Mathematical Model for Providing Remote Monitoring of Hardware and Software Complex at the Stage of Integration Testing

NATALIA MAMEDOVA, TIMOFEY BOLONIN
Basic Department of Digital Economy,
Plekhanov Russian University of Economics,
36, Stremyanny Lane, Moscow, 117997,
RUSSIA

Abstract: - This research is a contribution to the field of solutions for the effective process of technical hardware and software development. A mathematical model of providing remote monitoring of the hardware-software complex under development at the stage of integration testing is proposed. Monitoring of the hardware-software complex functioning in the process of development allows for leveling the limitations connected with resource-intensive integration testing and determining the influence of the developed parts of the complex (hardware and software) on the operability of the inherited functionality of the systems of the external environment. The data of mathematical modeling is proposed to be used to develop an emulator of the external environment systems of the developed hardware-software complex on a test bench and to conduct integration testing. The solution is a stochastic model since the subject of remote monitoring is defined as random events of the process of integration testing of the hardware-software complex. The mathematical model for time series modeling takes into account a set of metrics of hardware-software complex functioning and requirements for the future software implementation of the solution - the remote monitoring service being developed. The implementation of the mathematical model can be used by the IT developer when integrating monitoring data into the automated test system of hardware and software complex development.

Key-Words: - Mathematical modeling, ARIMA, hardware-software complex, remote monitoring system, development, integration testing, system engineering, machine experiment, software implementation.

Received: May 17, 2024. Revised: December 8, 2024. Accepted: January 11, 2025. Published: March 10, 2025.

1 Introduction

Hardware and software systems provide a variety of often critical information processes. Their importance in computing and their place in systems engineering have been described in many studies, [1], [2], [3], [4]. Applied to the concept of the algorithm of a complex system, [5], [6], [7] hardware-software complexes combine software and hardware parts and constitute a complex of hardware and software of the system, allowing the collection, processing, storage, and display of information about the state of objects in real-time (National standard of the Russian Federation 52980-2008 “Functional safety of electrical, electronic, programmable electronic safety-related systems applications for railways. Software requirements”). In the applied aspect, they function in the form of access control systems, database systems, technological branch equipment, and complexes for production automation, [8], [9], [10], [11].

Despite the fact that the hardware-software complex does not realize the information technology of performing the established functions, but serves for solving certain technical tasks, monitoring of its state is an important activity throughout the entire product life cycle. And this statement finds confirmation in a number of studies, [12], [13], [14].

Common solutions in the IT market in the field of monitoring the operation of hardware and software complexes mainly offer systems that implement real-time monitoring - showing only current indicators, [15], [16], [17], [18]. But in the process of development and testing of hardware-software complexes, it is also necessary to study the state of the machine during the test period and during the periods of the prototyping cycle with varying degrees of detail. This need is reflected in studies, [17], [19], [20], [21], [22] abstract reviews [23], [24], [25] on hardware, software development methods and practices. Solutions for monitoring operating hardware and software systems have

been implemented many and with different functionality, [10], [26], [27], [28], [29], [30]. At the same time, there are no open monitoring solutions for hardware-software complexes being designed, and developed, and even more so for future hardware-software complexes, which can be built into the system of hardware-software complexes development.

Familiarisation with separate components of the technical documentation of several test management systems, [31], [32], [33], [34] allowed us to establish that the mathematical models on the basis of which these systems were developed were based on the analytical modeling apparatus, but no prediction tools were used. In addition, the studied solutions are designed for automatic testing, which certainly reduces operational costs, but does not enrich the process of developing an emulator of systems emulator of the external environment of the hardware and software complex under development.

IT developers widely use the practice of integration tests, [19], [35], [36], [37], [38]. Their results to some extent duplicate the data that are formed based on the results of monitoring the operation of a hardware-software complex, but only in terms of data on the interaction of hardware and software parts of the complex with other systems. Of course, it is extremely important to have an idea of their interrelation and check their joint performance. This allows to detect and leveling out of defects related to modular assembly, launching, and synchronization of information architecture components, [39], [40], [41], [42].

However, the emulation mode of the complex environment on the test bench does not provide enough information about the state of the machine under test directly - they appear to be noisy as a result of simultaneous execution of multiple tests on different code or data instances. Besides, integration testing (namely regression testing as its part) is limited to checking that changes in the environment have not affected the operability of the inherited functionality. This is not applicable to designed and future machines, whose task is to make changes from the system engineering point of view.

The authors believe that monitoring the hardware-software complex at the stages of the life cycle, preceding the commissioning of the complex, allows not only to control of the process of its development but also to determine the impact of the developed parts of the complex (hardware and software) on the environment, mainly in terms of metrological parameters of hardware. It is also necessary to take into account the following peculiarity of the hardware-software complex - the operator (user) from the point of view of the system architecture is not included in its structure, i.e. is outside the system. On the one hand, this simplifies the monitoring process, because the operator's algorithm is outside the observed process. On the other hand, the operator's observations and experience cease to be a source of monitoring data, which limits the results of monitoring to the data of hardware state, and measurements of physical quantities characterizing the performance of technical tasks. This aspect should be emphasized, as monitoring results become the basis for making decisions on information infrastructure management, and also become a source of data for designing new IT products and selecting their architecture. After all, today monitoring systems do not just accumulate the data of observations of system behavior, they fully implement the logic of the Data-driven approach, [43], [44], [45], [46].

Thus, there is a need to develop a solution for monitoring the functioning of hardware and software complexes (hereinafter - machines) at the stage of their development and integration testing. We deliberately complicate the task of developing such a solution by identifying the need to adapt it to the functionality of the automated remote monitoring service. The solution should provide such service functions as real-time data collection and/or one-time collection of certain data (disk read and write tests, local network search) and/or data collection over a period of time (for example, the program starts monitoring and stops it at the end of its work). The automated remote monitoring service will need to consider the following metrics of hardware and software system operation:

1) the lifecycle of the processes;
2) specified algorithm for measuring physical quantities of equipment operation;
3) system load;
4) network activity;
5) state of services/background processes;
6) information about installed applications;
7) information about the local network.

In this study, the authors focused on the aspect of mathematical support and administration of hardware and software complexes. The authors' area of interest was the processes of functioning of technical and mathematical means of automation of computations and information processing and, search for solutions for effective administration of computing equipment. The purpose of the research is to show how the problem of providing remote

monitoring of a machine at the stage of integration testing is mathematically solved. The authors propose to use the monitoring data to develop an emulator of the systems of the external environment of the hardware-software complex on a test bench and to conduct integration testing. The mathematical model is intended to avoid the limitations and difficulties of testing associated with the resource-intensive process, [47], [48]. The implementation of the mathematical model is supposed to be used for the subsequent integration of monitoring data into the automated development test system.

Achievement of the set goal is relevant not only for IT developers of hardware and software complexes (the objects of this study) but also for companies specializing in software development. Problems at the stage of integration testing arise, in particular, due to the fact that the IT developer has no control over the load on the system during the parallel launch of integration tests. This leads to the failure of the whole system, disrupting the work of the tests. To prevent this, the authors propose to monitor the machine on the basis of a mathematical model.

