below) are mainly software products: the operating system of the communication node, data flow control programs, programs for logical communication of the processing process, data transfer with the physical device, storage of data structures and logical connections between them, etc. Errors in the software of the node can pose a significant threat, and the use of methods for their detection, localization and recovery allows solving a number of problems. In this paper, this issue will be analyzed in more detail [1].

Literature Review

A system for ensuring real-time software fault tolerance has been considered in a number of works [2]. T. Anderson, J.K. Knight and K.M. Krishna, A.D. Singh present checkpoint-based restart and time-based backup methods as the main models for predicting software reliability. A. Burns et al. introduced a probabilistic model in scheduling analysis as part of a probabilistic guarantee that all tasks will always be executed on schedule and on time. I. Broster, A. Burns, G. Rodriguez- Navas extended this method to CAN networks to calculate accurate predictions of failure probabilities based on response time probability distributions. However, these approaches have certain limitations and often produce overly conservative results [3]. Safety-critical applications must operate correctly and on time even if they contain errors. G. M. de A. Lima, A. Burns proposed a more correct response time scheduling analysis for fault-tolerant demanding real-time systems, taking into account the recovery of tasks running with high priority, and introduced a special privilege algorithm to improve the fault tolerance of the system. As a result, attempts were made to consider the relationship between processor overtime and fault tolerance [4].

Citation: Abduxamidovich, A. A. Experience in Detecting Software Errors in Real Time. American Journal of Economics and Business Management 2026, 9(4), 638-644

Received: 10th Jan 2026
Revised: 21th Feb 2026
Accepted: 08th Mar 2026
Published: 15th Apr 2026

Copyright: © 2026 by the authors. Submitted for open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/)

2. Materials and Methods

The research methodology used methods for collecting and analyzing scientific and practical data to cover the topic of "Application of error detection methods in online information systems". Information was collected from scientific sources such as articles, books, and studies by international organizations concerning methods to ensure software reliability and stability of software, concepts of error detection, their localization, and software recovery were studied. The stability of information systems in different countries and the process of recovery from the consequences of their failures were also compared using a comparative analysis method [5].

The data obtained during the study were systematically analyzed and evaluated in terms of the fault tolerance of the communication channel and the advantages of application recovery methods. Based on this information, innovative approaches can be recommended to improve the stability of information systems [6].

3. Results and Discussion

A comparison of detection methods based on the criteria of complexity and probability of error detection allows us to recommend the following methods for detecting critical errors in the software of a communication node: copy comparison, boundary value testing, element-by-element comparison, and module comparison.

Table 1 describes the execution predicates for each of these methods, and the detected critical errors in the communication node software are described in [7].

To ensure timely localization of errors and data volumes, element-wise or byte-wise monitoring of the communication node software tables, contexts, pools, and queues is performed. Situation check programs have been created for this purpose. In contains a list of symptoms for these checks, their scope, error causes, critical error types, and corresponding recovery methods. [8]

The scientific works present the relationship between the main blocks of the communication node and internal data. It is possible to create an algorithm that adapts to various conditions of communication node failure. This algorithm includes the following steps: launching the necessary procedures from the set of existing situations of the verifier P(1,1), P(2,1), ..., P(17,1), analyzing a certain situation after checking, choosing a solution from the diagnostic table, and the study of the implementation of adaptive action is covered in detail in the following work [9].

Table 1. Detection methods used for software of economic objects and critical errors detected by them

№	Detection method	What is monitored and what is the essence of the method?	Identification
1	Copy comparison method	Compares the current value α of the parameter Υ and the previously saved values β. If $\alpha \neq \beta$ is not equal, then an error is considered to have occurred	MOSK
	Threshold value method	It is checked that the value of the specified parameter Υ does not deviate from the threshold values. The success predicate is taken into account in the interval $\alpha < ◎ < \beta$. Here α, β are the minimum and maximum permissible values of the parameter Υ. ◎ - present value.	MOPR

№	Detection method	What is monitored and what is the essence of the method?	Identification
3	Element- wise comparison method	The values of the elements of a given component Q are compared with the elements of a given instance of the component Ǵ. The success predicate must be Qj = Ǵj for all j, where j=1,n; n is the number of elements.	MOESR
4	Modular comparison method	Aj - the address of the element (A1) is compared with the address of the first element to be multiplied. The success predicate Aj=(A1+r)*j, where j=2,n; n is the number of elements, r is the element size.	MOMD

Further works are devoted to the software implementation (development) of these algorithms [10].

