Vladimir Vychuzhanin, Alexey Vychuzhanin

DIGITAL METHODS AND MODELS FOR CONTROL AND SURVIVABILITY OF COMPLEX TECHNICAL SYSTEMS

MONOGRAPH

Львів-Торунь
Liha-Pres
2025

UDC 004.896:629.5.064
D 56

Author Team:
V. Vychuzhanin, A. Vychuzhanin

Recommended for publication by the Academic Council
of the National University "Odesa Polytechnic"
(protocol No. 5 dated 05.11.2025)

Reviewers:
*Professor **Ye Zhengmao**, Southern University (USA);*
*Professor **A. Kupin**, Kryvyi Rih National University (Ukraine).*

D 56 **Digital** methods and models for control and survivability of complex technical systems : monograph / V. Vychuzhanin, A. Vychuzhanin. – Lviv-Torun : Liha-Pres, 2025 – 366 pages. Language: English.

ISBN 978-966-397-556-6

DOI 10.36059/978-966-397-556-6

The monograph presents modern methods for diagnostics, prognostics, and reliability management of complex technical systems. It examines system properties, digital twins, and Industry 4.0 concepts, along with approaches to technical condition assessment and failure risk analysis. The work highlights information technologies for processing streaming data and hybrid models that integrate physical calculations with machine learning. Cognitive and simulation modeling methods, as well as intelligent decision support tools, are described. Experimental results for ship power plant models are provided. The monograph is intended for researchers, engineers, and graduate students in the field of reliability and digitalization of technical systems.

UDK 004.896:629.5.064

ISBN 978-966-397-556-6

CONTENTS

LIST OF ABBREVIATIONS AND SYMBOLS

AI - Artificial Intelligence
BD - Database
BZ - Knowledge Base
BN - Bayesian Network
CA - Critical Application
CBR - Case-Based Reasoning
CCS - Complex Critical Systems
CSM - Cognitive Simulation Model
CTS - Complex Technical System
DES - Discrete-Event Simulation
DM - Decision Maker
DMI - Damaging Modeling Impulse
DSS - Decision Support System
DT - Digital Twin
DTRS - Data Transmission and Reception System
FC - Set of Intercomponent, Interelement Connections
FE - Set of Subsystems, Components, Elements
IIS - Intelligent Information System
IC - Intercomponent Connection
k-NN - k-nearest neighbors
L-BFGS-B - Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm with Box
MAE - Mean Absolute Error
MAPE - Mean Absolute Percentage Error
ME - Main Engine
ML - Machine Learning
MM - Markov Model
MSE - Mean Squared Error
OLAP - Online Analytical Processing
OREDA - Offshore Reliability Data database
PM - Planned Maintenance
PHM - Prognostics and Health Management
R^2 - Coefficient of Determination
RMSE - Root Mean Squared Error
RUL - Remaining Useful Life
SIM - Simulation Modeling
SPP - Ship Power Plant
SW - Software
TC - Technical Condition
TM - Technical Maintenance
TS - Technical System
TTF - Time-to-Failure

INTRODUCTION

Reliable operation of complex technical systems (CTS) remains one of the key challenges in modern engineering disciplines. The increasing complexity of system architectures, the integration of cyber-physical components, the use of intelligent control systems, and the rapid growth of operational data volumes require a revision of traditional approaches to monitoring, diagnostics, and prognostics of equipment health. Modern CTS feature a high degree of interconnection between elements and are highly sensitive to deviations in operating modes, which leads to accelerated degradation processes and complicates failure prediction. At the same time, the requirements for accuracy, speed, and validity of technical decisions continue to grow, as errors in health assessment may result in significant economic losses or safety risks.

Classical diagnostic methods based on scheduled inspections, simple statistical models, and expert judgments provide only a partial solution, as they do not account for the dynamics of structural and functional interactions or the influence of digital factors. In the context of industrial digitalization, there is a growing need for systems capable of analyzing large data streams, integrating physical models with machine learning methods, and providing adaptive and autonomous decision-making. Of particular importance are approaches that can handle uncertainties, detect weak degradation signals, evaluate risks under various accident development scenarios, and consider rapidly changing operating conditions.

These challenges highlight the necessity of a new generation of diagnostic and prognostic tools that unite engineering principles with advanced computational intelligence. As technical systems evolve into distributed, data-driven infrastructures, traditional engineering knowledge must be complemented by modern IT competencies, including data engineering, cloud computing, semantic modeling, and algorithmic decision support. The convergence of these fields expands the possibilities for real-time monitoring, automated pattern recognition, and predictive analytics capable of revealing latent dependencies that were previously inaccessible through classical analysis. Furthermore, the development of digital twins creates a foundation for safe virtual experimentation, optimization of operating strategies, and proactive risk mitigation long before critical events occur.

Despite notable progress in information technologies, their application in technical diagnostics and failure risk management of CTS remains fragmented. The development of a methodological basis for integrating data, models, and computational methods has become increasingly essential to enable continuous health assessment, predictive modeling under external

influences, and evaluation of structural–functional risk. Digital platforms, streaming analytics, artificial intelligence methods, hybrid digital twins that combine physical laws with empirical models, and architectures capable of high-speed real-time data processing play a critical role in this process.

This monograph examines the key properties of complex technical systems, their operational characteristics, and the fundamental concepts of technical diagnostics as a field encompassing theory, methods, and tools for determining system health. A systematic analysis of modern approaches to monitoring, diagnostics, and prognostics of CTS is presented, including classical engineering techniques, statistical models, machine learning algorithms, and hybrid digital methods. Special attention is given to failure risk assessment, degradation modeling, and the application of intelligent information technologies to enhance system reliability.

The monograph consists of four chapters. Chapter 1 discusses the properties of CTS, the transition toward digital twins, issues of technical condition and failures, as well as modern approaches to risk assessment and risk management. The evolution of methods—from traditional to hybrid and intelligent models - is examined. Chapter 2 focuses on information technologies and digital platforms applied to diagnostics, monitoring, and system health management. Architectures for streaming data processing, elements of digital ecosystems, and intelligent analysis methods for emergency and pre-emergency conditions are presented. Chapter 3 describes the methodological foundations of information support for monitoring, diagnostics, and prognostics. Approaches to integrating heterogeneous data, Big Data processing solutions, and components of the digital environment for technical state analysis are introduced. Chapter 4 presents methods and models for assessing technical condition and failure risk of functionally interconnected components. Hybrid diagnostic and prognostic models combining physical computations and machine learning techniques are proposed. Experimental results are discussed, along with software tools that automate risk modeling and provide practical implementation of digital twins in engineering applications.

The monograph is intended for engineers, researchers, graduate students, and IT specialists working in the fields of reliability, diagnostics, and digitalization of complex technical systems.

CHAPTER 1. METHODS OF ASSESSING CONDITION, FAILURE RISK PREDICTION, AND MANAGEMENT IN COMPLEX TECHNICAL SYSTEMS

1.1 Complex technical systems: from properties to digital twins

A CTS is generally understood as a hierarchically organized set of elements and subsystems, the interaction of which leads to the emergence of new properties that cannot be reduced to the sum of the properties of individual parts. A key feature of such systems is that changes in the properties of individual elements or connections may give rise to qualitatively new characteristics of the entire system's behavior. Common features of CTSs include the presence of structured links between elements, integrity and hierarchical organization, finiteness, observability, relative autonomy of subsystems, as well as nonlinearity, stochasticity, heterogeneity, and the presence of feedback [1, 2]. It is precisely the combination of these properties that makes the behavior of CTSs difficult to predict and often poorly formalizable, which necessitates the application of systemic approaches to their analysis and modeling.

From a methodological perspective, CTSs are considered as a set of interconnected subsystems, with each subsystem being able to be distinguished as an individual element of a higher level of organization. This representation makes it possible to take into account not only the structure of the system, but also the nature of interactions between its parts and the external environment, which is critically important for understanding the patterns of functioning and development [3, 4, 5]. To streamline analysis and to select adequate research methods, CTSs are classified according to several criteria: by the degree of determinacy of functioning, they are divided into deterministic and probabilistic; by the degree of organization – into well-organized, poorly organized (diffuse), and self-organizing; by the nature of interaction with the environment – into closed and open. Such classification reflects both the internal structure of CTSs and the specifics of their interaction with external conditions, providing a foundation for building models, diagnostics, and predicting the behavior of complex technical objects.

These properties complicate the modeling of CTSs. In recent years, the emergence of digital technologies has posed a challenge for science and industry, changing the approach to understanding CTSs. The appearance of the concepts of Industry 4.0 and digital transformation of industry opens new opportunities for the creation, management, and analysis of such systems, expanding their functionality and increasing the efficiency of their operation. These trends lead to the fact that classical CTSs cease to be merely physical

objects with a set of properties and turn into cyber-physical systems with integrated digital representations and continuous exchange of information between the physical and virtual environments. As a result, there arises a need to rethink approaches to diagnostics, monitoring, and management of CTSs based on new digital technologies.

Industry 4.0 describes the transition from traditional manufacturing to smart factories with a high degree of digitalization and automation. Key principles include cyber-physical systems (CPS) – the integration of physical equipment and digital models, the Internet of Things (IoT), which ensures communication between devices, as well as the use of big data and artificial intelligence for process prediction and optimization. These technologies make it possible to create complex cyber-physical systems in which digital twins play a key role in monitoring and management. Digital transformation of industry encompasses a broader range of changes, including new business models ("equipment-as-a-service"), integration of information and operational technologies, as well as organizational and cultural changes necessary for the effective use of digital solutions.

The digital transformation of industry is closely connected with the concept of "Industry 4.0," where the integration of cyber-physical systems, the Internet of Things, and data processing technologies play a key role. This paradigm sets the foundation for the formation of digital twins, since it is precisely end-to-end digitalization of processes and continuous data exchange that make possible the dynamic representation of an object's state. In generalized form, the relationship between digital transformation and "Industry 4.0" is shown in Fig. 1.1.

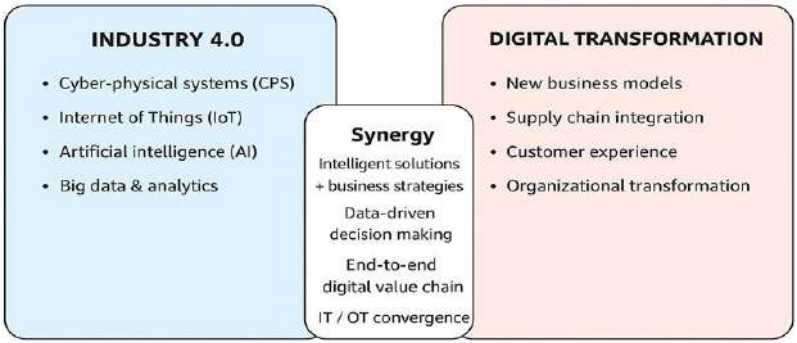

Figure 1.1. Multilevel comparison and interrelation of the concepts of Industry 4.0 and digital transformation

The modern understanding of CTSs goes beyond their classical representation as predominantly physical objects with a set of properties (nonlinearity, heterogeneity, etc.). A key new property is the cyber-physical nature of CTSs. Increasingly, they are described as complexly connected pairs of "physical asset – its virtual representation," which forms a new paradigm of management and diagnostics based on digital twins. The term *Digital Twin* was proposed by Michael Grieves in the early 2000s in the context of product lifecycle management (PLM). The original idea was to create a virtual copy of a physical product that would accompany it throughout its entire lifecycle – from design to disposal.

The purpose of the digital twin concept is to ensure continuous connection between the physical object and its virtual model through sensor data. Such connectivity makes it possible to predict failures, optimize operation, and reduce the total cost of the lifecycle.

Key features of the digital twin:

- system integration: unification of online sensor data, mathematical models, and expert knowledge into a single information space;
- real-time coupling: continuous synchronization of the virtual and physical entities;
- embedding into operational processes: use of the model to support decisions on maintenance, repair, and optimization of equipment operation;
- lifecycle orientation: focus not only on the current state, but also on forecasting, planning, and optimization;
- interdisciplinarity: the digital twin becomes a common point of interaction for engineers, IT specialists, data analysts, and operational services.

The use of the digital twin concept facilitates the integration of national developments into the global context of Industry 4.0 and digital transformation, as well as promotes the unification of terminology and the enhancement of solution interoperability at the international level.

The creation of a digital twin involves the integration of models of different natures – physical, mathematical, and data-driven – as well as the organization of continuous information exchange between the virtual and the real object. For clarity, the structure of such interaction is presented in the architectural scheme of the digital twin (Fig. 1.2).

The scheme illustrates the main levels of the digital twin: from physical sensors and actuators to analytical models, integration with enterprise systems, and decision-making loops that form a closed cycle of asset lifecycle management.

Thus, a digital twin is a dynamic, "living" virtual copy of a physical object or system that maintains a two-way connection with it through

continuous data exchange in near real-time mode. This connection is carried out by means of sensors and IoT platforms, ensuring constant updating of the digital twin's state based on real measurements and model parameters [6]. The purpose of the digital twin is not merely to provide a static representation of the object, but to deliver a dynamic depiction of its state and behavior over time, which makes it possible to perform monitoring, diagnostics, and prognostics (PHM – Prognostics and Health Management) primarily in the virtual space.