The mathematical model of solving the problem of providing remote monitoring of the machine at the stage of integration testing takes into account the list of requirements for the future software implementation of the solution - the developed service of automated remote monitoring. This is necessary so that the constructed mathematical model can be effectively used by the object.

The authors also defined the parameters, the sample of which was used for a) setting the mathematical problem of providing remote monitoring of the machine at the stage of integration testing; b) setting the problem of integrating the mathematical model into the automated test system used by the object during the development of the hardware-software complex. The selection of parameters includes the following items:

1. Running processes. It is necessary to display a list of all running processes in a tree format to track the inheritance structure. During the test bench operation, many background processes with a deep level of inheritance are created, so if an error occurs in one of them, it is necessary to understand the whole path to the exception source.
2. System load. Output information about the current system load of the main hardware components: hard disc, processor, RAM, and others. It is also supposed to run a performance test of certain parameters: the speed of reading and writing from the disc, the average response time of the disc, and its active time.
3. Networking. Monitoring of network activity, occupied ports and available network interfaces is required.
4. Service Monitoring. Output a list of all services and services, their description, and current status.
5. Information about installed applications. A list of installed applications and a description of each of them.
6. Local network status. Realization of interaction with the local network, search for a user or group in the network, ping nodes.

2 Building a Mathematical Model

In order to successfully pass the integration test, the optimal build of the external environment emulator of the hardware/software system under test must take into account the machine monitoring data. Since the machine devices may be under load or free at the time of monitoring, metrics at the current moment in time are not a sufficient indicator. It is necessary to predict the future behavior of the machine. Stochastic models exist to fulfillthis task.

Stochastic models are mathematical tools that help to understand and predict the behavior of systems subject to random influences, they are definitely popular in solving mathematical modeling problems, [49], [50]. Since the performance of a hardware and software system can be affected by many external factors, i.e. random events, this model is suitable for system monitoring tasks. The set of metrics is a time series - collected at different points in time values of the process under study. The most common and accurate model for modeling time series is ARIMA, based on the Box-Jenkins methodology.

Despite the popularity of the ARIMA model in general, its use in integration testing scenarios is poorly understood. Testing takes at least as much time as the development process. Therefore, optimization at this stage is an urgent problem in the field. Due to the limited computational resources of test benches, it is important to efficiently distribute tasks between machines, for this purpose, it is necessary to predict their behavior and workload in the future. At the moment there are no offers on the market that can solve this problem. Therefore, it was decided to develop a mathematical model using ARIMA, on the basis of which such a solution could be created. This approach will reduce the time spent on the integration testing stage, which will have a positive

impact on the speed of new product releases. Studies show that ARIMA, as well as its variations ARIMA-ANN, demonstrates high forecasting accuracy compared to other time series models. This is due to its ability to adapt to different data structures and identify hidden dependencies, [51], [52], [53].

The authors chose the ARIMA model because it allows the modeling of non-stationary time series that can be made stationary by differentiation. This is applicable to the problem to be solved because the model is particularly useful for analyzing data where trends are present, which is entirely relative to the integration testing process.

It is worth mentioning a number of other methods that were considered as ARIMA analog: extrapolation and machine learning methods. Such basic methods as simple extrapolation are convenient for the simplest series with a small amount of data. Such methods allow us to give a quick prediction, which, however, may have a high error. In the case of integration testing, data are collected continuously over a long period of time. Such a series includes external factors and noise, which are important to consider in forecasting. Thus, a basic extrapolation method will not be able to give accurate results. Speaking about machine learning methods (DeepAR, NBEATS, LSTM), it is important to understand that training a neural network and further forecasting using it is an extremely resource-intensive process that requires a large set of data for training. On the plus side, the accuracy of the calculation can be emphasized, which will allow forecasting for longer periods of time. However, the prediction is needed at the time of test distribution, which will be immediately sent to the required machine, meaning that the target prediction time is relatively close to the reference point. Therefore, the use of such complex neural network models would be impractical, especially considering the ultimate goal of optimizing the time taken for the entire testing phase. Thus, given the specificity of time series, the complexity of the task, and the need for resource efficiency, the ARIMA model is the most optimal option.

An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to predict future trends. The model consists of three components:

1. autoregression (AR). Regression in probability theory and mathematical statistics is the dependence of the mean value of some quantity on some other quantity or on several quantities. Autoregression is a model in which the values of a time series depend on the previous values of the same series.

2. Integration (I). Differentiation of initial observations to achieve stationarity of the series. A stationary time series is a series whose mean value does not change over time, i.e. the series has no trend.

3. Moving average (MA). A family of functions whose values at each point are equal to some average value of the original function for the previous period.

The ARIMA equation takes as input three standard parameters, each of which is responsible for the corresponding component of the model:

p – autoregressive order. It allows you to determine whether the next element of the series will be close to the value of X, if p previous values were close to it;

d – order of integration. Shows how close an element of a series is to d previous values if the difference between them is minimal;

q – moving average order. Allows to set the model error as a linear combination of previously observed error values.

The model is usually referred to as ARIMA (p, d, q), where p, d and q are non-negative integers.

As stated earlier, autoregression refers to the dependence of a subsequent value on some number of previous values. The prediction is the sum of metrics of previous measurements with some coefficients, which are constant and determine the autoregressive parameters. That is, the final formula for calculating the value at time t will be represented as a linear combination as shown in formula 1.

$$Y_t = c + \varepsilon_t + \alpha_1 Y_{t-1} + \cdots + \alpha_p Y_{t-p}, \quad (1)$$

where α1, ..., αp - coefficients, Yt - value at the moment of time

Two additional terms are introduced:

- c – constant value that is added to the prediction;
- ε_t – white noise.

White noise in an autoregressive model is a random sequence of independent and identically distributed random variables with zero mean and constant variance. It is usually assumed in time series models to account for the random component that is not explained by the autoregressive model.

The least squares method is used to calculate the autoregressive coefficients. That is, the coefficients are chosen in such a way that the sum of squares of deviations of points from the regression line is minimized. The coefficients are

estimated on the basis of experimental data containing random errors. At the end of the experiments, the coefficients that minimize the difference between the experimental data and the theoretical data are selected.

The moving average method is used to smooth time series in order to eliminate the influence of a random component. The method consists in replacing the initial values of the members of the series by the arithmetic mean of the values of several members nearest to it. The value at time t is given in formula 2.

$$Y_t = \mu + \varepsilon_t + \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q} \quad (2)$$

In essence, the series Yt is expressed through the sum of some fixed mean μ, the value of white noise at the current moment of time εt and not more than q previous values of white noise multiplied by some coefficients, which are the model parameters. The calculation of the coefficients is similar to the autoregressive case.

Summarizing the two formulas, we obtain formula 3 - the value of the ARMA model series.

$$Y_t = c + \varepsilon_t + \alpha_1 Y_{t-1} + \cdots + \alpha_p Y_{t-p} + \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q} \quad (3)$$

At the current stage, the ARMA model is fully described, assuming that the original series are stationary. In the context of system monitoring of machines, the series are not stationary, as they may have trends (e.g., the highest loads occur at night during test runs). Therefore, it is necessary to bring such series to stationary form after applying a series differentiation procedure. The ARIMA model is defined as an ARMA model for a series of finite differences of order d. Therefore, to obtain from the ARMA formula the ARIMA formula, it is necessary to apply the difference operator of degree d to each Yi.