Let us consider the development of software tools that increase the reliability of the communication node, including functions and tasks of checking conditions, a solution table and adaptation tools.To study (research) the developed methods for detecting and eliminating errors, it is necessary to programmatically consider the situations given in Table 1. Table 2 indicates the name, code, purpose of the software tool and its relationship with the situational checks presented in the previous section.

All operational control checks are carried out as a separate procedure. Each of them is performed independently. The functions of these procedures include: selecting actions from the decision table to restore the normal state Table 2 [11]. Table 2 lists all the procedures included in the periodic monitoring - PERKONTR.

Table 2. Modules for ensuring and monitoring the stability of the information system

pp	Module identifier code	Tasks of software modules
1	**2**	**3**
1-4	OK1- OK4	Procedure for identifying and restoring the normal state of indicators of levels 1-4
5.	OK5	Access control to the FIFO queue buffer address and correction of critical errors
6.	OK6	Access control to the LIFO queue buffer address and correction of critical errors
7.	KONTAB	@ group 1 (fixed data) security and control of value recovery
8.	KONKBAZI	@ group 2 (quasi-variable data) security of values and recovery control
9.	PKI	Run KONTAB, KONKBAZImodules

pp	Module identifier code	Tasks of software modules
10.	PROW3	Compare the current total number of buffers in all queues and pools with the initial number of buffers
11.	ANALOS, OSANAL	Checking the correctness of storing values in FIFO queue control elements
12.	SHIBOS	Monitoring and restoring buffers in funds LIFO in
		ANALOS
13.	PERKONTR	Implementation of RKI, PR0W3, ANAL0S periodic control procedures
14.	KBAZITRL, COPTRLIN	Copy path/line context after KK creation or destruction
15.	KBAZIKK, COPTKK	Create copies of the KK input/output table of the path/line context
16.	KBAZID	Run the KBAZITRL, KBAZIKK procedures
17.	COPTAB	Copy values of a constant data group after the initial startup
18.	COPTABUD	Copy values of a quasi-variable data group after the initial startup
19.	SHICOP	Copying from long and short buffer addresses
20.	PKCOP	Execute the COPTAB, COPTABUD, SHICOP procedures

The PERKONTR procedure is a generalized software block (PKB, KKB, etc.) and is included in the communication node software as a separate function with its own flag and status [12]. Hereinafter it is referred to as the control and diagnostics block (BCD).

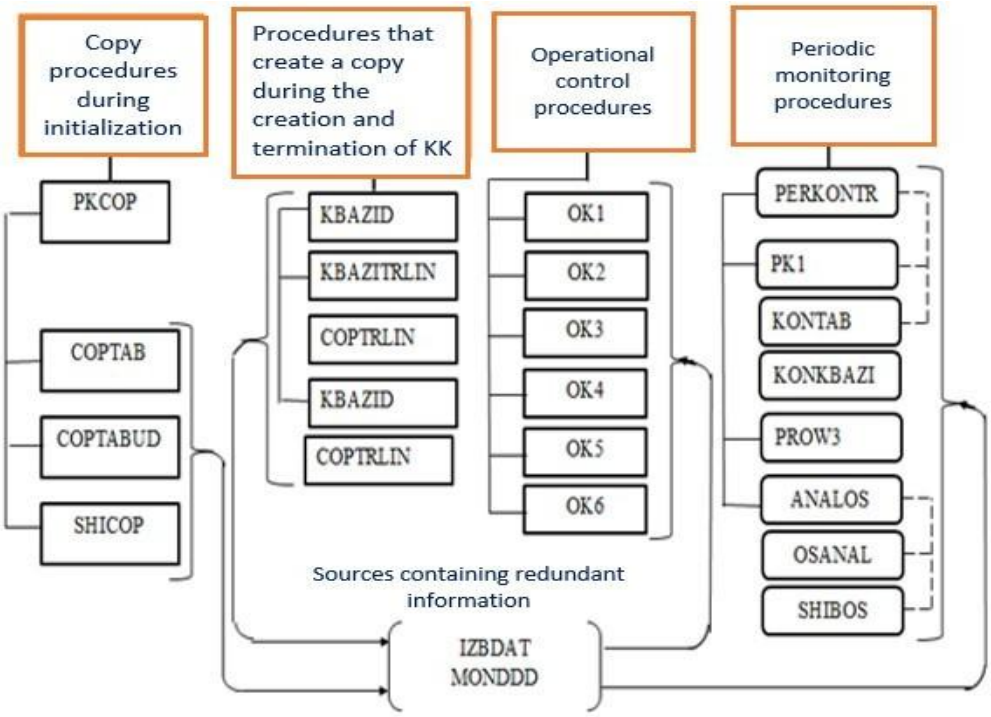
When servicing each request to establish or terminate a KK, the channel switching unit starts the procedure for copying the KK tables of the path or line context.