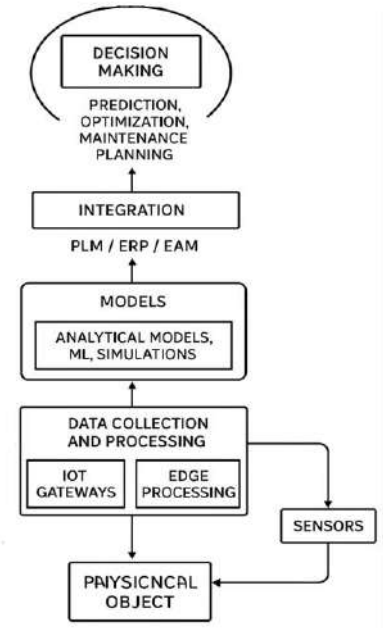

Figure 1.2. Architecture of the digital twin of a CTS

In order to emphasize the dynamic nature of the digital twin and its dependence on data, control actions, and model parameters, a formal notation has been introduced. In general form, the digital twin at time t can be represented as a mapping of the observable parameters of the object and external factors into the state of the virtual model:

$$DT(t) \approx f(X(t), \Theta, U(t)),$$

$$\hat{X}(t+\Delta t) \approx \Psi(DT(t), U(t), \Delta t), \quad (1.1)$$

where $DT(t)$ - the state of the digital twin at time *t*;

X(t) - the vector of observable parameters of the physical object (sensor data, operational indicators);

Θ - model parameters (both physical and data-driven);

$U(t)$ - control actions or environmental factors;

$f(\cdot)$ - mapping of physical data into the state of the digital twin;

$\hat{X}(t+\Delta t)$ - the predicted state of the object at time $t+\Delta t$;

$\Psi(\cdot)$ - the prediction function that takes into account dynamics and impacts

This representation emphasizes that a digital twin is a hybrid system integrating sensor data, physical models, and machine learning methods to create an accurate and up-to-date reflection of the object. This opens up opportunities for failure prediction, operational optimization, and reduction of the total lifecycle cost.

This approach fundamentally changes the methodology of data collection and analysis:

1. Data collection: from episodic to continuous and virtualized. Traditional data collection for diagnostics was often selective or periodic. A digital twin requires the creation of a "digital thread" – an end-to-end, connected chain of data throughout the entire product lifecycle [7]. Data from physical sensors (vibration, temperature, load) are transmitted in real time to update the state of the twin. In addition, virtual sensors (*soft sensors*) are used – algorithms that compute parameters that are difficult to measure directly, based on models and data from other sensors [8], which economically and effectively expand the monitoring system.

Figure 1.3 presents the concept of the Digital Thread – an end-to-end data thread ensuring continuous integration of information at all stages of the product lifecycle: from design and manufacturing to operation, maintenance, and disposal. The Digital Thread forms a unified information space, where data about the physical object and its digital twin are connected and constantly updated. Such an architecture makes it possible not only to track the current state of the system, but also to perform forecasting, optimization, and process adaptation, thereby increasing the efficiency of managing technical objects.

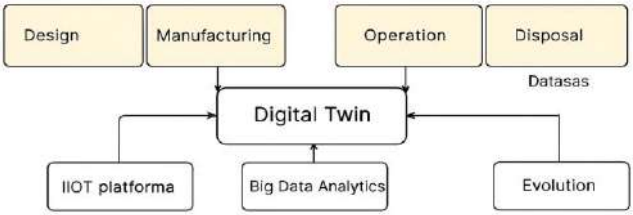

Figure 1.3. Digital thread and integration of the digital twin into the product lifecycle

2. Analysis and diagnostics: from post-mortem analysis to live simulation. Analysis is no longer merely a retrospective study of data after a failure. On a digital twin, it is possible to conduct "what-if" analyses and proactive simulation modeling [9]. For example, one can artificially "age" a specific component within the model to predict its remaining useful life, or simulate operation under extreme conditions without putting the real object at risk. This shifts diagnostics from simply recording facts to the domain of prediction and prevention.

3. Model synthesis: from physics-based models to hybrid models. A high-maturity digital twin integrates several types of models [10]:

- physics-based models: accurate mathematical models describing physical processes;
- data-driven models: machine learning models trained on historical and real-time data (e.g., for predicting remaining useful life).

A hybrid approach (*physics-informed machine learning*) makes it possible to overcome the limitations of each method individually, significantly increasing the accuracy and reliability of predictions [11, 12].

Special attention in modern research is given to hybrid models, which combine the advantages of physics-based and data-driven approaches. Traditional physics-based models ensure strict compliance with the laws of nature and allow results to be interpreted; however, they can be overly complex and may not always reflect real operating conditions. In turn, machine learning methods effectively identify hidden dependencies and anomalies in large datasets, but often suffer from a lack of physical justification and robustness.

To overcome these limitations, the concept of *Physics-Informed ML*, or hybrid modeling, is used, where machine learning algorithms are complemented with physical constraints and prior knowledge. This approach improves the accuracy of predictions, reduces the need for large volumes of data, and ensures the reliability of the obtained results.

Thus, the property of cyber-physicality and the implementation of the digital twin concept transform CTSs into fundamentally new objects for diagnostics, where the boundary between the physical and virtual worlds becomes blurred, and condition analysis becomes continuous, predictive, and deeply integrated into operation and management processes.

1.2 Technical condition and failures of complex technical systems

Failures of CTS are directly determined by their technical condition. Technical condition (TC) is understood as the set of properties of an object that change during operation and are characterized by indicators defined by standards or technical documentation [1, 13, 14]. The combination of conditions that either meet or do not meet established requirements forms different types of TC, reflecting the operability or inoperability of the system. A change in reliability status is associated with the transition of a CTS into a failure state or into pre-failure modes. At the same time, the failure itself is characterized by the extent of damage it can cause. If the consequences of the failure exceed a predetermined critical level, such a failure is classified as hazardous, as it can lead to an emergency situation.

The following main technical conditions of CTS are distinguished:

- operability – compliance of the system and its elements with the requirements of regulatory, technical, and design documentation;
- inoperability – non-compliance with at least one of the regulated requirements;
- serviceability – a state in which all parameters characterizing the ability to perform specified functions remain within the established limits;
- unserviceability – a situation in which at least one parameter does not meet regulatory requirements;
- limiting condition – a state in which further operation of the system is unacceptable, and restoration of its serviceability is either impossible or economically impractical.

To improve the quality of TC analysis, complex systems must be considered with regard to their structure. They are usually divided into subsystems (complexes and units) and elements (assemblies and parts). Subsystems are structurally and functionally complete components, whose interaction ensures the achievement of the target function. Elements, in turn, include constituent parts identified by dividing the structure without necessarily observing the principles of structural or functional completeness. At the same time, each element is interconnected within the overall structure, and identical elements may have different characteristics depending on the specific system. Therefore, the study of TC should include the identification of inter-element connections and the implementation of structural analysis.

In reliability analysis practice, failures are usually classified by the nature of their occurrence. They are divided into sudden (catastrophic) and gradual (degradation-related), complete and partial, as well as evident and hidden. Pre-failure states, which reflect the early stages of element degradation, are of particular importance. Their timely detection through modern diagnostic and monitoring methods significantly reduces the probability of the system entering a failure state. In this regard, the study of TC should be based not only on structural analysis but also on the application of prognostic and diagnostic technologies that ensure the transition from reactive to proactive maintenance [15].

It is important to take into account that the technical condition of systems demonstrates pronounced dynamics throughout their life cycle. At the initial stage of operation, the so-called "early" failures appear; during the nominal operating period, relative stability of parameters is observed; and at the aging stage, degradation processes accumulate. This dependence is clearly illustrated by the example of ship power plants (SPP). For them, the typical "bathtub curve" of failure intensity is characteristic, reflecting three phases of the life cycle: an increased frequency of initial failures, a prolonged period of stable operation, and an increase in failure probability as the equipment ages (Fig. 1.4).

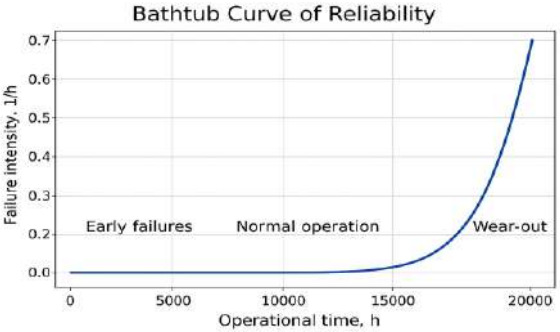

Figure 1.4. Bathtub curve of failure intensity for a ship power plant

The presented dependence emphasizes that diagnostic and prognostic methods of TC must be differentiated for each stage of the SPP life cycle. At the early stage, it is advisable to implement enhanced procedures for detecting and eliminating hidden defects (quality control of assembly, early diagnostics). During the nominal operating period, the key role is played by maintaining parameter stability and identifying weak signals of degradation. At the aging stage, the focus shifts to forecasting the residual resource and making decisions regarding repair or replacement of components. Thus, the

"bathtub curve" provides the methodological basis for developing adaptive maintenance strategies for complex technical systems.

External factors operating conditions, load regimes, environmental influences, as well as the human factor-have a significant impact on changes in TC. Therefore, the correct determination and formalization of TC are necessary both for failure classification and for subsequent application in diagnostic and prognostic models (including digital twins).

1.3 Failure risk assessment of complex technical systems

Failure risk assessment is considered the most important measure that allows the selection of the optimal solution among many alternatives at the pre-design stage of a CTS. The main objective is to minimize both the costs of creating the object and the potential losses during its operation. In this context, failure risk is understood as the combination of the probability of an undesirable event and the magnitude of potential damage. The concept of risk assessment is based on the principle of the inadmissibility of system damage probability [16, 17]. The probabilistic approach is implemented through the use of reliability theory methods for predicting the fault-free operation of CTS.

The process of constructing scenarios of possible failures or emergency situations is iterative in nature (Fig. 1.5).

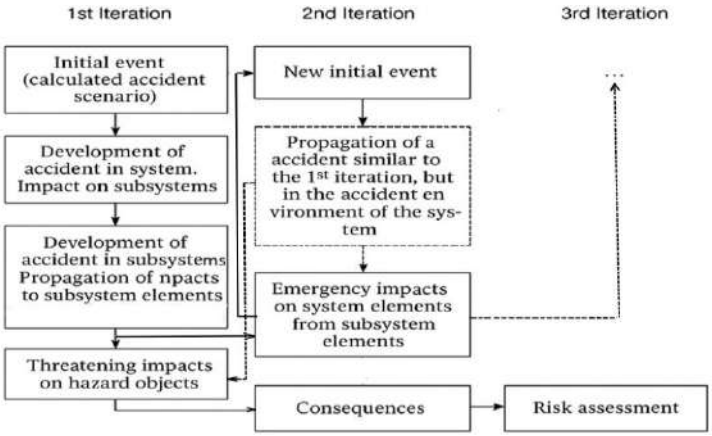

Figure 1.5. Iterative logical scheme of accident development in failure risk assessment within the system

At each stage, system performance is analyzed, potential threats are identified, and measures to improve the safety level are developed. In recent

years, digital twins and big data processing technologies have been increasingly implemented in practice. Their use makes it possible to reproduce the behavior of CTS in a virtual environment, simulate degradation processes, and predict failure probability in real time. This approach provides not only static construction of scenarios but also their dynamic refinement based on monitoring data.

The key objective of failure risk assessment of complex technical systems is the identification and qualitative or quantitative evaluation of threats that may affect their functioning. The outcome of such a procedure is the development of a set of measures aimed at reducing the probability of failure occurrence and mitigating the severity of its consequences. Failure risk assessment includes several procedures [16] (Fig. 1.6): risk analysis (defining the scope of application, identifying threats) and risk evaluation. Risk analysis provides initial data for subsequent assessment as well as for decision-making regarding risk treatment methods and the selection of strategies for their minimization.

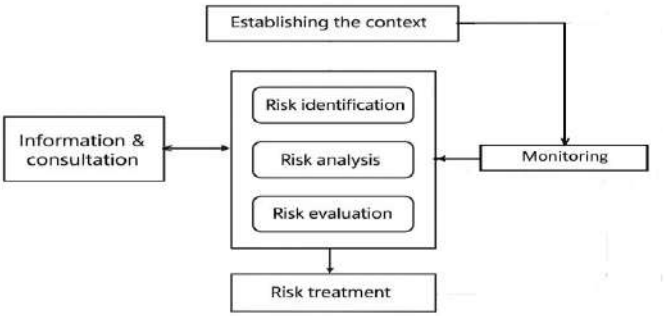

Figure 1.6. Procedures for failure risk assessment of CTS

The task of risk identification involves compiling a comprehensive list of threats associated with events that may either facilitate or hinder the achievement of system objectives. The probability of occurrence of specific events and their consequences can be determined through modeling, as well as by extrapolating the results of experimental studies.

Since the early 2020s, methods of machine learning and artificial intelligence have been actively developed, enabling the construction of adaptive risk assessment models. Unlike classical probabilistic models, these approaches can account for nonlinear dependencies and update as new data becomes available, significantly improving the accuracy of residual life forecasts and failure probability predictions. It should be emphasized that risk assessment procedures are closely linked with tasks of diagnostics and

prognosis of CTS condition. The integration of technical diagnostic methods with risk analysis procedures forms the basis for decision-making regarding maintenance strategies, modernization, or decommissioning of equipment. Risk evaluation is aimed at supporting decision-making based on analysis results. It helps determine the necessity of risk treatment and the prioritization of measures. Failure risk treatment is a cyclical process that includes risk level assessment, decision-making, implementation of corrective actions, re-treatment in the case of unacceptable residual risk, and evaluation of the effectiveness of the actions taken.