The finite difference is denoted by Δ^d. The finite differences are calculated recurrently, as shown in formula 4.

$$\begin{aligned} \Delta^0 y_i &= y_i \\ \Delta^1 y_i &= y_{i+1} - y_i \\ \Delta^2 y_i &= \Delta^1 y_{i+1} - \Delta^1 y_i \\ \Delta^n y_i &= \Delta^{n-1} y_{i+1} - \Delta^{n-1} y_i \end{aligned} \quad (4)$$

For example, for y0 the finite differences will look as follows:

$$\begin{aligned} \Delta^1 y_0 &= y_1 - y_0 \\ \Delta^2 y_0 &= y_2 - 2y_1 + y_0 \\ \Delta^3 y_0 &= y_3 - 3y_2 + 3y_1 - y_0 \\ \Delta^4 y_0 &= y_4 - 4y_3 + 6y_2 - 4y_1 + y_0 \end{aligned}$$

Note that the moduli of the coefficients in the calculation of finite differences correspond to the rows of Pascal's triangle shown in Figure 1 (Pascal's triangle is given as an example of how the ratios can be easily memorised). The sign of the first coefficient is always positive, and further they alternate.

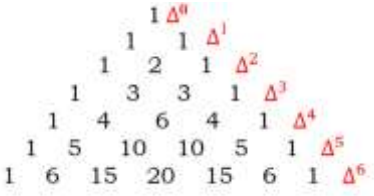

Fig. 1: Pascal's triangle (created using Adobe photoshop)

If we take yk instead of y0, then when computing finite differences, the indices of the summands will be equal to $k+n, \ldots, k$. For example, let us calculate $\Delta^2 y_1$:

$$\Delta^2 y_1 = y_3 - 2y_2 + y_1$$

The final formula of the series by ARIMA model at time t is presented in formula 5.

$$\Delta^d Y_t = c + \varepsilon_t + \sum_{i=1}^{p} \alpha_i\left(\Delta^d Y_{t-i}\right) + \sum_{j=1}^{q} \theta_j(\Delta^d \varepsilon_{t-j}) \quad (5)$$

It is important to take into account the fact that in real conditions differentiation does not always lead to stationarity of the series. Therefore, the Dickey-Fuller test is used to verify that the series is really stationary. According to this test, a series is stationary if it does not have a unit root. A time series has a unit root if its first differences form a stationary series (formula 6). Thus, further, when working with real data, it will be possible to confirm the fact that the resulting time series is indeed stationary.

$$\Delta y_t \sim I(0) \quad (6)$$

Now, having a set of already known metrics, we can predict future values. The more starting points there are, the more accurate the result will be. The problem is that storing all metrics in minimal time intervals naturally leads to a large database size, [36], [39]. n 1 night of monitor operation, the size of a single collection was about 1GB. Thus, there is a need to store data with a wider range, but still be able to access the information for any point in time. To provide both components, it is necessary to approximate known values.

Let some function bounded by a segment be given. In the initial conditions it is not given explicitly, but as a set of points and values corresponding to these points. As a solution, it is necessary to obtain a polynomial that will be considered as an approximation to the original function.

The process of calculating such a polynomial is called approximation. In the context of a machine monitor, the points are timestamps, and the values in the points correspond to the values of the metrics.

When defining the concept of function proximity, there are several approaches, one of which is called interpolation. Interpolation is a method of finding intermediate values from an existing discrete set of known values. Functions are considered approximate if their values at each known point are equal. Since a discrete set with metrics is initially given, and the function and polynomial by which it is formed are unknown, only interpolation is a suitable method of approximation.

A polynomial obtained by interpolation is called an interpolation polynomial. The degree of the polynomial is less than or equal to the number of initial points n. It is worth noting that the interpolation polynomial is singular. Indeed, let us take two such polynomials P(x) and Q(x). Since they are interpolative, for all x P(x) - Q(x) = 0. Consider R(x) = P(x) - Q(x). For all given points, R(x) = 0. Thus, R(x) is of degree no greater than n (from the definition of interpolation), but it has n+1 solutions (that is, degree n+1). We have obtained a contradiction, and so P(x) and Q(x) are identical. Since the polynomial is singular, all methods of calculating it will lead to the same result.

One of the most accurate approximation methods is interpolation by cubic splines. A spline is a continuous function defined on some segment, which is divided into several sections, each of which is represented as a polynomial. Splines consist of several polynomials joined at intersection points. The junctions must be smooth and there must be no change in the curvature of the polynomials. Smooth function - a function that has a continuous derivative over the entire set of definition. Cubic spline - a spline in which each segment is represented by a cubic polynomial that satisfies the smoothness criterion because it has a continuous first and second derivative. The first derivative determines at what angle the polynomial enters the boundary point, and the second derivative determines the curvature it has at that point. Therefore, to ensure the smoothness of the spline over the entire segment, the first and second derivatives of the joint polynomials must be equal. From the definition, it is clear why this approach is better than conventional interpolation.

The original function may have discontinuity points where the rate of change of the function increases dramatically (e.g., a hyperbola near point 0). Therefore, an attempt to approximate the entire function with a single polynomial will lead to serious errors. Whereas spline approximation breaks the function into sections, each of which interpolates separately, which minimizes the computational error.

Let there be a set of n points. It follows that there are a total of n-1 intervals, each of which must be approximated by a cubic polynomial. The equation of the polynomial of degree 3 is given in formula 7.

$$P_3(x) = a + b(x - x_0) + c(x - x_0)^2 + d(x - x_0)^3 = \text{y} \tag{7}$$

This formula has four unknown coefficients a, b, c, d; x and y are the coordinates of the point, x0 is the abscissa of the initial point from which the function starts. The total is an n-1 polynomial, that is, an n-1 equation with 4(n-1) unknowns. For each equation of the polynomial, we know its two extreme points through which it exactly passes, so the total of equations becomes 2(n-1) with the same set of unknowns. So far, there are not enough equations to find a single solution, since the number of unknowns must be no greater than the number of equations.