To do this, it is enough to enter one command in the channel switching block. By calling the PKCOP procedure after the initialization phase, the BI block allows the node to copy constants and quasi-variables to the shared memory area.

By calling the PKCOP procedure after the initialization phase, the BI block allows the node to copy constants and quasi-variables from the shared memory area.

The IZBDAT and MONDIAG files serve these purposes and contain copies of this data. Figure 1 shows the relationship between the developed procedures and the data containing additional information. To improve the reliability of the communication node software, all procedures of the software package were implemented in the C language [13].

The layout of the proposed operational control procedures in accordance with the specified structure of the software package and the logical connection of the procedures with the files created during the development of the package can be seen in Picture 1.



Picture 1. Dependencies of node modules of an object with redundant information

The order of execution of operational control procedures is arbitrary. It is developed as a separate procedure (part of the program), which allows adding the necessary procedure to each part of the communication node software. The need for a special study is determined in Table [14]. Its brief analysis shows the feasibility of using operational control procedures that exist in all main blocks of the node software, with the exception of the PKB block. The OKZ and OK4 procedures are not connected to the BKD block, since the BKD compensation buffers do not interact with the internal data of the input and output tables.

The connection of these procedures to software blocks is carried out by adding these command blocks to the original program code of the communication node to call the above operational control procedures. Then the block (request) is translated and then compiled.

This part considers the algorithm of periodic controls.

Periodic checks are useful for identifying and eliminating critical errors that affect the performance of the communication node. The main procedures of periodic monitoring are: PK1, PROW3, ANALOS. They are carried out under the supervision of the control and diagnostic department. The general scheme of operation of the control and diagnostic unit is as follows:

1. Start the control and diagnostic unit:
2. Run the RK1 procedure.
3. Run the PROW3 procedure.
4. Analysis of PROW3 performance:

In case of failure (the PROW3 procedure compares the current sum of all buffers in the queue with the value saved during initialization (the state of the software node at startup), and if they are not equal, it is considered that critical error number 7 (cr.osh.7) has occurred), go to point 5.

If there are no problems, then go to point 6.

5. Run the ANALOS procedure (process).
6. Termination of the control and diagnostic unit.

The algorithm of the procedures included in the control and diagnostic unit is implemented in software.

From the above generalized scheme of the control and diagnostic block operation it is evident that the data management of the constant memory area (procedure PC1) is

performed in all cases when the control and diagnostic block occupies processor time. The current sum of all queue buffers is compared with the specified number at the beginning of each periodic control. In necessary situations (if it is necessary to perform the ANALOS procedure, i.e. if the PROW3 procedure detects the loss or damage of a buffer in the communication node software), the state of queues and pools is analyzed in order to identify incorrect connections, parameter violations and incorrect indicators in them. The main tasks of the copying procedures are the creation of copies of constant and quasi-variable data. After the initial initialization and re-initialization (these functions are performed by the BI block), the data must be prepared in triplicate, making two copies. Data copying is performed using the RKSOR procedure. To do this, it is sufficient to enter the RKSOR call command ("variables") in the source text of the BI block; When the BKK block services a request to create or terminate a BKK, changes occur in the KK tables, paths or line contexts. These changes are recorded by the KBAZID procedure, and the data are tripled.[15]