The problem of failure risk assessment of complex technical systems is determined by several factors [16, 17]:

- diversity of equipment, differing in physical principles and operating modes, which complicates the use of universal diagnostic methods;
- differences in equipment structure – approaches to risk analysis in multi-channel and single-channel systems differ significantly;
- the complex composition of CTS, including interrelated subsystems, blocks, and units with nontrivial inter-element connections and exchanges of energy, matter, and information;
- difficulties in obtaining reliable expert data on the reliability of individual elements and their interactions;
- the need to account for a large number of parameters determining CTS functioning;
- the dependence of risk assessment on statistics of past failures of elements and connections;
- the need for high-speed monitoring and risk assessment tools;
- the significant role of the human factor.

In modern conditions, special attention is given to cyber-physical threats arising from failures in control systems, distortion of diagnostic data, or external cyberattacks. The consideration of such factors becomes a mandatory element in assessing the reliability of modern CTS, where digital and physical components are closely integrated.

The acceptable risk level is determined by acceptability criteria, which depend on analysis methods, availability of information, as well as the objectives and capabilities of the researcher. These criteria are based on technical, operational, economic, regulatory, and environmental factors, or a combination thereof. Three risk levels are generally distinguished:

- highest level - the risk is unacceptable and requires immediate intervention;
- medium level - the risk may be accepted, but costs and consequences must be considered in planning;
- lowest level - the risk is so small that treatment is not required [16].

An analysis of the literature shows that in CTS condition diagnostics,

there is often no consistent analysis of systems considering their structural and functional features. Therefore, in assessing structural and functional risk, it is necessary to divide the system into subsystems and elements. Structural analysis should identify interconnections and interactions between them, as well as consider threats arising from changes in external and internal factors. Functional analysis is aimed at determining risks based on the current state of elements and subsystems.

In recent years, the concept of Prognostics and Health Management (PHM) has been increasingly adopted, which provides for the integration of risk assessment with residual life prediction and equipment lifecycle management. This approach enables the transition from reactive measures to proactive maintenance strategies aimed at preventing failures. For practical implementation, various methods are used: deterministic; probabilistic; expert-based and combined. Their classification and comparative analysis are presented in subsection 1.4.

1.4 Evolution of methods for assessing failure risks of complex technical systems

Within the technocratic concept, failure risk assessment of CTS is performed using methods classified as deterministic, probabilistic, expert-based, methods for evaluation under conditions of non-static uncertainty, and combined methods ([15, 18, 19], Fig. 1.7).

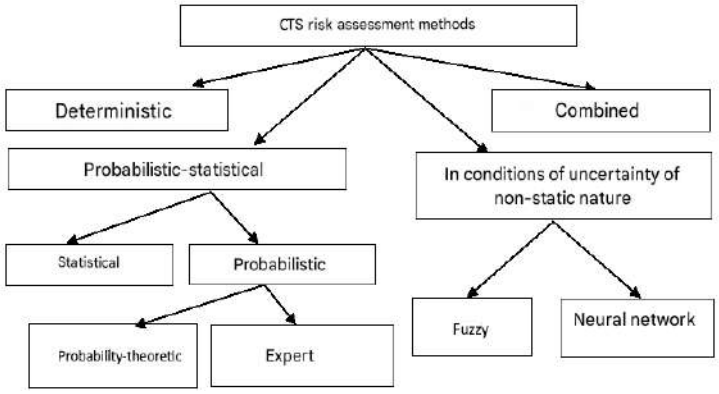

Figure 1.7. Classification of CTS failure risk assessment methods

The deterministic method involves analyzing the sequence of accident development stages, starting from the initiating event through successive stages of failures, deformations, and component destructions to the

stabilized final state of the system.

The probabilistic method of failure risk analysis involves assessing the probability of negative events and calculating the relative probabilities of process developments. This includes analyzing branched chains of events and equipment failures, selecting the appropriate mathematical apparatus, and estimating the probability of adverse events. Computational mathematical models in this approach can be simplified compared to deterministic methods. The main limitations of probabilistic safety analysis are related to insufficient information on parameter distribution functions and inadequate statistical data on equipment failures. The use of simplified calculation schemes reduces the reliability of risk estimates for accidents. Nevertheless, the probabilistic method is currently considered one of the most promising.

In practice, expert evaluation methods are widely used to assess failure development trends, based on obtaining quantitative risk estimates through the aggregation of expert opinions. The most acceptable approach in practice is the combination of probabilistic and expert-based methods.

With the development and application of information technologies, CTS failure risk assessment increasingly uses the representation of systems as directed graphs of structural interconnections of all or key elements. Such methods include the logic-probabilistic method [20], the Bayesian network method, and others, which are based on the probabilistic paradigm for evaluating and predicting CTS condition [21, 22, 23].

A promising direction in data analysis today is the method based on neural network technologies. The implementation of neural network technologies in monitoring systems allows the system not only to record parameters and compare them with reference values but also to analyze the obtained system parameters as a whole, forecasting the likelihood of CTS failures.

CTS failure risk assessment methods are grouped into quantitative, qualitative, and mixed methods. Quantitative methods allow calculation of risk in terms of the probability of causing a specific type of harm. Qualitative methods aim to obtain relative risk characteristics (low, medium, elevated, and high), while mixed methods combine both approaches.

A comparison of qualitative and quantitative CTS failure risk assessment methods in terms of ease of analysis, availability of initial data for calculations, time, and cost of obtaining such data shows that qualitative methods are simpler and cheaper, as they do not require special studies or tests and involve less time expenditure. The distinction between the methods lies in their accuracy. In qualitative risk assessment, consequences, probability, and risk level are determined using various scales. In quantitative risk analysis, the practical significance and cost of

consequences, their probabilities, and the resulting risk level are evaluated. Quantitative methods require a larger amount of initial information, some of which may be unreliable. Therefore, qualitative methods are often preferred. In mixed methods, a numerical scale is used to evaluate consequences, probabilities, and their combination to determine the overall risk level.

An analysis of publications and regulatory materials on CTS failure risk assessment methods has shown that many of them are based on probabilistic safety analyses of limited-scope systems and consider only individual sequences of technological equipment events. The diversity of CTS failure risk assessment methods is based on engineering, modeling, expert, and other approaches, often with a narrow industry focus, which limits their applicability. Many CTS failure risk assessment methods assume that all system elements operate under normal conditions. Some methods lack scalability, are significantly influenced by subjective factors that affect their accuracy and reliability, and involve complex and costly calculations that determine risk assessment values with no better than first-order accuracy. Failure risk assessment for CTS used in automated decision-making under emergency scenarios is often performed without a comprehensive evaluation of structural and functional risk, or without accounting for the interconnection and interaction of structural components.

Current trends indicate a shift from exclusively probabilistic and expert-based approaches to data-driven risk assessment methods and their hybridization with physical models.

Data-Driven and hybrid risk assessment methods. Classical approaches to CTS equipment failure risk assessment, based on theoretical distributions and expert evaluations, have certain limitations when applied to complex, unique, or rapidly changing systems. In modern scientific and engineering practice, the paradigm of Data-Driven Risk Assessment, which uses historical and operational data to build empirical models reflecting the current state and operational dynamics of a specific object, is gaining increasing adoption [24]. The implementation of this approach has been made possible by the development of digital twins and the widespread use of sensor technologies that provide a continuous stream of high-resolution data. An important trend is the emergence of hybrid methods that combine traditional probabilistic models with machine learning algorithms and big data analysis methods, enhancing the accuracy and adaptability of risk assessment.

Current CTS risk assessment practice demonstrates the need to combine physical models with data analysis methods. This approach allows consideration of both fundamental process patterns and empirical dependencies identified from operational data. The concept of hybridization is visually represented in Fig. 1.8.

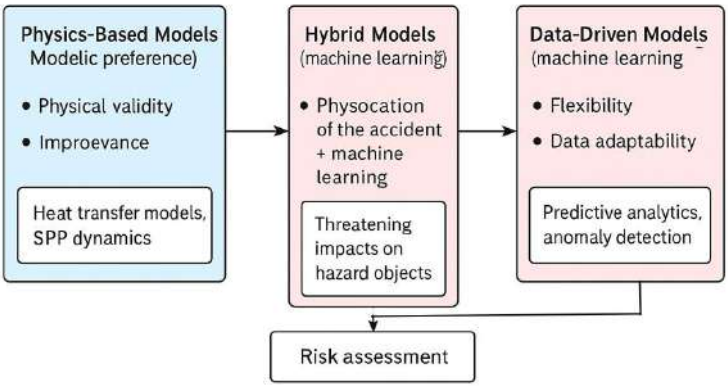

Figure 1.8. Hybridization of models – integration of physical and data-driven approaches

This approach is implemented through three main classes of methods:

1. Deep Learning for diagnostics and prognostics. Deep learning methods enable automatic extraction of high-level features from raw data, surpassing the capabilities of manual feature engineering. Convolutional Neural Networks (CNNs) are used for analyzing spatial and spectral data. They are standard for:

- vibration analysis – automatic classification of defect types in bearings and gears based on vibration spectra [25];

- thermogram analysis – detecting overheating in joints, insulation, and mechanical components from infrared images [26];

- defect image analysis – automated inspection of surfaces for cracks, corrosion, and erosion using computer vision [27].

Recurrent neural networks (RNNs), particularly Long Short-Term Memory (LSTM) networks and temporal convolutional networks (TCNs), dominate in time series analysis. They effectively capture long-term dependencies and temporal contexts for:

- remaining useful life (RUL) prediction – forecasting the remaining time to failure based on sequential operational data [28];

- anomaly detection – identifying subtle deviations in system behavior that precede failures [29].

2. Reinforcement learning (RL) for strategy optimization. RL extends risk assessment beyond prediction and turns it into an optimization tool. An AI agent is trained in a simulated environment (often based on a digital twin) to select actions that maximize a “reward,” such as minimizing total cost of ownership. Applications include:

- maintenance optimization - the agent determines optimal intervals and scope of maintenance, balancing cost of maintenance and cost of failure [30];

- adaptive monitoring - the agent decides which data to collect, at what frequency and level of detail, to maximize information value for diagnostics while minimizing network load and data storage costs.

3. Survival Analysis for censored data. Classical reliability statistics often cannot handle censored data, where failure times are unknown for many samples (still operating at the time of analysis). Survival analysis methods are specifically developed for such data and include:

- cox proportional hazards model - a semi-parametric model assessing the impact of covariates (e.g., temperature, load, vibration) on the hazard rate (instantaneous failure intensity) [31];

- random survival forests - a non-parametric method that effectively identifies nonlinear dependencies and interactions between variables without strict assumptions about data distribution [32];

- deep survival analysis - neural network architectures trained to directly predict the survival function, combining the ability of deep learning to handle high-dimensional data with the mathematical framework of survival analysis [33]. This allows raw sensor data to be integrated directly into the failure time prediction model.

Synthesis of approaches - hybrid modeling (Physics-Informed ML).

The highest accuracy and reliability are achieved not by replacing, but by integrating physical models with data-driven methods**.**

In practice, the greatest precision and dependability are obtained through the integration of physical models with data-driven approaches (Fig. 1.9).

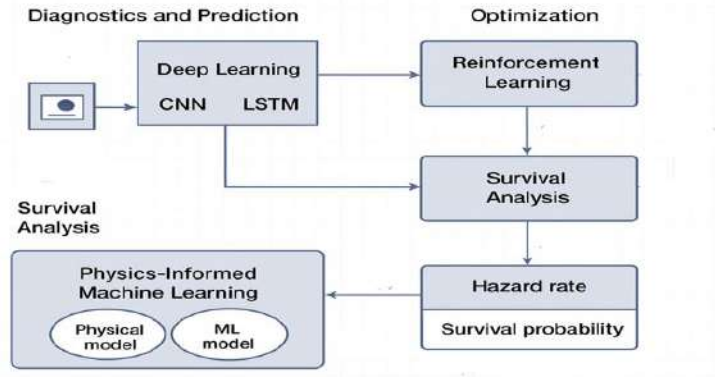

Figure 1.9. Integration of data-driven risk assessment and physics-informed machine learning in risk evaluation

Physics-informed machine learning (PIML) incorporates physical laws (in the form of differential equations or constraints) directly into the training process of the ML model [34]. This enables:

- improved prediction accuracy under conditions of limited data;
- ensuring the physical interpretability of model predictions;
- extrapolation beyond the range of training data based on physical principles.

Formally, a hybrid prediction model can be represented as:

$$\hat{X}(t+\Delta t) = f(X(t), U(t), \Phi), \tag{1.2}$$

where $\hat{X}(t+\Delta t)$ - predicted system state at time $t+\Delta t$;

$X(t)$ - current state obtained from sensors and diagnostic algorithms;

$U(t)$ - control actions or operating conditions (load, mode of operation);

Φ - parameters of the physical model (e.g., coefficients from thermodynamic, hydrodynamic equations, etc.);

$f(\cdot)$ - approximating function implemented by the hybrid model (Physics-informed ML), which accounts for both data and physical laws.

This form emphasizes that the prediction concerns the dynamic evolution of the system state while incorporating physical information.

The distinction from classical machine learning models lies in the fact that the function f(·)f(\cdot)f(·) is constructed not only based on empirical data but also considering physical constraints and laws. This provides:

- physical interpretability of predictions;
- reduced risk of overfitting under limited data conditions;
- ability to extrapolate beyond the training dataset based on fundamental system laws.

Thus, the formula reflects the fundamental characteristic of hybrid models the integration of empirical and physical information, which qualitatively differentiates them from purely data-driven approaches.