As noted earlier, at the junctions of two polynomials, smoothness, for which the first and second derivatives are responsible, must be observed. Therefore, in order that smoothness is not violated during the transition from one function to another, it is necessary that at the junction point their first and second derivatives are equal. The number of junctions is 1 less than the number of polynomials, for each of which two equations are made (formula 8). Thus, there are now 2(n-1) + 2(n-2) equations, that is, 4n - 6, which is two less than the number of unknowns.

$$P_i'(x_{\text{стык}}) = P_{i+1}'(x_{\text{стык}}),$$
$$P_i''(x_{\text{стык}}) = P_{i+1}''(x_{\text{стык}}) \tag{8}$$

The behavior of the spline at the ends of the interval is characterized by curvature equal to 0, that is, at these points the spline is neither convex nor concave. To comply with the achievement of this model, it is necessary that the second order

derivatives of the extreme polynomials at these points are equal to 0. That is, we add two more equations of spline behavior at the ends of the segment, described in formula 9.

$$P''_{n-1}(x_n) = 0,$$
$$P''_1(x_1) = 0 \quad (9)$$

Thus, we obtain a system of 4n - 4 equations with 4n - 4 unknowns. The final system of equations is shown in Figure 2.

$$\begin{cases} a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 + d_1(x_1 - x_1)^3 = y_1 \\ a_1 + b_1(x_2 - x_1) + c_1(x_2 - x_1)^2 + d_1(x_2 - x_1)^3 = y_2 \\ \dots \\ a_{n-1} + b_{n-1}(x_{n-1} - x_{n-1}) + c_{n-1}(x_{n-1} - x_{n-1})^2 + d_{n-1}(x_{n-1} - x_{n-1})^3 = y_{n-1} \\ a_{n-1} + b_{n-1}(x_n - x_{n-1}) + c_{n-1}(x_n - x_{n-1})^2 + d_{n-1}(x_n - x_{n-1})^3 = y_n \\ (a_1 + b_1(x_2 - x_1) + c_1(x_2 - x_1)^2 + d_1(x_2 - x_1)^3)' = \\ (a_2 + b_2(x_2 - x_2) + c_2(x_2 - x_2)^2 + d_2(x_2 - x_2)^3)' \\ (a_1 + b_1(x_2 - x_1) + c_1(x_2 - x_1)^2 + d_1(x_2 - x_1)^3)'' = \\ (a_2 + b_2(x_2 - x_2) + c_2(x_2 - x_2)^2 + d_2(x_2 - x_2)^3)'' \\ \dots \\ (a_{n-2} + b_{n-2}(x_{n-1} - x_{n-2}) + c_{n-2}(x_{n-1} - x_{n-2})^2 + d_{n-2}(x_{n-1} - x_{n-2})^3)' = \\ (a_{n-1} + b_{n-1}(x_{n-1} - x_{n-1}) + c_{n-1}(x_{n-1} - x_{n-1})^2 + d_{n-1}(x_{n-1} - x_{n-1})^3)' \\ (a_{n-2} + b_{n-2}(x_{n-1} - x_{n-2}) + c_{n-2}(x_{n-1} - x_{n-2})^2 + d_{n-2}(x_{n-1} - x_{n-2})^3)'' = \\ (a_{n-1} + b_{n-1}(x_{n-1} - x_{n-1}) + c_{n-1}(x_{n-1} - x_{n-1})^2 + d_{n-1}(x_{n-1} - x_{n-1})^3)'' \\ (a_1 + b_1(x_1 - x_1) + c_1(x_1 - x_1)^2 + d_1(x_1 - x_1)^3)'' = 0 \\ (a_{n-1} + b_{n-1}(x_n - x_{n-1}) + c_{n-1}(x_n - x_{n-1})^2 + d_{n-1}(x_n - x_{n-1})^3)'' = 0 \end{cases}$$

Fig. 2: Final system of equations

Solving the resulting system, we obtain the values of all coefficients, and hence the equations of polynomials that make up the spline. Therefore, it becomes possible to obtain metrics for any intermediate moment of time, even if it is not saved in the database.

As a result, having approximated the input set of points, we obtain a curve of values for each sample. Once predicted by the ARIMA method, the curve continues for the time interval requested by the user. Thus, the system allows to determine the estimated values of the device state at any time in the future.

3 Description of the Calculation Logic

Since the constructed mathematical model is supposed to be implemented programmatically, it is necessary to describe the development of the internal logic of the server and how calculations will be performed on the server in advance.

We have identified five main functional modules in the server structure:

1. urls.py – contains all basic URL paths processed by the server, correlated with handler methods.
2. views.py – contains the implementation of all handler methods.
3. db.py – implements methods of interaction with the database.
4. statistics.py – module responsible for the reporting section.
5. common.py – module that implements general-purpose functions for packet conversion, identifier generation, etc.

The implemented server should work with the database, exchange messages with agents and clients, respond to incoming requests according to API and perform all the provided functionality. For this purpose, a base for reporting should be prepared (documents “avg” and “avg_step” are created). Work with reports completes the development of the monitoring system and is at the same time the starting point of the mathematical model.

The module statistics.py is responsible for realization of work with reports. The work itself is proposed to be divided into three components:

1. Metrics prediction. A set of functions implementing the described mathematical apparatus is used.
2. Construction of graphs to track the dynamics of metrics change. Collection of “avg_step” documents, approximation of metrics and their visualization are planned.
3. General information about the machine, its averages, problems, etc. The “avg” document is used for this purpose.

In the program implementation of the approximation algorithm, a matrix of equations A with coefficients by which the unknowns are multiplied, a vector B of values (the right part of each equation) is created.

Let the final vector with unknowns be denoted by X, then we obtain the equation: AX = B. Let us multiply both parts on the left by the matrix inverse of A, i.e. A-1AX = A-1B. An inverse matrix is such a matrix, the multiplication of which by the original matrix yields the unit matrix E.

Therefore, the equation will take the form: EX = A-1B. By the property of a unit matrix, its multiplication by a square matrix gives the same square matrix. Therefore, the final form of the equation is as follows: X = A-1B.

Thus, to find the vector of coefficients of the polynomials, we need to multiply the inverse matrix A-1 by the vector B. Next, each segment is divided into 50 points, for each of which a value is calculated by substituting x into the polynomial. At the output, the program gives a list of points on which the spline graph is plotted.

4 Software Implementation of the Mathematical Model

The forecasting stages include two steps: ARIMA forecasting and interpolation of the results.

The ARIMA prediction stage is implemented as follows:

```
def predict(x, y, n):  # x - points, y - their values, n - how many points to predict
    x2 = list(range(len(y)-1, len(y)+n))
    df = pd.DataFrame({'timestamp': x, 'value': y})
    model = ARIMA(df['value'], order=(5,1,0))
    model_fit = model.fit()
    forecast = model_fit.forecast(steps=len(x2))
    y2 = list(forecast)
    y2 = [abs(yy) for yy in y2]
    res = statistics.generatePoints(y+y2)
    x = [x1[0] for x1 in res]
    y = [y1[1] for y1 in res]
    return x, y
```

Now the prediction results need to be interpolated. The stage of results interpolation is realized as follows.