4. Conclusion

The economic system operates through interconnected structures, including a communication node, a control system, and has a complex architecture consisting of physical, logical, software, etc. The possibility of using a priori adaptation for the data structures of the communication node is analyzed and it is established that its model can be presented as a decision table. This process is carried out using simulation modeling, and in a number of works this solution is presented more comprehensively.

REFERENCES

[1] T. Anderson and J. C. Knight, “A framework for software fault tolerance in real-time systems,” IEEE Transactions on Software Engineering, vol. SE-9, no. 3, pp. 355–364, 1983.

[2] C. M. Krishna and A. D. Singh, “Reliability of checkpointed real-time systems using time redundancy,” IEEE Transactions on Reliability, vol. 42, no. 3, pp. 427–435, 1993.

[3] A. Burns, S. Punnekkat, L. Strigini, and D. R. Wright, “Probabilistic scheduling guarantees for fault-tolerant real-time systems,” in Dependable Computing for Critical Applications, pp. 361–378, 1999.

[4] I. Broster, A. Burns, and G. Rodriguez-Navas, “Probabilistic analysis of CAN with faults,” in Proc. 23rd IEEE Real-Time Systems Symposium, pp. 269–278, 2002.

[5] G. M. de A. Lima and A. Burns, “An effective schedulability analysis for fault-tolerant hard real-time systems,” in Proc. 3rd Euromicro Conference on Real-Time Systems, pp. 209–216, 2001.

[6] G. M. de A. Lima and A. Burns, “An optimal fixed-priority assignment algorithm for supporting fault-tolerant hard real-time systems,” IEEE Transactions on Computers, vol. 52, no. 10, pp. 1332–1346, 2003.

[7] A. A. Abidov, Issues of Ensuring Information Security of Economic Objects in the Digital Environment. Tashkent: Tashkent State University of Economics, 2023.

[8] S. I. Samoylenko, A. A. Davydov, V. V. Zolotarev, and E. I. Tretyakova, Computer Networks (Adaptability, Noise Immunity, Reliability). Moscow: Nauka, 1983.

[9] V. Izosimov, P. Pop, P. Eles, and Z. Peng, “Design optimization of time- and cost-constrained fault-tolerant distributed embedded systems,” in Proc. Design, Automation and Test in Europe, pp. 864–869, 2005.

[10] A. A. Abidov, Functional Aspects of Adaptation, Modeling, and Algorithmization of Reliable Operation of Real-Time Systems. Tashkent: Tashkent State University of Economics, 2022.

[11] E. U. Rakhmatkariev and A. A. Abidov, “Procedures for control and diagnostics of packet switching unit,” Moscow: VINITI, 1986.

[12] A. A. Abidov, “Software for identification and initialization of switching node parameters in online mode,” Certificate DGU No. 22739, Intellectual Property Agency of the Republic of Uzbekistan, Mar. 2, 2023.

[13] B. Kernighan, D. Ritchie, and A. Feuer, The C Programming Language: Problems and Exercises in C. Moscow: Finance and Statistics, 1985.

[14] A. A. Abidov, "Simulation modeling of reliability of packet switching unit," in Y. Koucheryavy and A. Aziz, Eds., Internet of Things, Smart Spaces, and Next Generation Networks and Systems (NEW2AN 2022), Lecture Notes in Computer Science, vol. 13772, Springer, Cham, 2023, pp. 38–45. doi: 10.1007/978-3-031-30258-9_4.

[15] M. Sebastian and R. Ernst, "Modelling and designing reliable on-chip communication devices in MPSoCs with real-time requirements," in Proc. 13th IEEE Int. Conf. on Emerging Technologies and Factory Automation, pp. 1465–1472, 2008.