Table 1.1 presents a comparison of Data-Driven Risk Assessment approaches in terms of the type of data used, target tasks, strengths, and weaknesses. This systematization helps to better understand under which conditions each method demonstrates maximum effectiveness and what limitations should be considered in practical application.

As shown in Table 1.1, deep learning methods provide high accuracy but require large volumes of data and suffer from limited interpretability. In contrast, survival analysis performs well with limited data; however, classical models are less effective in capturing nonlinear dependencies. The greatest potential lies in hybrid approaches (Physics-Informed ML), which combine the advantages of physical models and machine learning algorithms.

Table 1.1

Comparison of approaches to Data-Driven risk assessment

Method / model	Data type	Primary task	Strengths	Limitations
Deep learning (CNN, LSTM, TCN)	Images, time series	Defect classification, RUL prediction	High accuracy, extraction of hidden features	Requires large datasets; "black box" behavior
Reinforcement learning	Sensor data, digital twins	Maintenance strategy optimization	Considers long-term consequences; self-learning	Complex tuning; high computational cost
Survival analysis (Cox, RSF, DeepSurv)	Failure time series, operational data	Failure risk assessment, time-to-failure prediction	Suitable for small datasets, interpretable	Limited nonlinear dependencies (classical methods)
Physics-informed ML (PIML)	Sensor data + physical models	Hybrid prediction	Physically meaningful predictions; reduces overfitting	Requires domain-specific physics knowledge

The analysis indicates that modern methods for assessing failure risk in complex technical systems are evolving from deterministic and probabilistic approaches toward more flexible and adaptive data-driven methods. The most significant trend is the development and implementation of hybrid models that integrate machine learning with physical constraints and process models.

The key challenges of existing approaches are associated with: insufficient and incomplete failure data; high cost and labor intensity of data collection; and the limited interpretability of purely neural network-based models.

A promising direction is the integration of hybrid methods into digital twins of technical systems and their use for automating risk assessment and management processes in real time. This approach not only enhances prediction accuracy and adaptability but also ensures the soundness of decisions in the operation and maintenance of complex engineering systems.

1.5 Monitoring, diagnostics, and prediction in the lifecycle management of complex technical systems

Monitoring, diagnostics, and prognostics of complex technical systems represent interconnected elements of a unified loop for managing their reliability and safety. Monitoring provides real-time data collection and preliminary processing, diagnostics enables the identification and classification of emerging defects, and prognostics assesses the prospective development of the technical state to prevent possible failures. Together, these processes form the foundation for transitioning from reactive to proactive maintenance, which is especially critical for complex technical systems where failure consequences are highly significant.

Monitoring of technical systems is a dynamic information-analytical system operating in real time, allowing continuous control of system elements and CTSs, generating preliminary predictive assessments and recommendations for operational and design organizations. This process utilizes databases and knowledge bases, as well as data from periodically conducted diagnostic inspections, which help refine the location of defect occurrences and the causes of malfunctions. The information-technological process of failure risk monitoring refers to the transformation of risk information into a risk assessment. The objectives of risk monitoring are presented in Table 1.2. Information technologies for monitoring technical condition have several inherent limitations. Some of these include: fragmentation of test, control, and diagnostic databases, and the lack of intelligent components that would enable qualitative and efficient support for critical decision-making, thereby reducing the overall time spent on maintenance; non-stationarity of physical processes in many CTS, the complexity of their mathematical description, dependence of technical characteristics on external operating conditions, limited composition of monitored system parameters, technological variability, and so on.

Table 1.2

Tasks of failure risk monitoring

Risk monitoring		
Analytical monitoring	**Equipment monitoring**	**Operational monitoring**
Identification of informational risk indicators	Observation of equipment	
Risk calculation	Tracking changes in risk development	
Risk assessment	Control of parameters determining risk	Monitoring outcomes of risk management
Formation of a knowledge base of the risk problem domain		
Development of information technology for risk monitoring		

An important element of protection and ensuring the safe functioning of CTS are regular and emergency diagnostic systems. Technical diagnostics (TD) [35] investigates the condition of diagnostic objects and is aimed at developing methods and tools for fault detection, as well as creating diagnostic systems using information technology methods. The specifics of TD in CTS are determined by the distinctive properties of failures in such systems and by the scenarios of accident development in CTS. Diagnostics includes three main stages: recording deviations of diagnostic parameters from their nominal values; analyzing the nature and causes of these deviations; determining the remaining service life. Technical diagnostics studies methods for obtaining and evaluating diagnostic information, diagnostic models, and decision-making algorithms, and is based on recognition theory and testability theory [36].

Recognition theory, by applying diagnostic models in the study of the object, determines decision rules for recognizing the current state and type of failure. Thanks to the known characteristics of failures, it becomes possible to develop optimal recognition algorithms. Testability theory addresses issues of rational sequencing in the search for failed or defective elements and monitoring the condition of the object. The solutions are based on the use of diagnostic information characterizing the state of the object.

The definition and selection of diagnostic parameters of CTS is a complex, poorly formalized process consisting of several stages. At the first stage, essential and diagnostic parameters are designated, and subsystems of CTS are identified. At the second stage, the initially selected parameters are refined using various formal methods. If the defining parameters are determined, diagnostic models are compiled and selected, the number of which depends on the specifics of the system being diagnosed and its operating conditions.

A promising direction in CTS diagnostics is the use of expert systems, in particular, the intellectualization of the expert-diagnostic process based on neural network modeling and fuzzy logic [37, 38]. Such systems often lack the necessary flexibility and are isolated, functioning within a closed computing environment and/or using narrowly specialized algorithms.

Monitoring, diagnostics, and prediction of the state of CTS s represent interconnected components of a unified system for managing their reliability and safety. Monitoring ensures the collection and primary processing of data on operating parameters, diagnostics provides detection and classification of emerging defects, and prediction. The third step based on the results of monitoring and diagnostics makes it possible to assess the prospective development of the technical condition and prevent possible failures. Such unity of processes forms the basis for the transition from

reactive to proactive maintenance, which is especially important for CTS with high criticality of failure consequences.

A significant problem in predicting the state of CTS lies in the neglect of correlating factors when developing such systems. To address this problem, an autoregressive model is used, in which time series are analyzed independently, even though the studied characteristics may be correlated. At the same time, compliance with the basic assumptions of regression analysis is not verified, nor are appropriate methods of adaptation to their violations applied. All this leads to errors in predicting the technical state.

The main difficulties in developing prediction and reliability management systems for CTS are associated with the fact that, for many systems, the initial information about the regularities of parameter changes is limited. The solution to this problem is found in the use of classical mathematical statistics as well as the theory of random functions for studied and described regularities. For applied research, an adaptive algorithm for predicting the development of CTS is used, based on an interdisciplinary approach of classical reliability theory, as well as the application of logical-probabilistic methods. This approach makes it possible to obtain a practical method for determining the optimal value of the period of further system operation, used as a baseline for decision-making.

In prediction, methods based on stochastic modeling are applied, using traditional techniques that represent a mathematical formalization and refinement of the concepts of technical and fundamental analysis. A widely used method is stable individual condition prediction, based on the Bayesian approach and robust statistics.

Traditional monitoring and prediction tasks, which were previously solved in isolation, are now transformed into an integrated reliability management strategy that directly influences business performance. This transformation encompasses several levels.

The modern development of digital technologies and data analysis methods is leading to the transformation of classical monitoring and prediction tasks. They are no longer considered in isolation but instead form an integrated reliability management strategy. In this context, several key directions of evolution can be identified:

- transition from predictive maintenance to prescriptive maintenance (PdM → RxM);
- formation of the asset performance management (APM) concept;
- proliferation of edge computing and AI at the network boundary (Edge AI);
- implementation of human–machine interaction and explainable AI (XAI).

1. From predictive maintenance to prescriptive maintenance (from prediction to prescription).

The comparison of the approaches predictive maintenance (PdM) and prescriptive maintenance (RxM) clearly shows that RxM not only predicts the time of failure but also generates management recommendations that optimize the asset lifecycle. Table 1.3 presents the key differences between these concepts.

Table 1.3

Comparison of predictive and prescriptive maintenance approaches

Criterion	Predictive maintenance (PdM)	Prescriptive maintenance (RxM)
Main objective	Determine when a failure will occur (RUL)	Determine what to do and why
Type of output	Prediction of time to failure	Action recommendations with assessment of consequences
Applied methods	Time series analysis, ML	ML + optimization methods, digital twins, simulations
Benefit	Reduction of unplanned downtime	Minimization of total costs, increased availability

Modern systems do not stop at predicting when a failure will occur (predictive maintenance). The next step is prescriptive maintenance (RxM) [39]. The task of RxM is to answer the questions of what to do and why specifically this:

- recommendation generation - based on failure prediction, the system automatically generates a set of possible actions (e.g., “replace the unit,” “reduce the load by 15%,” “perform unscheduled diagnostics”);

- consequence evaluation - each recommendation is accompanied by an assessment of its consequences in terms of costs, risks, required resources, and downtime;

- decision optimization - optimization methods and artificial intelligence are used to select the optimal action that minimizes total costs or maximizes system availability [40].

2. Asset performance management (APM).

Predictive models become the core of a broader concept - asset performance management (APM). APM is a comprehensive strategy that integrates monitoring data, risk management, financial planning, and

operational activities to maximize returns on physical assets throughout their lifecycle. Focus on business outcomes: The goal shifts from "preventing failure" to "ensuring maximum profitability and asset reliability at minimal total cost." Integration with ERP and EAM systems: Predictive solutions are no longer isolated. They are integrated with enterprise resource planning (ERP) and enterprise asset management (EAM) systems, providing a unified information space for decision-making [41].

3. Edge Computing and AI at the Network Edge (Edge AI).

The data flow from thousands of sensors makes centralized processing in the cloud inefficient due to latency and transmission costs. This has led to the transfer of intelligence to the network edge - edge computing [42]. Preprocessing on edge devices: data is processed directly at gateways or sensors. Filtering, compression, aggregation, and preliminary analysis are performed, which drastically reduces the volume of transmitted data. Running lightweight ML models on the edge: Trained models (e.g., for anomaly detection or state classification) are deployed on edge devices to enable real-time decision-making with minimal latency. This is critical for tasks requiring immediate response (e.g., automatic shutdown of equipment upon detection of a critical anomaly).

4. Human-in-the-loop interaction and explainable AI (XAI).

The deployment of complex AI models requires trust from experts and operators. Therefore, explainable AI (XAI) becomes critically important - the ability of a model to explain its predictions and recommendations [43]. Visualization and interpretation: The operator should not only see "failure probability 85%," but also which specific parameters deviate from the norm and which features indicate a particular type of defect. Human-in-the-Loop: The final decision often remains with the human. The system provides not raw data but ready, interpretable recommendations, which increases the effectiveness and quality of decision-making [44].

Architectural solutions of recent years are focused on shifting the computational load closer to data sources (Edge AI), followed by integration into cloud analytics platforms and ensuring the interpretability of results (XAI). Figure 1.10 presents an integrated asset performance management architecture using these approaches.

Thus, modern approaches to monitoring, diagnostics, and prediction of the technical condition of complex technical systems have evolved from simple parameter control to integrated architectures that include distributed computing ("edge–cloud") and elements of artificial intelligence. This makes it possible not only to promptly detect deviations and predict failures but also to generate scientifically grounded decisions to improve the reliability, safety, and efficiency of CTSs.

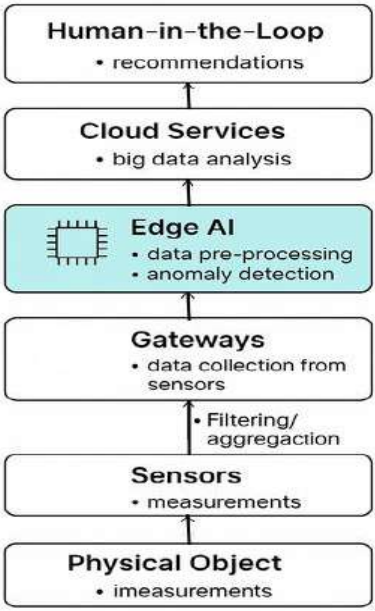

Figure 1.10. Integrated asset performance management architecture using edge AI and explainable AI

1.6 Failure risk management of complex technical systems

In the previous subsection (1.5), monitoring, diagnostics, and prediction were considered as tools for managing the condition of complex technical systems. In this subsection, the focus shifts: failure risk management represents a broader framework within which these tools are integrated into a unified strategy aimed at minimizing the probability and consequences of failures.

The goal of CTS failure risk management is to minimize the impact of random events and prevent unjustified damage to systems. In this context, risk management does not mean corporate risk management, but rather the technical management of equipment failure risks, i.e., a set of methods and tools for forecasting, preventing, and minimizing the consequences of failures during the operation of complex technical systems.

Effective failure risk management is based on a methodology [14, 45, 46] aimed at: identifying risks and assessing the probability of their occurrence and the scale of consequences; determining the maximum possible loss;

selecting risk management methods and tools; developing a risk management strategy to reduce the probability of risk occurrence and minimize potential negative consequences; implementing the risk management strategy; evaluating the results achieved; and adjusting the risk management strategy.

The process of CTS failure risk management [47] is illustrated in Fig. 1.11, where R, Rac represent the actual and acceptable risk levels. One of the challenges in CTS risk management is the lack of effective control methods and tools that would not only record the technical state of the system (simple control) and establish the causes of a component's failure state (diagnostic control), but also predict its future condition, thereby preventing failures (predictive control) [14, 18].