Input point of the program:

```
def generatePoints(values):
    points = [[x, y] for x, y in enumerate(values)]
    A = generateSplineMatrix(points)  # Generation of spline matrix
    B = generateAnswersVector(points)  # Generating a vector of responses
    X = np.dot(np.linalg.inv(A), B)  # Multiplication of a matrix by a vector
    x, y = collectResult(points, X)  # Combining segments with results into one object
    return [[xi, yi] for xi, yi in zip(x, y)]
```

Next, we go through the functions that are called inside generatePoints. The implementation of the data prediction component is presented below. The component itself also uses “avg_step” and builds new points based on the received data:

```
  def generateSplineMatrix(points: list) -> np.array:
    result_array = []
    num_sectors = len(points) - 1
```

Filling of the lines responsible for the equality of the spline points to the points of the table function is realized as follows:

```
    for i in range(num_sectors):
        row = [0 for j in range(num_sectors * 4)]
        row[i * 4] = pow(points[i][0], 3)
        row[i * 4 + 1] = pow(points[i][0], 2)
        row[i * 4 + 2] = points[i][0]
        row[i * 4 + 3] = 1
        result_array.append(row)
        row = [0 for _ in range(num_sectors * 4)]
        row[i * 4] = pow(points[i + 1][0], 3)
        row[i * 4 + 1] = pow(points[i + 1][0], 2)
        row[i * 4 + 2] = points[i + 1][0]
        row[i * 4 + 3] = 1
        result_array.append(row)
```

Filling the lines responsible for the equality of first derivatives at the points of the tabular function between adjacent plots is realized as follows:

```
      for i in range(1, len(points) - 1):
        row = [0 for j in range(num_sectors * 4)]
        row[(i - 1) * 4] = 3 * pow(points[i][0], 2)
        row[(i - 1) * 4 + 1] = 2 * points[i][0]
        row[(i - 1) * 4 + 2] = 1
        row[(i - 1) * 4 + 4] = -3 * pow(points[i][0], 2)
        row[(i - 1) * 4 + 5] = -2 * points[i][0]
        row[(i - 1) * 4 + 6] = -1
        result_array.append(row)
```

Filling the lines responsible for the equality of second derivatives at the points of the tabular function between adjacent plots is realized as follows:

```
      for i in range(1, len(points) - 1):
        row = [0 for j in range(num_sectors * 4)]
        row[(i - 1) * 4] = 6 * points[i][0]
        row[(i - 1) * 4 + 1] = 2
        row[(i - 1) * 4 + 4] = -6 * points[i][0]
        row[(i - 1) * 4 + 5] = -2
        result_array.append(row)
```

Filling the lines responsible for the equality of the second derivatives to zero at the extreme points of the segment is realized as follows:

```
    row = [0 for j in range(num_sectors * 4)]
    row[0] = 6 * points[0][0]
    row[1] = 2
    result_array.append(row)
    row = [0 for j in range(num_sectors * 4)]
    row[num_sectors * 4 - 3] = 6 * points[-1][0]
    row[num_sectors * 4 - 2] = 2
    result_array.append(row)
    return np.array(result_array)
def generateAnswersVector(points: list) -> np.array:
    B = []
    for i in range(1, len(points)):
        B.append(points[i - 1][1])
        B.append(points[i][1])
    return np.array(B + [0] * len(B))
def collectResult(points: list, X: np.array):
    xArgs = []
    yValues = []
    for i in range(1, len(points)):
        x = np.linspace(points[i - 1][0], points[i][0])
        start_index = (i - 1) * 4
        y = X[start_index]*(x**3)+X[start_index + 1]*(x**2) + X[start_index + 2]*x+X[start_index + 3]
        xArgs += x.tolist()
        yValues += y.tolist()
    return xArgs, yValues
```

It is noteworthy that the program implementation uses the DataFrame library, which performs the calculations. For cases when there is no access to this library or its use is considered inexpedient by the researchers, the author's program implementation of the developed mathematical model is presented below.

Program entry point:

```
def predict_ARIMA(values, p, d, q, n):
    phi = np.zeros(p)
```

```
theta = np.zeros(q)
diff_data = np.copy(values)
```

Then the differentiation by finite differences of values (arIma) takes place:

```
for i in range(d):
    diff_data = diff(diff_data, d, i)
```

The realization of the generation of vectors with coefficients using correlation is carried out as follows:

```
for i in range(p):
    phi[i] = corrcoef(diff_data[p - i:-1], diff_data[p - i - 1:-2])
for i in range(q):
    theta[i] = corrcoef(diff_data[q - i:-1], diff_data[q - i - 1:-
2])
predicted_values = np.zeros(n)
```

The program implementation also provides for the generation of noise values:

```
noise_samples = np.random.normal(size=len(values)+n)
d_noise_samples = np.copy(noise_samples)
for i in range(len(noise_samples)):
    d_noise_samples[i] = diff(noise_samples, d, i)
```

Next, the generation of predicted values is realized:

```
for i in range(n):
    predicted_values[i] = (np.sum(phi * diff_data[-p:]) +
np.sum(d_noise_samples[i]*theta) + mean(values) +
noise_samples[len(values)+i])
return predicted_values
```

The description of functions that predict_ARIMA calls is presented below.

The average value of the series is calculated:

```
def mean(arr):
    return sum(arr) / len(arr)
```

The covariance is calculated:

```
def covariance(x, y):
    x_mean = mean(x)
    y_mean = mean(y)
    covariance_sum = 0
    for i in range(len(x)):
        covariance_sum += (x[i] - x_mean) * (y[i] - y_mean)
    return covariance_sum / len(x)
```

The method of least squares is applied:

```
def std_dev(arr):
    arr_mean = mean(arr)
    variance = sum((x - arr_mean) ** 2 for x in arr) / len(arr)
    return variance
```

The correlation is calculated:

```
def corrcoef(x, y):
    cov = covariance(x, y)
    x_std = std_dev(x)
    y_std = std_dev(y)
    correlation_coefficient = cov / (x_std * y_std)
    return correlation_coefficient
```

The finite difference method is used:

```
def diff(arr, n, i=0):
    if n == 0:
        return arr[i]
    return diff(n-1, i+1) - diff(n-1, i)
```

The final set of values is interpolated, returning a curve, some of which is based on real data and some on predicted data. Examples of predictions are shown in Figure 3 and Figure 4.

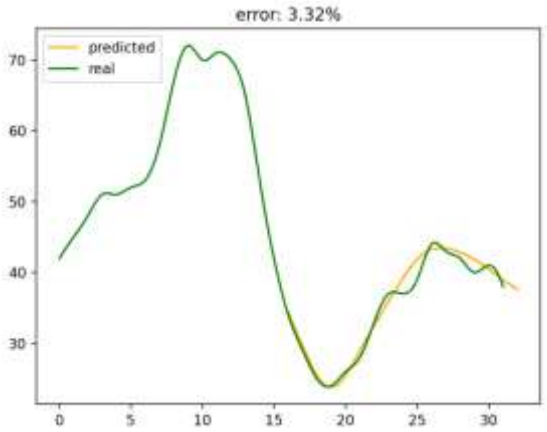

Fig. 3: Predicting metrics. RAM

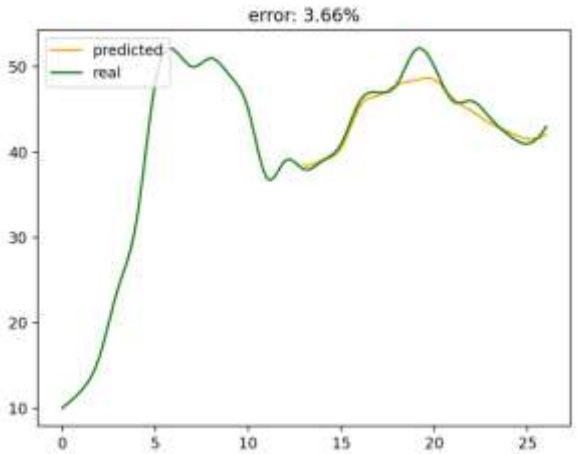

Fig. 4: Predicting metrics. CPU utilization

After taking several measurements, the average forecast error was calculated to be 5.1%.