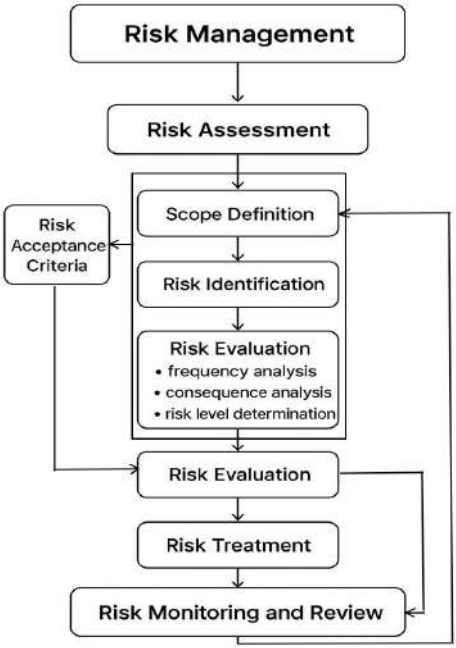

Figure 1.11. CTS failure risk management process

From the analysis of studies in the field of CTS risk management [48, 49, 50, 51], it can be concluded that the risk management methodology should adhere to the following concepts:

- decisions related to risk must be economically sound;
- in risk management, decisions should be based on an adequate volume of reliable information;
- failure risk management should have a systemic character;
- failure risk management should involve continuous analysis of the effectiveness of adopted decisions and prompt adjustment of the applied principles and methods of risk management.

Existing concepts, methods, and techniques used in risk management systems primarily differ in the level of development and sophistication of the mathematical frameworks underlying risk assessment procedures.

The evolution of modern failure risk management is characterized by a shift from discrete and reactive procedures to continuous, adaptive, and simulation-based processes. The introduction of artificial intelligence methods goes beyond improving prediction accuracy: it fundamentally changes the nature of risk management. Risk ceases to be viewed as a static probability and transforms into a dynamic category that is modeled and controlled in real time. In engineering practice, there are still no universal and adequate models for analyzing and managing failure risks of CTS. Addressing these challenges is possible through the use of innovative information technologies, including cognitive-simulation and fuzzy risk assessment models.

Analysis of existing CTS risk management methods indicates the need for methodological approaches that consider the specific interaction and functioning characteristics of the systems. For example, in studies from 2019, risk management was primarily treated as a methodology and strategy. However, by 2024, the focus shifted: risk management is implemented through intelligent decision support systems (IDSS), which aggregate data from thousands of sensors in near-real time, predict system states using AI models, and simulate the consequences of various control decisions on a digital twin before applying them in practice. As a result, risk management becomes a continuous, automated cycle in which the operator increasingly functions as the approver of strategic decisions rather than the direct initiator of actions.

Modern approaches to failure risk management have evolved from periodic procedures based on regulations and expert assessments to a continuous, automated, data-driven cycle. IDSS, leveraging artificial intelligence technologies, play a key role in this transformation [52]. A comparative characterization of traditional and AI-oriented approaches is presented in Table 1.4.

Table 1.4

Evolution of failure risk management: from traditional to AI-Based

Criterion / Approach	Traditional (rule-based)	AI-driven DSS
Decision frequency	Periodic (quarterly)	Continuous
Basis for risk assessment	Expert judgment	Predictive models (AI/ML)
Role of operator	Analyst and executor	Strategic approver
Nature of recommendations	General, regulated	Contextualized, ranked
Feedback	Limited, delayed	Continuous, closed-loop
Modeling capabilities	None	Digital twin, “what-if” analysis

Table 1.4 reflects a paradigm shift - from a reactive, episodic logic to a proactive, predictive approach based on continuous learning.

These systems fundamentally change the role of humans in the control loop:

1. From reactive response to proactive management and consequence simulation. Modern IDSS operate continuously, analyzing data from thousands of sensors in near real time:

- data aggregation and contextualization - the systems integrate equipment condition indicators, external conditions, historical maintenance data, and business context (downtime costs, spare parts expenses);

- situation development forecasting - using artificial intelligence models (PIML, survival analysis, deep learning), they predict not only the probability of failure but also the trajectory of degradation.

A comparison of the most commonly used failure prediction models is presented in Table 1.5.

Modeling the consequences of decisions (“What-If” Analysis). On the digital twin, simulations are automatically run to assess the consequences of alternative managerial actions [53].

For example, the system can compare scenarios such as continuing operation for another 48 hours, reducing the load by 20%, or immediate shutdown for maintenance. Each option is evaluated according to multiple criteria: probability of catastrophic failure, financial losses, resource consumption, and impact on related processes.

2. Shift in the role of humans: from operator to approving strategist. Initial data analysis is performed by AI algorithms that process information volumes beyond human perception. The operator receives not signals or raw

data, but ranked recommendations with justifications and simulated outcomes [54]. The human factor remains crucial in critical cases: the operator validates system recommendations, makes decisions in non-standard situations, and approves strategic plans proposed by AI.

Table 1.5

Comparison of failure prediction models

Method	Description	Advantages	Limitations	Applicability in IDSS
Survival Analysis	Modeling time to failure	Interpretability, statistical rigor	Limited flexibility	High
Deep Learning (LSTM, CNN)	Processing time series and images	High accuracy, automatic feature extraction	Requires large datasets, lack of transparency	High
Physics-Informed ML (PIML)	Combines physical models and ML	Accounts for the physical nature of processes	Requires model formalization	Promising
Reinforcement Learning	Learning based on action outcomes	Adaptivity, strategy optimization	High complexity, sensitivity	Promising

3. Closed-Loop Management and Continuous Learning. AI-driven DSS form an adaptive closed-loop cycle [55]: monitoring - collection of data from physical and virtual sensors; analysis - state prediction and generation of prescriptive recommendations; decision - selection and approval of action (automated or by a human); action - implementation of the control measure; validation - assessment of results and feedback for retraining models.

Thus, risk management ceases to be a set of prescribed procedures and transforms into a self-learning system that continuously improves the accuracy and efficiency of operation.

CHAPTER 2. INFORMATION TECHNOLOGIES AND DIGITAL PLATFORMS FOR DIAGNOSIS AND MANAGEMENT OF COMPLEX TECHNICAL SYSTEMS

2.1 Information processing and intelligent analysis in emergency situations in complex technical systems

Due to the presence of informational, computational, and telecommunication components, CTSs acquire new opportunities for improving operational efficiency. However, such systems face typical “big data” challenges:

- processing of significant volumes of information;
- complexity of interconnections, which complicates the search for optimal solutions and generates parasitic feedback;
- increased analysis time while simultaneously facing growing demands to shorten design and production cycles [56, 57].

The first problem is addressed through more powerful computational resources, parallel systems, and corporate information processing technologies. The second results in a conflict of interests between system elements and the system as a whole, which leads to the effect of information dissipation and reduced efficiency. Overcoming this requires analytical methods and information logistics tools. The third problem is solved by means of concurrent engineering methods and organizational-technical measures.

The information base of CTS monitoring and diagnostic procedures consists of arrays of physical parameters. Given the need for their processing, automated and robotic quality control systems are increasingly being introduced. At the same time, a contradiction arises: the growth of information volume is not always accompanied by an increase in its value. Therefore, reliability, relevance, consistency, and accuracy of data become particularly important.

When developing diagnostic systems, it is necessary to apply the fundamental principles of modeling the information processing workflow. At the same time, it is important to consider the specifics of diagnostic systems as a special class of subject areas characterized by large volumes of analyzed data and a limited time frame for decision-making. Practical work with large-scale information shows that one of the key problems in interacting with information resources is the contradiction between the quantity and the quality of information. Reliability, relevance, consistency, integrity, and accuracy of data thus become of paramount importance.

To evaluate the efficiency of a CTS diagnostic system, a systems approach should be applied, covering the entire information processing cycle: acquisition; verification; input into a computer; issues of information

security; user-related aspects; characteristic time intervals for obtaining information; and expected volumes. One of the main tasks of CTS diagnostics is to optimize the information support for decision-making, which consists in minimizing the volume of information required for this process. The main optimization criterion is the requirement for unambiguous classification of each possible situation. Consequently, the minimization of volume must be carried out with consideration of the significance of different information elements for ensuring the unambiguity of situation descriptions.

The effectiveness of CTS diagnostics should be evaluated within the framework of a systems approach, covering the entire information processing cycle: from acquisition and verification to storage, security, and delivery to end users. The key task here is the optimization of information support for decision-making. This consists in minimizing the volume of transmitted information while maintaining the unambiguous classification of every possible situation.

The outlined problems determine the shift from traditional post-factum accident analysis to the integration of modern approaches based on digitalization and artificial intelligence. Accident analysis is becoming a continuous process that includes predictive analytics and automated root cause detection.

Traditional control system logs are often fragmented, and in emergency conditions they may be damaged or intentionally modified by a malicious actor. This limits their value as a source of objective information. Under such circumstances, the concept of a digital trace becomes particularly significant—an automated, continuous, and immutable record of all significant system events and states.

The basic scheme for digital trace formation is shown in Figure 2.1.

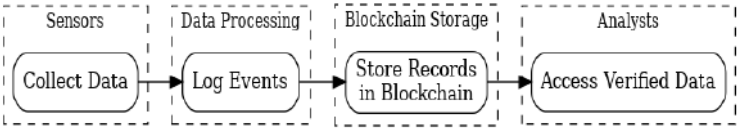

Figure 2.1. Digital trace formation scheme

The scheme reflects the sequence of data transformation: from the initial collection of information from sensors, through event logging and their storage in a secure blockchain repository, to granting access for analysts. Such an organization ensures the continuity, reliability, and immutability of the event chronology, which becomes the foundation for objective investigation of emergency situations. Secure logging and blockchain

technology are applied. Data from key sensors, control commands, and changes in software states are recorded in an append-only storage mode. This creates an immutable timeline of events that cannot be retrospectively edited or deleted. The digital trace becomes a "source of truth" for reconstructing an accident with millisecond accuracy, enabling analysts to identify not only what happened, but also in what sequence the failure developed [58].

Thus, the digital trace forms the foundation for post-factum analysis. However, its role is not limited to reconstructing past events. The logic of modern approaches suggests that if data reliability can be guaranteed after an accident, it is equally important to use this data for pre-accident analysis, in order to detect precursors of impending failures.

Catastrophic failures rarely occur instantaneously. They are almost always preceded by weak, barely noticeable anomalies. Traditional analysis methods register the already accomplished fact, whereas the modern approach requires shifting the focus toward detecting early signs of malfunction. For this purpose, time-series analysis algorithms and machine learning methods are applied. A special role is played by recurrent neural networks such as LSTM (Long Short-Term Memory), which are capable of recognizing long-term dependencies and capturing hidden deviations preceding known failures [59]. Constructing a "normal state profile" of the system and tracking deviations from it makes it possible to move from reaction to prevention. Analyzing data hours or even days before an incident helps answer the question: *"Which early signs did we miss?"* and transform accumulated experience into a tool for preventing future accidents.

Figure 2.2 visualization of anomaly detection using an LSTM model.

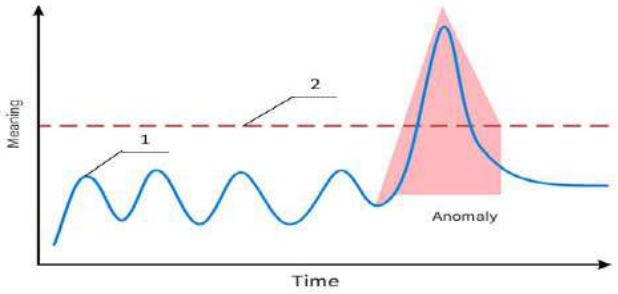

Figure 2.2. Visualization of anomaly detection using an LSTM model

Line 1 reflects the normal behavior of the system, line 2 denotes the threshold value, and the shaded area indicates a deviation—an anomaly that signals the onset of degradation. This approach makes it possible to model the system's dynamics and promptly highlight potentially dangerous trends that precede failure.

In this way, the digital trace transforms from a passive archive into an active source of knowledge, enabling predictive analysis. However, even anomaly detection does not always answer the key question: *Why did the failure occur, and which specific event initiated it?* Addressing this requires the next level of analysis-automated root cause identification.

From predictors to Root Cause Analysis (RCA) with AI

Traditional RCA relied on lengthy manual examination of logs and telemetry. In the case of large-scale incidents, this process could take weeks, and in critical systems, it led to enormous losses. The integration of artificial intelligence methods makes it possible to radically accelerate this process and increase its objectivity.

Modern ML models are designed not to search for a "needle in a haystack," but to systematically uncover cause-and-effect relationships. To achieve this, the following are applied:

- clustering methods that group events by temporal and functional proximity;
- graph models that construct dependency networks among thousands of events;
- hypothesis ranking algorithms that highlight the most probable initial event [60].

Experts receive not a raw mass of data, but a structured set of hypotheses with an indication of their probability. This transforms the investigation from an art into a data-driven, formalized procedure, significantly reducing system downtime.

Figure 2.3. Example of a cause-and-effect graph generated during the RCA process. The nodes of the graph represent individual events, while the arrows denote their dependencies.

The primary event ("Root Cause") initiates a sequence of subsequent events (Event1–Event6), ultimately leading to the accident. Visualization of this kind helps experts quickly localize the initial failure point and trace the entire chain of its propagation, which significantly accelerates the investigation and increases the reliability of conclusions.

Comparison of approaches

For clarity, Table 2.1 compares traditional and modern approaches to accident analysis in cyber-physical systems (CPS). The comparison shows that in the traditional paradigm, the main focus was on post-event investigation and localized engineering interpretation of causes, whereas

modern methods are oriented toward predictiveness, integration with digital twins, and the use of artificial intelligence to reveal hidden patterns.

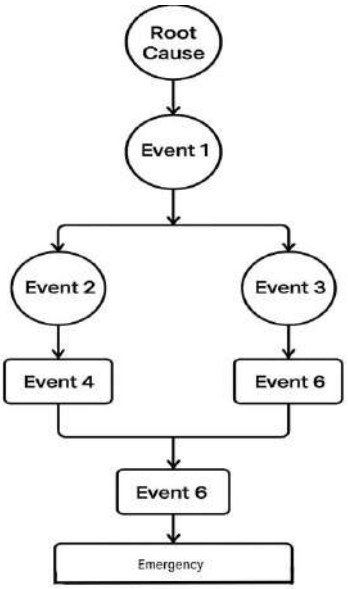

Figure 2.3. Cause-and-effect graph

Table 2.1

Comparison of traditional and modern accident analysis in CPS

Aspect	Traditional approach	Modern approach
Temporal focus	Post-factum (after the accident)	Before, during, and after (prediction + investigation)
Data	Logs, journals	Telemetry streams, IIoT, digital trace
Nature of analysis	Manual, expert-based	ML/AI, automated
Result	Event reconstruction	Prevention + root cause identification

The data in Table 2.1 illustrate the shift from reactive accident management toward a proactive failure-prevention strategy. This confirms that modern approaches transform accident analysis from a localized engineering task into an integral component of comprehensive risk and reliability management in cyber-physical systems.

To systematize the discussed directions - digital trace, predictor analysis, and RCA with AI - Table 2.2 provides a comparative overview of methods and tools for accident analysis. The table highlights which technologies form the foundation of each approach and the results they deliver: from improving diagnostic accuracy to expanding predictive capabilities and enhancing the explainability of failure causes.

The data presented in Table 2.2 demonstrate that the integration of digital trace, predictor analysis, and AI-driven RCA does not constitute a set of isolated tools but rather a complementary system in which each technology reinforces the previous one. This confirms the transition from fragmented analysis to comprehensive intelligent decision support in emergency situations.

Table 2.2

Methods and tools for accident analysis

Approach	Applied technologies	Expected outcome
Digital trace	Logging, blockchain	Immutable chronology of events
Predictor analysis	LSTM, time series analysis	Early anomaly detection
RCA with AI	Clustering, graph models	Rapid root cause identification

Thus, the three sequential stages - creation of the digital trace, identification of failure predictors, and automated RCA — form a holistic methodology for modern accident analysis in complex technical systems. Unlike traditional approaches, this methodology goes beyond retrospective reconstruction: it provides proactive prevention, accelerates investigation, and improves the accuracy of conclusions. The methodology is most fully realized within the framework of a digital twin, which integrates individual technologies into a unified system for decision support and reliability management.

2.2 Concepts and architectures of data management in complex technical systems

A characteristic feature of modern trends in computer analysis, data interpretation, and decision support is the intensive development of technologies and tools for extracting knowledge from data. A special place is occupied by data mining (Data Mining, DM), which represents an evolution of statistical and information technologies made possible by the progress of hardware and software, as well as the widespread availability of powerful and affordable computing and data storage systems.

Classical concepts and methods of data processing

Traditional concepts of data storage and analysis include interrelated components: data warehouses (Data Warehouse); on-line analytical processing (On-Line Analytical Processing, OLAP); data mining (Data Mining, DM).

The technology of data warehouses implies preliminary data cleaning and integration, as well as their structuring for subsequent analysis. OLAP technologies provide multidimensional analysis, including consolidation, aggregation, summarization, and presentation of information from different perspectives. However, despite advanced analysis tools, OLAP is limited mainly to data aggregation and visualization. For deeper analysis, additional methods are required: classification, clustering, studying the dynamics of changes, and forecasting.

The applied technology of data mining is the result of the natural evolution of information technologies, determined by the progress of hardware and software and the emergence of powerful and affordable computers and data storage devices. This contributed to the development of the information technology industry and made numerous databases and information repositories available for managing data extraction transactions and data analysis [57, 61].

The most frequently solved tasks of data mining include:

- identification of significant factors and dependencies;
- reduction or expansion of the number of analyzed features;
- detection of exceptions, associations, and hidden patterns;
- classification, forecasting, modeling;
- segmentation and ranking of objects.

The methods used to solve these tasks cover a wide range: decision trees, neural networks, statistical methods, fuzzy logic algorithms, evolutionary and genetic algorithms, case-based analysis methods, variational modeling, and combined approaches. From a theoretical point of view, many data mining methods are based on statistical approaches: correlation, regression, and variance analysis. However, the main limitation of the statistical approach is the averaging of values, which reduces the informativeness of data and decreases the amount of extracted knowledge.

It is important to note that data mining is applicable in almost all areas where the task of automatic data analysis arises: from processing medical images and analyzing market trends to monitoring the functioning of potentially hazardous objects. For processing data with a high degree of uncertainty or with elements of linguistic fuzziness, methods of probability theory, classical and fuzzy set theory, as well as extended approaches of artificial intelligence, are used. A special role in modern systems is played by neural networks, which make it possible to solve a wide range of data

analysis tasks, provided that the architecture and structure of the network are chosen correctly.

Despite significant achievements, classical concepts of data management (Data Warehouse, OLAP, Data Mining) cannot cope with the scale, speed, and variety of data generated by modern cyber-physical systems. These data include telemetry, sensor signal streams, log files, images, CAD models, textual documentation, and many other types of information. It is precisely this problem that predetermined the transition to new architectural concepts of data processing, focused on integration, scalability, and flexibility.

In Fig. 2.4, the evolution of data management architectures is shown — from classical data warehouses (Data Warehouse) to DTs. Each stage is characterized by an increase in scalability, flexibility, and the degree of data integration.

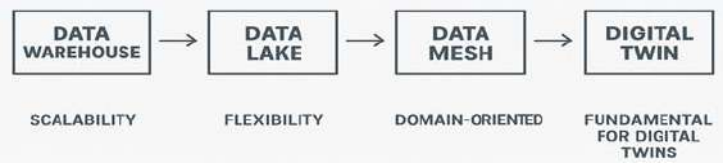

Figure 2.4. Evolution of data architecture: from data warehouse to digital twin

Modern concepts of data processing for digital twins

1. **Digital thread**

The Digital thread is an integration platform that unites data about all aspects of an asset's lifecycle - from design and manufacturing to operation and disposal [62]. Unlike isolated "digital islands" (CAD, CAE, MES, ERP, IoT), the Digital thread ensures continuous traceability and bidirectional synchronization of data between the physical object and its digital twin [63].

2. **Big data storage and industrial data mesh**

A Data lake is a centralized repository of large volumes of heterogeneous data in its raw form. The "schema-on-read" principle assumes that data is first stored in its original format and then structured according to analytical tasks [64]. The main problem of a Data Lake is the risk of turning into a "data swamp," when data becomes unstructured and difficult to use.

Data mesh represents a modern decentralized data management architecture oriented toward domain ownership ("data as a product"). Its key principles include: domain-oriented data management; self-service data platforms; federated governance of quality and security.

For industry, this means that, for example, vibration monitoring data (domain "Diagnostics") can be used by the "Maintenance" domain to predict the remaining useful life of equipment [65].

A comparative characteristic of the architectures is presented in Table 2.3

Table 2.3

Comparison of data architectures

Characteristic	Data warehouse	Data lake	Data mesh
Purpose	Storage of structured data for reporting and BI	Storage of any data in raw form	Data management as domain products
Data type	Structured	Structured, semi-structured, unstructured	Any
Data schema	Schema-on-write	Schema-on-read	Domain models
Scalability	Medium	High	Very high (decentralization)
Main problems	Rigidity, limited flexibility	"Data swamp," search difficulties	Coordination complexity, requirements for data culture

As can be seen, the Data Warehouse is oriented toward strictly structured data and reporting, the Data Lake provides flexibility but may lead to "data swamps," whereas the Data Mesh ensures scalability and domain ownership, which is especially important for digital twins.

Industrial Data mesh:

This is a modern decentralized socio-technical data management architecture. Data Mesh solves the problem of "data swamps" by organizing data not around technology but around business domains (for example, "Engine," "Control System," "Maintenance") [65].

- Principles:

1. Domain-oriented data ownership: the team responsible for the engine is also responsible for its data (as products - "data-as-a-product");
2. Self-service data platforms: the central team provides a platform (based, for example, on Kubernetes and Spark) so that domain teams can easily publish, discover, and use their data;
3. Federated governance: unified standards of security, quality, and data interoperability that are followed by all domains.

- For industry: This means that data from a vibration control system (domain "Diagnostics") can be easily and securely consumed by a residual life prediction model (domain "Maintenance").

Single source of truth (SSOT): the foundation for trust. The SSOT concept assumes the creation of a unified, reliable, and up-to-date representation of critical data, which is used by all consumers within the organization [66]. For CPS, this means creating an authoritative source of data on the configuration, condition, and history of each asset.

- Importance for AI/ML: machine learning models trained on contradictory or poor-quality data produce unreliable and dangerous

predictions. SSOT ensures the quality and consistency of the data on which digital twin models are trained and operate.

- Implementation: SSOT does not mean one physical repository. It is a logical concept provided through:

- standardized processes of data entry;
- protocols for data quality assurance and cleansing (Data Quality, Data Cleansing).
- Use of master data (Master Data Management - MDM) for unification of reference information (for example, which specific sensors are installed on the asset and what their characteristics are).

Thus, the SSOT concept can be represented as a central core that ensures the reliability and consistency of data, which is accessed by various consumers - from enterprise systems (ERP) and IoT platforms to analytics and machine learning models.

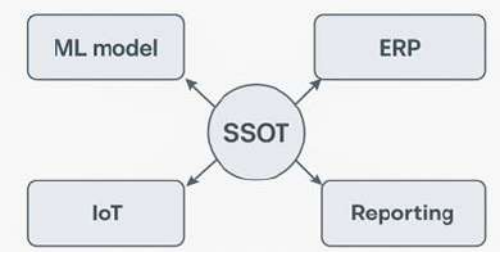

Figure 2.5. Concept of single source of truth (SSOT): central core and data consumers

The absence of a single source of truth leads to significant risks, summarized in Table 2.4.

Table 2.4

The absence of a single source of truth leads to significant risks

Risk	Manifestation	Impact on DT
Data inconsistency	Different departments use different versions of data	Erroneous forecasts and simulations
Lack of trust	Lack of trust in data between teams	Reduced efficiency of DT usage
AI/ML errors	Models are trained on “dirty” data	Incorrect decisions, emergency situations
Increased costs	Duplication of data and analytics	Increased total cost of system ownership

The implementation of SSOT is a critically important condition for ensuring the reliability of the digital twin and the adequacy of the results obtained during its operation.

Thus, the transition from classical data management systems (Data Warehouse, OLAP, Data Mining) to modern concepts (Digital Thread, Data Lake, Data Mesh, SSOT) reflects the evolution of approaches to storing, integrating, and using information. These concepts create the foundation for building digital twins and ensuring the reliability of cyber-physical systems.

2.3 Architectures and software platforms for stream big data processing in complex technical systems

With the expansion of the functionality of developing CTS, the requirements for the reliability of their operation are increasing [61, 67]. This is accompanied by the growth in the number of information sources and the volume of computational tasks for processing Big Data coming from information-measuring systems (IMS). Effective processing of these data is necessary to maintain the operability of CTS and to reduce the risk of failures [68, 69].

For the collection and processing of data from IMS of CTS, as well as for the management of technological processes in them, SCADA systems operating in real time are used [70]. However, the topological complexity of such systems is associated with significant costs, as well as with scaling and adaptation to the large number of information-measuring signals collected for the reconfiguration of the CTS control structure.

It should also be noted that most SCADA systems are used mainly to provide an overview of controlled processes in CTS with the ability to perform Process Analyzer methods in order to analyze the state of systems, primarily for statistical data processing. In this regard, new data processing technologies and methods of Big Data analysis for this field are becoming more in demand.

Big Data processing methods [71, 72] provide for the analysis of large datasets at terabyte or petabyte scale in real time. A complex task is the rapid (with low latency) analytics of the full volume of Big Data. This means the need to scan terabytes of data within seconds, which is possible only with highly parallelized data processing. Based on the analysis of the interpretation of information flows in distributed Big Data information systems, it has been established that when creating such systems, there are no unified methods and technologies that combine all stages of constructing the corresponding cluster systems.

One of the architectures for Big Data processing that uses search algorithms is the relational database management system (DBMS). The disadvantages of applying such DBMSs led to the development of an

adaptively adjustable architecture, capable of expanding and scaling when necessary and with the constant increase of data.

This technology includes NoSQL [73]. The use of NoSQL architectures as systems for storing and processing data is not always optimal, which is connected with the speed and performance limitations of hard drives. A partial solution to these problems has been found in the search architecture - the Hadoop ecosystem.

The Apache Hadoop framework [74, 75] is an open-source environment in Java for the development and execution of distributed programs operating on computing clusters consisting of hundreds and thousands of nodes. The basic modules of Apache Hadoop [76] are:

- Hadoop Common - a set of infrastructure software libraries and utilities used for other modules and related projects;
- HDFS (Hadoop Distributed File System) - a distributed file system;
- Hadoop YARN - a cluster resource management module for executing user applications;
- Hadoop MR - a programming model for Big Data processing.