These results show that the chosen mathematical model allows to determine the metrics in the future quite accurately. Figure 3 and Figure 4 show that the model was able to predict the RAM and CPU load of the test bench in the future. The maximum deviation of the prediction from the real data was 3.32% and 3.66% respectively. This deviation is insignificant in the context of machine load. Thus we can conclude that the developed mathematical model really works in practice and allows us to make an optimal choice of a test bench to run tests on it. This feature will allow efficient allocation of computing resources at the stage of integration testing, which will have a positive impact on its duration.

5 Interpretation of the Result of the Machine Experiment

To provide general information about the device, data from the “avg” package is extracted, and

divided by the "count" field (at the current moment or in the predicted one, depending on the request). The user will be able to see the average performance of the machine, and the acceptable norms for each of them. Based on this information, he will be able to conclude how suitable the device is for operation.

An overall assessment of the state of the machine is also formed. The following parameters are used to form the assessment: CPU load, number of physical cores, number of logical cores, RAM load, free disk space, average disk response time, disk active time, and average operation speed. All the parameters have different units and orders of magnitude, so they must first be put on the same scale from 0 to 10. Those values that are measured in percentages are divided by 10.

Other parameters are given individually:

1. The number of physical and logical processors in the most advanced mass-market models reaches the limit of 16 pieces. Therefore, we will consider this value as a reference value, and therefore divide it by 1.6.
2. We will consider 500 mb/s as the benchmark value of disk operations speed. Therefore, the metric value is divided by 50.
3. The average response time is the only parameter whose increase is negative. We will assume that the worst time is equal to 500 ms. Therefore, this sample is also divided by 50.

The result, when the parameters are out of the scale, will be considered acceptable, as it reflects the condition of the machine as ideal, above the norm. For a more accurate assessment, we will introduce a scale of weights for each parameter, which is responsible for their importance:

1. The load is 25.
2. Physical cores - 15
3. Logic cores - 5
4. RAM - 5
5. Disk space - 10
6. Response time - 10
7. Active time - 10
8. Speed of operations – 10

Then the value of each item is multiplied by its weight, and the resulting products are added together (the exception is the response time, its product is subtracted). As a result, the reference result is 1000, and the worst possible result is 0. All weights were selected on the basis of the author's experience (which values are critical, change of which parameter will have a stronger impact on the system as a whole) and manual selection of coefficients on different samples. When visualizing such statistics, the scores (denoted as R) will be marked with different colors, indicating the status of the state:

1. $R \geq 1000$ – " perfect"
2. $700 \leq R < 1000$ – " excellent"
3. $500 \leq R < 700$ – "good"
4. $300 \leq R < 500$ – "normal"
5. $100 \leq R < 300$ – "bad"
6. $0 \leq R < 100$ – "terrible"

A critical state is entered for each individual item:

1. CPU load is greater than 80%.
2. The number of cores is less than 3.
3. RAM load is greater than 80%.
4. Free disk space is less than 10%.
5. Response time is greater than 300 ms.
6. Active disk time is greater than 70%.
7. Disk transaction rate is less than 100 mb/s.

Thus, the function responsible for collecting general statistics queries the "avg" document in the database, divides its values by the "count" field, and adds the state score to it. An example of the document is shown in Figure 5.

Fig. 5: Example of a report with general information

6 Conclusion

The mathematical model provides the ability to process the monitoring data of the hardware and software complex and is fair with respect to the selected metrics and specified parameters. When selecting metrics and parameters as a methodological basis for setting the mathematical problem, the authors proceeded from the prospect of software implementation of the solution in the form of a service for remote monitoring of hardware and software complex functioning.

Thus, the obtained result of the research has theoretical significance, because it expands the field of solutions for effective development and

testing of hardware devices and software. Also the result claims applied significance, being recommended for integration into the work of the test bench at the stage of integration testing. The solution needs pilot testing, which can be considered as an applied task following the logic of research prolongation by the authors.

The possibilities for further development in this area are the deepening of research in the direction of supplementing the ARIMA methodology with other components, such as exogenous factors (ARIMAX). This will expand its capabilities and make it an even more powerful tool for the analysis of time series to monitor the functioning of hardware and software complexes. It is also worth considering the possibility of optimizing the parameters of its mathematical model for application beyond the integration testing stage, for example, for the purpose of system testing.

References:

[1] J. M. Alvarez-Rodríguez, R. Mendieta, E. Cibrián, and J. Llorens, "Towards a method to quantitatively measure toolchain interoperability in the engineering lifecycle: A case study of digital hardware design," *Comput. Stand. Interfaces*, vol. 86, p. 103744, 2023, https://doi.org/10.1016/j.csi.2023.103744.

[2] L. O. Freire, L. M. Oliveira, R. T.S. Vale, M. Medeiros, R. E.Y. Diana, R. M. Lopes, E. L. Pellini, E. A. de Barros, "Development of an AUV control architecture based on systems engineering concepts," *Ocean Eng.*, vol. 151, pp. 157–169, 2018, https://doi.org/10.1016/j.oceaneng.2018.01.016.

[3] A. Safwat and M. B. Senousy, "Addressing Challenges of Ultra Large Scale System on Requirements Engineering," *Procedia Comput. Sci.*, vol. 65, pp. 442–449, 2015, https://doi.org/10.1016/j.procs.2015.09.116.

[4] M. F. Khan and R. A. Paul, "Chapter 4 - Pragmatic Directions in Engineering Secure Dependable Systems," in *Dependable and Secure Systems Engineering*, vol. 84, A. Hurson and S. B. T.-A. in C. Sedigh, Eds. Elsevier, 2012, pp. 141–167, https://doi.org/10.1016/B978-0-12-396525-7.00005-8.

[5] L. R. Welch, A. L. Samuel, M. W. Masters, R. D. Harrison, M. Wilson, and J. Caruso, "Reengineering computer-based systems for enhanced concurrency and layering," *J. Syst. Softw.*, vol. 30, no. 1, pp. 45–70, 1995, https://doi.org/10.1016/0164-1212(94)00116-5.

[6] A. Ahmad, A. B. Altamimi, and J. Aqib, "A reference architecture for quantum computing as a service," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 36, no. 6, p. 102094, 2024, https://doi.org/10.1016/j.jksuci.2024.102094

[7] H. Beierling, P. Richter, M. Brandt, L. Terfloth, C. Schulte, H. Wersing, A.-L. Vollmer, "What you need to know about a learning robot: Identifying the enabling architecture of complex systems," *Cogn. Syst. Res.*, vol. 88, p. 101286, 2024, https://doi.org/10.1016/j.cogsys.2024.101286.