Examples of projects included in the Hadoop ecosystem are Apache Hive, Apache Pig, Apache HBase, Apache Spark, and others. HDFS is built on a master–slave architecture, where the *Name Node* is the master, and the *Data Nodes* are the slave nodes that store the actual data (Fig. 2.6). To access data in the Hadoop storage, the SQL-like language Hive is used, which is a kind of SQL for MR [77].

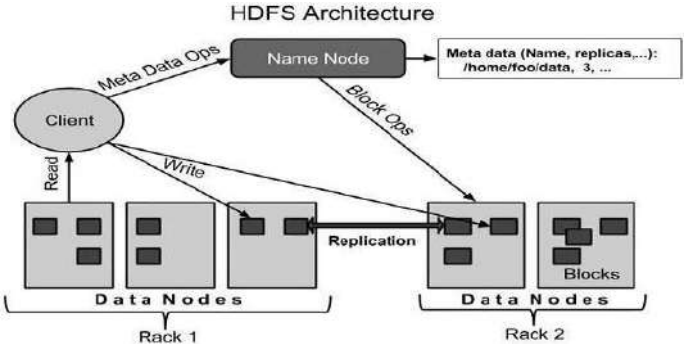

Figure 2.6. Architecture of the hadoop file system

The standard use in the Hadoop architecture is the application of MR tasks (Fig. 2.7). MR corresponds to the functional programming model [78] and performs explicit synchronization at the stages of computation.

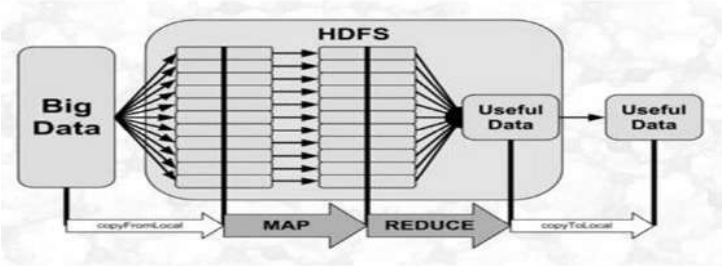

Figure 2.7. MapReduce operations

MR provides a simple programming API in terms of the map() and reduce() functions [79]. Apache Hadoop MR is an open-source platform for organizing the processing of Big Data at the petabyte scale using the easily scalable MR paradigm, which is a fault-tolerant solution. MR operates in real time with streaming Big Data, applying algorithms that were not originally designed for distributed processing across multiple nodes without changing their implementations. Hadoop MR allows the creation of jobs with both basic mappers and reducers written without the use of Java. The Hadoop streaming utilities use any executable file that works with the standard input-output of the operating system as mappers and reducers. There is also a SWIG-compatible application programming interface, Hadoop Pipes, in C++. The limitation of MR is that it assumes batch execution, in which significant volumes of data entirely pass through the transformation chain, taking much time, which is unsuitable for interactive responses to users.

For fast real-time analysis of streaming data, Apache Spark is also used [79, 80] - a universal tool for Big Data analytics that makes it possible to process data stored on Hadoop, Cassandra, Mesos, S3, etc. Apache Spark provides a stack of libraries, including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. Compared to Hadoop MR, Apache Spark provides up to 100 times higher performance when processing data in memory and 10 times higher performance when data is stored on disks. Apache Spark offers a user-friendly programming interface to reduce coding efforts by using the concept of Resilient Distributed Data (RDD). Apache Spark can be used interactively from Scala, Python, Java, R, and SQL shells. Figures 2.8 and 2.9 show iterative operations on Spark RDD with intermediate results stored in distributed memory instead of persistent storage (disk).

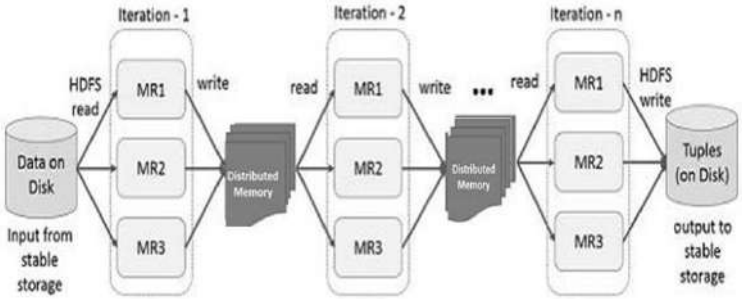

Figure 2.8. Iterative operations with apache spark RDD

The advantages of Apache Spark compared to Hadoop MR are as follows: high scalability through the addition of new nodes to the computing cluster without the need to modify the applied algorithms; built-in capability of operating in real-time mode, which makes it possible to design algorithms for streaming data processing; and a large number of auxiliary software solutions required to organize a system that supports the full cycle of domain-specific tasks. A comparison of distributed computing technologies makes it possible to choose in favor of Apache Spark.

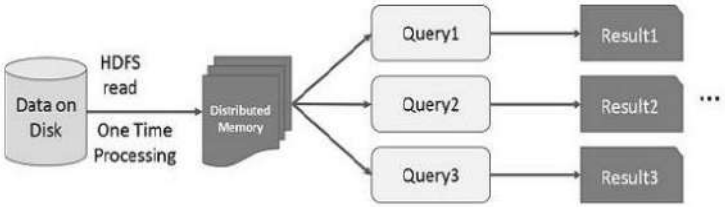

Figure 2.9. Interactive operations with Apache Spark RDD

For the purposes of Big Data analytics, continuously incoming from the information and measurement systems (IMS) of CTSs, as well as for managing technological processes within them, it is possible to use cloud computing infrastructure. For these purposes, process analyzer (PA) methods may be applied, creating an environment in which almost all stages of CTS operation can be recorded and used not only for system security, but also for optimizing the process of Big Data analysis in real time.

The use of cloud computing for processing Big Data opens up opportunities for a significant reduction in the costs of Big Data analytics

received from CTS measuring devices. It is necessary to take into account how data are processed and managed across different clusters, which have a significant impact when designing and operating control and data processing systems at multiple levels. The most challenging task is the rapid examination of the entire large dataset, which is possible through its preparation with high parallelism. Figure 2.10 shows how accessible and maintainable data through the Internet are processed via a web server in the form of MySQL or NoSQL queries, and then back-end level data analytics is performed on Apache Spark via HDFS. In the cloud architecture, a distributed message broker Apache Kafka is used - an open-source project written in the Scala programming language, enabling streaming information to be sent (or received) through the cloud.

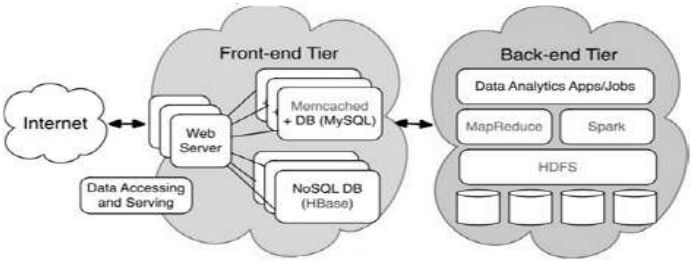

Figure 2.10. Data Access and maintenance via the Internet

Figure 2.11 presents the interaction of Apache Kafka and the Spark cluster during the transmission of real-time data streams on the state of CTS with the Kafka service engaged in the cloud.

In the cluster, data is checked, cleaned, aggregated, organized, and directed into the optimal control system of the CTS with the determination of appropriate recommendations for its operation. When transferring data to the cloud, a PC connected to the technological process of the CTS is used, based on the application of MefosService, which makes it possible to perform synchronization of data from the CTS elements, as well as to create a file in the JSON structure with fields, the number of which corresponds to the number of CTS elements. Input data are processed in the Kafka server. Spark receives measurement data from the Kafka server, stores them in memory, and transfers the existing models of technological processes in the STS elements at a specified time interval. The process of data transfer is shown in Fig. 2.12.

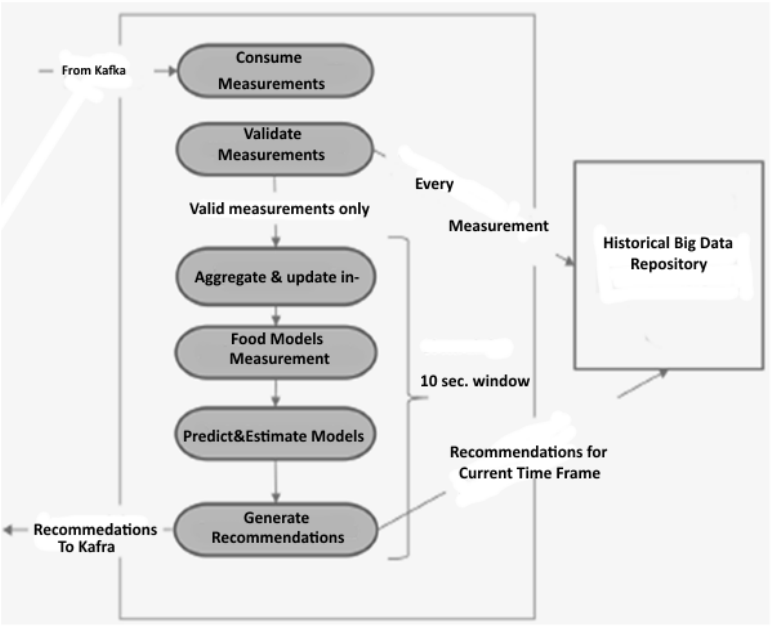

Figure 2.11. Overview of the streaming process

In the process of Spark-Streaming, metadata are synchronized and pre-processed and output from the MefosService PC into Kafka, and from there into the Spark cluster. In Spark-Streaming, raw data are accumulated in memory and stored in the data repository. Recommendation data for control are also accumulated in memory and stored in the Big Data repository.

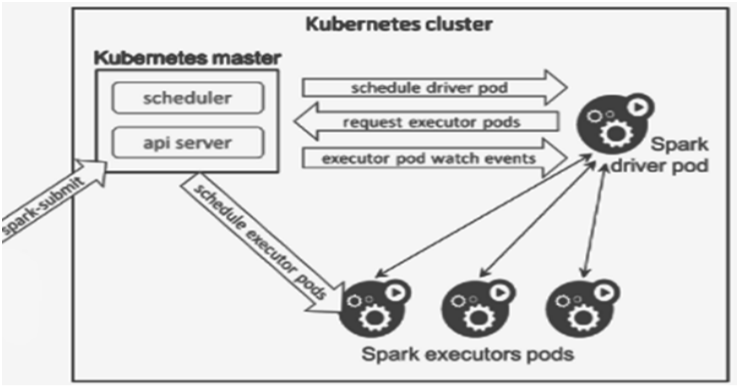

Figure 2.12. Apache Spark 2.3 with Kubernetes cluster support

It is recommended to use Apache Spark with Kubernetes cluster support (Fig. 2.12). Considering that Kubernetes is the de facto standard for managing containerized environments, it is quite natural to support the Kubernetes API interface in Spark.

To view the Apache Spark resources created in the cluster, the kubectl command can be used in a separate terminal window:

$ kubectl get pods -l 'spark-role in (driver, executor)' -w
NAME READY STATUS RESTARTS AGE
sparkpi-driver 1/1 Running 0 14s
spark-pi-da1968a859653d6bab93f8e6503935f2-exec-1
0/1 Pending 0 0s

A comparative analysis of the characteristics of two file systems - Hadoop MapReduce and Apache Spark MapReduce and Spark - made it possible to choose the Apache Spark system for streaming Big Data processing.

Modern distributed computing platforms have evolved from tools for batch processing and ETL procedures into a universal infrastructure for working with data and machine learning in real time. This makes them a critical component of the DT ecosystem, providing not only storage and transformation, but also continuous learning and model updating.

1 Shift of focus: from processing to learning (From ETL to Continuous Learning)

The classical use of Hadoop/Spark for batch processing of historical data (ETL) is shifting towards the support of end-to-end ML pipelines. These pipelines include the stages of feature engineering, model training, its evaluation, and deployment into production.

Role in the DT ecosystem: Platforms like Apache Spark with its MLlib library provide distributed machine learning algorithms capable of training on terabytes of data collected from CTS sensors [81]. This makes it possible to create and periodically retrain accurate models for predicting Remaining Useful Life (RUL) or classifying states, which form the core of the predictive capabilities of the DT.

Importance: Without such infrastructure, training on Big Data would be impossible or would take an unacceptably long time, making the DT "static" and unable to adapt.

2 Modern frameworks for stream processing: Kafka and Flink

To update the DT in near real-time, specialized stream processing frameworks are critically important.

Apache Kafka: acts as the "central nervous system" or reliable backbone for data transfer. It is a distributed publish–subscribe messaging platform

designed for real-time data stream processing with high throughput and fault tolerance [82].

Role for the DT: Kafka continuously ingests raw data from thousands of IIoT sensors and delivers them to all subscribers: monitoring systems, storage databases, and, most importantly, stream processing engines like Apache Flink for immediate analysis.

Apache Flink: a stateful stream processing engine that supports complex computations over infinite data streams with guaranteed delivery and exactly-once semantics [83].

Role for the DT: While Spark processes data in micro-batches, Flink works with truly continuous streams. This allows, in real time:

- computing moving averages and detecting anomalies;
- updating DT model states based on the latest incoming data;
- performing online learning of certain types of models, enabling the DT to adapt to concept drift in the data.