[8] H. Loschi, D. Nascimento, R. Smolenski, and P. Lezynski, "Cyber–physical system for fast prototyping of power electronic converters in EMI shaping context," *J. Ind. Inf. Integr.*, vol. 33, p. 100457, 2023, https://doi.org/10.1016/j.jii.2023.100457.

[9] T. Zhang, Y. Shi, Y. Cheng, Y. Zeng, X. Zhang, and S. Liang, "The design and implementation of distributed architecture in the CMOR motion control system," *Fusion Eng. Des.*, vol. 186, p. 113357, 2023, https://doi.org/10.1016/j.fusengdes.2022.113357.

[10] L. Yang, X. Li zhang, L. Yaoming, L. Liya, and S. Maolin, "Modeling and control methods of a multi-parameter system for threshing and cleaning in grain combine harvesters," *Comput. Electron. Agric.*, vol. 225, p. 109251, 2024, https://doi.org/10.1016/j.compag.2024.109251.

[11] P. Anistratov, Y. Golobokov, and V. Pavlov, "Hardware-software Complex Prototyping for the Pulse Power Supply Control System of Tokamak T-15," *Procedia Comput. Sci.*, vol. 66, pp. 546–555, 2015, https://doi.org/10.1016/j.procs.2015.11.062.

[12] S. Wiesner, S. Nilsson, and K.-D. Thoben, "Integrating Requirements Engineering for Different Domains in System Development – Lessons Learnt from Industrial SME Cases," *Procedia CIRP*, vol. 64, pp. 351–356, 2017, https://doi.org/10.1016/j.procir.2017.03.013.

[13] K. H. Kim, T. G. Lee, S. Baek, S. I. Lee, Y. Chu, Y. O. Kim, J. S. Kim, M. K. Park, Y.

K. Oh, "Software development of the KSTAR Tokamak Monitoring System," *Fusion Eng. Des.*, vol. 83, no. 2, pp. 291–294, 2008, https://doi.org/10.1016/j.fusengdes.2008.01.016.

[14] F. Cus, M. Milfelner, and J. Balic, "An intelligent system for monitoring and optimization of ball-end milling process," *J. Mater. Process. Technol.*, vol. 175, no. 1, pp. 90–97, 2006, https://doi.org/10.1016/j.jmatprotec.2005.04.041.

[15] S. Itaya, F. Ohori, T. Osuga, and T. Matsumura, "Smart Monitoring of Wireless Environments with Real-Time Aggregation and Analysis," *Procedia Comput. Sci.*, vol. 220, pp. 86–93, 2023, https://doi.org/10.1016/j.procs.2023.03.014.

[16] J. P. Calvez and O. Pasquier, "Real-time behavior monitoring for multi-processor systems," *Microprocess. Microprogramming*, vol. 38, no. 1, pp. 213–220, 1993, https://doi.org/10.1016/0165-6074(93)90146-C.

[17] M. Lin, D. Hou, P. Liu, Z. Yang, and Y. Yang, "Main control system verification and validation of NPP digital I&C system based on engineering simulator," *Nucl. Eng. Des.*, vol. 240, no. 7, pp. 1887–1896, 2010, https://doi.org/10.1016/j.nucengdes.2010.03.011.

[18] R. A. Swartz and J. P. Lynch, "3 - Wireless sensors and networks for structural health monitoring of civil infrastructure systems," in *Woodhead Publishing Series in Civil and Structural Engineering*, V. M. Karbhari and F. B. T.-S. H. M. of C. I. S. Ansari, Eds. Woodhead Publishing, 2009, pp. 72–112.

[19] H. Honka and M. Kattilakoski, "A simulation-based system for testing real-time embedded software in the host environment," *Microprocess. Microprogramming*, vol. 32, no. 1, pp. 127–134, 1991, https://doi.org/10.1016/0165-6074(91)90334-P.

[20] J. Van Noten, K. Gadeyne, and M. Witters, "Model-based Systems Engineering of Discrete Production Lines Using SysML: An Experience Report," *Procedia CIRP*, vol. 60, pp. 157–162, 2017, https://doi.org/10.1016/j.procir.2017.01.018.

[21] M. Foughali, P.-E. Hladik, and A. Zuepke, "Compositional verification of embedded real-time systems," *J. Syst. Archit.*, vol. 142, p. 102928, 2023, https://doi.org/10.1016/j.sysarc.2023.102928.

[22] A. EL Zerk, M. Ouassaid, and Y. Zidani, "Development of a real-time framework between MATLAB and PLC through OPC-UA: A case study of a microgrid energy management system," *Sci. African*, vol. 21, p. e01846, 2023, https://doi.org/10.1016/j.sciaf.2023.e01846.

[23] E. Stach, B. DeCost, A. G. Kusne, J. Hattrick-Simpers, K. A. Brown, K. G. Reyes, J. Schrier, S. Billinge, T. Buonassisi, I. Foster, C. P. Gomes, J. M. Gregoire, A. Mehta, J. Montoya, E. Olivetti, Ch. Park, E. Rotenberg, S. K. Saikin, S. Smullin, V. Stanev, B. Maruyama, "Autonomous experimentation systems for materials development: A community perspective," *Matter*, vol. 4, no. 9, pp. 2702–2726, 2021, https://doi.org/10.1016/j.matt.2021.06.036.

[24] E. de Araújo Silva, E. Valentin, J. R. H. Carvalho, and R. da Silva Barreto, "A survey of Model Driven Engineering in robotics," *J. Comput. Lang.*, vol. 62, p. 101021, 2021, https://doi.org/10.1016/j.cola.2020.101021.

[25] E. Hussein, B. Waschneck, and C. Mayr, "Automating application-driven customization of ASIPs: A survey," *J. Syst. Archit.*, vol. 148, p. 103080, 2024, https://doi.org/10.1016/j.sysarc.2024.103080.

[26] D. Zoni, L. Cremona, A. Cilardo, M. Gagliardi, and W. Fornaciari, "PowerTap: All-digital power meter modeling for run-time power monitoring," *Microprocess. Microsyst.*, vol. 63, pp. 128–139, 2018, https://doi.org/10.1016/j.micpro.2018.07.007.

[27] A. Askhedkar, B. Chaudhari, and M. Zennaro, "18 - Hardware and software platforms for low-power wide-area networks," B. S. Chaudhari and M. B. T.-L. T. for I. and M. A. Zennaro, Eds. Academic Press, 2020, pp. 397–407. https://doi.org/10.1016/B978-0-12-818880-4.00019-3.

[28] R. Ligeiro, "Monitoring applications: An immune inspired algorithm for software-fault detection," *Appl. Soft Comput.*, vol. 24, pp. 1095–1104, 2014, https://doi.org/10.1016/j.asoc.2014.08.021.

[29] M. J. Ringer, T. M. Quinn, and A. Merolla, "Autonomous power system: Intelligent

diagnosis and control," *Telemat. Informatics*, vol. 8, no. 4, pp. 365–383, 1991, https://doi.org/10.1016/S0736-5853(05)80060-7.

[30] P. Charalampidis, A. Makrogiannakis, N. Karamolegkos, S. Papadakis, Y. Charalambakis, G. Kamaratakis, A. Fragkiadakis, "A flexible Compilation-as-a-Service and Remote-Programming-as-a-Service platform for IoT devices," *Internet of Things*, vol. 20, p. 100617, 2022, https://doi.org/10.1016/j.iot.2022.100617.

[31] I. Heider, J. Baumgärtner, A. Bott, R. Ströbel, A. Puchta, and J. Fleischer, "Towards a Testing Framework for Machine Learning Model Deployment in Manufacturing Systems," *Procedia CIRP*, vol. 127, pp. 122–128, 2024, https://doi.org/10.1016/j.procir.2024.07.022.

[32] A. Lönnfält, V. Tu, G. Gay, A. Singh, and S. Tahvili, "An intelligent test management system for optimizing decision making during software testing," *J. Syst. Softw.*, vol. 219, p. 112202, 2025, https://doi.org/10.1016/j.jss.2024.112202.

[33] R. Seyyedi, S. Schreiner, M. Fakih, K. Grüttner, and W. Nebel, "Functional test environment for time-triggered control systems in complex MPSoCs," *Microprocess. Microsyst.*, vol. 76, p. 103072, 2020, https://doi.org/10.1016/j.micpro.2020.103072.

[34] R. Pitschinetz and J. Wegener, "TESSY - Management of Software Tests," *IFAC Proc. Vol.*, vol. 29, no. 2, pp. 11–16, 1996, https://doi.org/10.1016/S1474-6670(17)43770-0.

[35] S. Ali, L. C. Briand, M. J. Rehman, H. Asghar, M. Z. Z. Iqbal, and A. Nadeem, "A state-based approach to integration testing based on UML models," *Inf. Softw. Technol.*, vol. 49, no. 11, pp. 1087–1106, 2007, https://doi.org/10.1016/j.infsof.2006.11.002.

[36] F. Mattiello-Francisco, E. Martins, A. R. Cavalli, and E. T. Yano, "InRob: An approach for testing interoperability and robustness of real-time embedded software," *J. Syst. Softw.*, vol. 85, no. 1, pp. 3–15, 2012, https://doi.org/10.1016/j.jss.2011.02.034.

[37] Y. Ding, Y. Zhang, G. Yuan, S. Jiang, and W. Dai, "Progress on class integration test order generation approaches: A systematic literature review," *Inf. Softw. Technol.*, vol. 156, p. 107133, 2023, https://doi.org/10.1016/j.infsof.2022.107133

[38] Y. Wang, M. V Mäntylä, Z. Liu, and J. Markkula, "Test automation maturity improves product quality—Quantitative study of open source projects using continuous integration," *J. Syst. Softw.*, vol. 188, p. 111259, 2022, https://doi.org/10.1016/j.jss.2022.111259.

[39] F. Trautsch, S. Herbold, and J. Grabowski, "Are unit and integration test definitions still valid for modern Java projects? An empirical study on open-source projects," *J. Syst. Softw.*, vol. 159, p. 110421, 2020, https://doi.org/10.1016/j.jss.2019.110421.

[40] S. S. Yadav, A. Kumar, P. Johri, and J. N. Singh, "Chapter 6 - Testing effort-dependent software reliability growth model using time lag functions under distributed environment," in *Emerging Methodologies and Applications in Modelling*, P. Johri, A. Anand, J. Vain, J. Singh, and M. B. T.-S. A. Quasim, Eds. Academic Press, 2022, pp. 85–102.

[41] F. Saglietti, N. Oster, and F. Pinte, "White and grey-box verification and validation approaches for safety- and security-critical software systems," *Inf. Secur. Tech. Rep.*, vol. 13, no. 1, pp. 10–16, 2008, https://doi.org/10.1016/j.istr.2008.03.002.

[42] N. C. W. M. Braspenning, R. Boumen, J. M. van de Mortel-Fronczak, and J. E. Rooda, "Estimating and quantifying the impact of using models for integration and testing," *Comput. Ind.*, vol. 62, no. 1, pp. 65–77, 2011, https://doi.org/10.1016/j.compind.2010.05.011.

[43] E. Losi, L. Manservigi, P. R. Spina, and M. Venturini, "Data-driven approach for the detection of faults in district heating networks," *Sustain. Energy, Grids Networks*, vol. 38, p. 101355, 2024, https://doi.org/10.1016/j.segan.2024.101355

[44] Y. Tan, H. Tian, R. Jiang, Y. Lin, and J. Zhang, "A comparative investigation of data-driven approaches based on one-class classifiers for condition monitoring of marine machinery system," *Ocean Eng.*, vol. 201, p. 107174, 2020, https://doi.org/10.1016/j.oceaneng.2020.107174.

[45] Y. Xu, Y. Qamsane, S. Puchala, A. Januszczak, D. M. Tilbury, and K. Barton,

"A data-driven approach toward a machine- and system-level performance monitoring digital twin for production lines," *Comput. Ind.*, vol. 157–158, p. 104086, 2024, https://doi.org/10.1016/j.compind.2024.104086.

[46] A. Puliyanda, K. Srinivasan, K. Sivaramakrishnan, and V. Prasad, "A review of automated and data-driven approaches for pathway determination and reaction monitoring in complex chemical systems," *Digit. Chem. Eng.*, vol. 2, p. 100009, 2022, https://doi.org/10.1016/j.dche.2021.100009.

[47] S. Hamdan and S. Alramouni, "A Quality Framework for Software Continuous Integration," *Procedia Manuf.*, vol. 3, pp. 2019–2025, 2015, https://doi.org/10.1016/j.promfg.2015.07.249.

[48] Y. Li, J. Wang, Y. Yang, and Q. Wang, "An extensive study of class-level and method-level test case selection for continuous integration," *J. Syst. Softw.*, vol. 167, p. 110614, 2020, https://doi.org/10.1016/j.jss.2020.110614.

[49] S. M. Krone, "Spatial models: stochastic and deterministic," *Math. Comput. Model.*, vol. 40, no. 3, pp. 393–409, 2004, https://doi.org/10.1016/j.mcm.2003.09.037.

[50] M. Voskoglou, "3.7 - A Stochastic Model for the Modelling Process," C. Haines, P. Galbraith, W. Blum, and S. B. T.-M. M. Khan, Eds. Woodhead Publishing, 2007, pp. 149–157.

[51] C. N. Babu and B. E. Reddy, "A moving-average filter based hybrid ARIMA–ANN model for forecasting time series data," *Appl. Soft Comput.*, vol. 23, pp. 27–38, 2014, https://doi.org/10.1016/j.asoc.2014.05.028.

[52] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," *Neurocomputing*, vol. 50, pp. 159-175, 2003, https://doi.org/10.1016/S0925-2312(01)00702-0.

[53] C. Christodoulos, C. Michalakelis, and D. Varoutas, "Forecasting with limited data: Combining ARIMA and diffusion models," *Technol. Forecast. Soc. Change*, vol. 77, no. 4, pp. 558–565, 2010, https://doi.org/10.1016/j.techfore.2010.01.009.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

The research was funded by the grant Russian Science Foundation № 24-21-20089.

Conflict of Interest

The authors have no conflicts of interest to declare.