Let us summarize the key characteristics of the main frameworks that form the foundation of the Digital Twin infrastructure. Spark provides batch processing and model training, Kafka serves as a reliable channel for streaming data delivery, and Flink implements continuous processing and adaptive learning. Comparative characteristics are presented in Table 2.5

Table 2.5

Comparison of Spark, Kafka, Flink frameworks for DT

Characteristic	Apache Spark	Apache Kafka	Apache Flink
Main role	Batch ML, ETL	Streaming data delivery	Stateful stream processing
Data type	Historical (batch)	Real-time streams	Real-time streams
Strengths	MLlib, scalability	Reliability, high throughput	Exactly-once semantics, online learning
Role for DT	Model training	Sensor integration, backbone	State updates and adaptation

3. Architectural Patterns: Lambda and Kappa

To build fault-tolerant systems that combine real-time processing and deep analysis of historical data, two key architectural patterns have been developed.

Lambda architecture: this pattern assumes the creation of two separate layers for processing the same data [84]:

1. Speed Layer: Processes streaming data in real time (using Flink, Storm) to obtain fast but possibly approximate results. For DT: instant alerting about critical anomalies;

2. Batch Layer: Processes all historical data with a delay (using Hadoop, Spark) to obtain accurate and complete results. For DT: daily retraining of the main predictive model;

3. Serving Layer: Combines the results of both layers to respond to queries.

Disadvantage: the complexity of maintaining two different codebases and infrastructures. In modern conditions, traditional data management architectures (Data Warehouse, Data Lake) do not always provide the necessary speed and flexibility of information processing, especially in the operation of CTSs, generating large and heterogeneous data streams. To solve this problem, the Lambda Architecture has become widespread, combining batch and stream processing, ensuring a balance between reliable storage and the possibility of real-time analysis (Fig. 2.13).

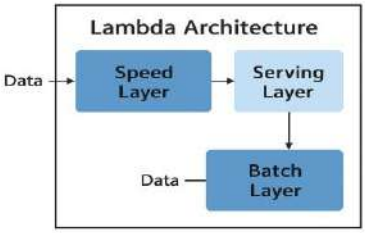

Figure 2.13. Lambda Data Processing Architecture

As shown in Fig. 2.13, the Lambda Architecture includes three layers:

- Batch Layer - storage and periodic processing of large volumes of historical data, formation of reference models and aggregated representations;

- Speed Layer - real-time data processing, identification of events and anomalies requiring immediate response;

- Serving Layer - integration of batch and stream processing results to provide comprehensive information to users and application services.

For analyzing emergency situations and predicting failures in CTS, this approach is especially valuable: batch processing ensures accounting for long-term trends, while stream processing allows instant response to emerging deviations. Thus, the Lambda Architecture becomes an important tool for building reliable Digital Twins and intelligent monitoring systems.

○ Kappa Architecture: A simplified and more modern alternative, proposed by Kafka creator Jay Kreps [85]. In this architecture, there is only one layer – streaming. Principle - all data is treated as a stream. Historical data is replayed from long-term log storage (often Kafka with long retention)

through the same streaming engine (Flink) for recalculation of historical metrics or retraining models. Advantage for DT - simplifies the architecture by eliminating the need for two separate codebases. Allows the same logic to be used for processing both real-time and historical data, ensuring consistency.

The Digital Thread demonstrates the interconnection of all stages of the product lifecycle - from design to disposal. However, the implementation of such end-to-end integration is impossible without a reliable computing infrastructure capable of processing both streaming and historical data. For this purpose, modern systems employ Lambda and Kappa architectural patterns, providing a balance between real-time analytics and deep processing of accumulated data sets. Their comparison is shown in Fig. 2.14.

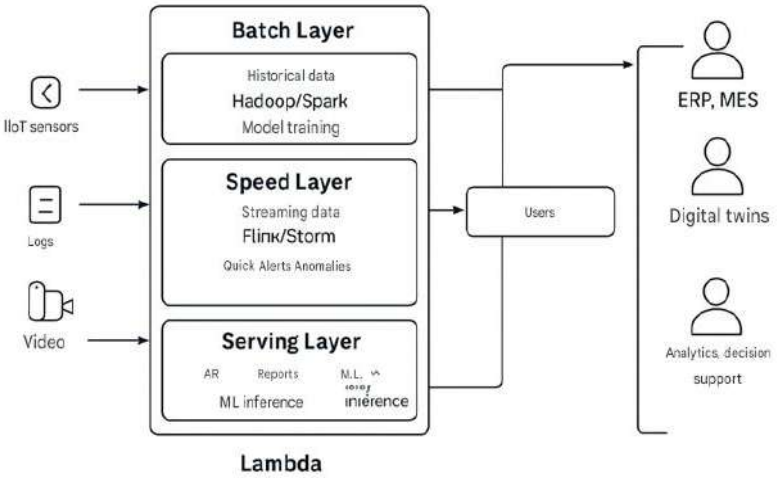

Figure 2.14. Architectural data processing patterns

Figure 2.14 demonstrates the Lambda data processing architectural pattern, representing a hybrid approach that combines batch and stream data processing. This architecture was proposed to address tasks related to Big Data, where both high computational accuracy and minimal response latency to incoming events are simultaneously required. The architecture highlights three functional layers:

- batch layer provides long-term storage and processing of historical data. At this level, distributed computing platforms (Hadoop, Spark) are used, enabling repeated processing of accumulated information arrays. Reference data sets are formed, machine learning models are trained, and analytical

data marts are built. The main function of the Batch Layer is to ensure high accuracy of analysis and the capability for large-scale reprocessing of data;

- speed layer is designed for real-time data processing with minimal latency. Specialized streaming systems (Flink, Storm) implement anomaly detection, generation of rapid alerts, and responses to critical changes. This layer compensates for batch processing delays and ensures timely reaction to rapidly changing events;

- serving layer integrates the results of batch and stream processing and provides unified data access for users. At this level, APIs for external applications are implemented, reports are generated, and ML inference of trained models is performed. The Serving Layer serves as a link between the processing system and the end consumers of data.

Data users within the Lambda architecture include enterprise management systems (ERP, MES), DTs of assets, as well as analytical platforms providing predictive analytics and decision support. Thus, the Lambda architecture provides a comprehensive solution: on one hand, reliable and accurate analytics through batch processing, and on the other hand, high response speed to incoming signals via stream processing. Taken together, this makes it applicable in industrial, transport, and corporate systems, where both the reliability of long-term analytics and operational responsiveness to current changes are required.

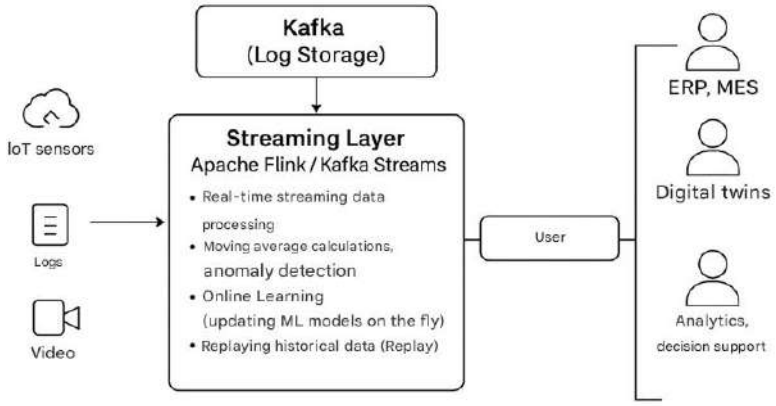

Figure 2.15. Kappa architecture for data streams

The scheme presented in Figure 2.15 illustrates the key elements of the Kappa architecture, designed for unified stream data processing. Unlike the Lambda architecture, where two independent pipelines (batch and speed) are

used, the Kappa model relies on a single streaming processing layer (Streaming Layer). All incoming data enters a distributed event bus, which simultaneously serves both for current processing and for subsequent replay of historical data.

The central element of the architecture is a stream processing engine (e.g., Apache Flink or Kafka Streams), which performs event processing in real time. It implements operations such as filtering, aggregation, feature computation, and machine learning model updates. If recalculation or retraining of models is required, the same data can be replayed from long-term log storage (Kafka Log, HDFS), passing through the same computational logic.

The processing results are transferred to the Serving Layer, which provides access for analytical applications, monitoring systems, ML models, and business services to up-to-date data and forecasts. Thus, the Kappa architecture ensures:

- simplification of infrastructure - a single processing code instead of two parallel pipelines;
- flexibility and consistency - the same logic is applied both to real-time data and to historical streams;
- high adaptability - the ability to promptly update DT models when stream characteristics change or concept drift occurs.

Overall, the scheme in the figure demonstrates the transition from the redundant and more complex Lambda model to the elegant Kappa solution, focused on continuous learning and DT support in real time.

The scheme presented in Figure 2.15 illustrates the key elements of the Kappa architecture, designed for unified stream data processing. Unlike the Lambda architecture, where two independent pipelines (batch and speed) are used, the Kappa model relies on a single streaming processing layer (Streaming Layer). All incoming data enters a distributed event bus, which simultaneously serves both for current processing and for subsequent replay of historical data. The central element of the architecture is a stream processing engine (e.g., Apache Flink or Kafka Streams), which performs event processing in real time. It implements operations such as filtering, aggregation, feature computation, and machine learning model updates. If recalculation or retraining of models is required, the same data can be replayed from long-term log storage (Kafka Log, HDFS), passing through the same computational logic.

Both architectural patterns have their advantages and limitations. Lambda is suitable for complex scenarios with combined processing but complicates maintenance. Kappa simplifies the system by providing a unified processing stream for both historical and real-time data. A summarized comparison is presented in Table 2.6.

Table 2.6

Comparison of Lambda and Kappa architectures for supporting the DT

Criterion	Lambda architecture	Kappa architecture
Processing Layers	Batch + Speed + Serving	Streaming + Replay
Complexity	High (two codebases)	Lower (single layer)
Consistency	Requires synchronization of layers	Unified logic for all data
Suitable for DT	Long-term analysis + operational alerts	Unified processing and retraining

Thus, the choice of an architectural pattern is determined by the nature of the tasks and the requirements of the system. The Lambda model remains relevant for scenarios where a combination of long-term batch analytics and rapid reactions to streaming events is required, but it imposes additional overhead on maintenance and data consistency. The Kappa architecture, on the contrary, simplifies the construction of the computational infrastructure and proves to be especially in demand in digital twin applications, where continuous signal processing and the possibility of rapid model retraining are important. In general, both architectures can be considered as complementary tools that form the basis of the computational environment for intelligent management of the state of complex technical systems.

2.4 Modern information systems for monitoring and diagnostics: from SCADA to digital twin platforms

A common feature of the methods for diagnosing CTSs ystems in terms of obtaining and processing information is the use of procedures for identifying diagnostic parameters. According to [86], a diagnostic system includes three main elements: the diagnostic object, the technical means of diagnostics, and the operator. Taking into account this information, as well as the structure of information support in the decision support system, an architecture of information support in the diagnostic system can be proposed (Fig. 2.16).

For CTS, information monitoring systems (IMS) are applied, which ensure improved quality and efficiency of managing the operation of objects (Fig. 2.17). The IMS includes mathematical support—an interpreter of the values of controlled parameters into system state indicators, which makes it possible to analyze the dynamics of their functioning and to generate diagnostic conclusions.

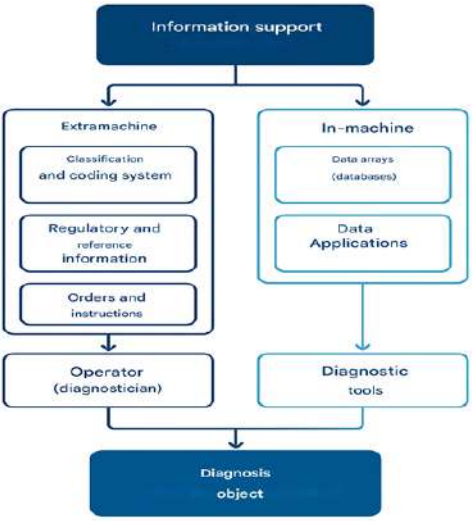

Figure 2.16. Structure of information support in the diagnostic system for transport (according to [57])

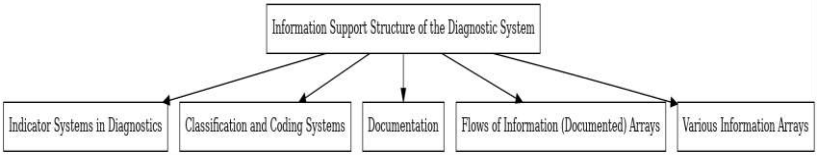

Figure 2.17. General scheme of information support for diagnostic systems (according to [86])

At the same time, with a sufficiently large response time of IMS to changes in the state of CTS and the occurrence of rare abrupt changes during this interval (for example, misadjustments or failures of individual elements), the controlled parameters may change in such a way that the system does not manage to adequately register these events. This indicates the limitation of traditional IMS, associated with significant response duration.

To overcome this contradiction between the tendency towards increasing complexity of CTS and the requirements to reduce the probability of their failures, it is necessary to improve IMS. First of all, this is related to the development of mathematical support, the reduction of

computation time, and the acceleration of processing of controlled parameters.

Evolution of IMS towards digital twin platforms.

Classical IMS performed the functions of data acquisition (SCADA) and their visualization (HMI) for the operator. Modern solutions are evolving into digital twin platforms (DT platforms), which represent active computational environments ensuring the full life cycle of predictive and diagnostic models.

The development of IMS towards DT platforms can be characterized by three key directions:

1. **Data ingestion and contextualization.** The platform provides not only the collection of raw data but also their cleansing, normalization, and enrichment with contextual information (for example, linking sensor readings to specific components or accounting for operating modes). This allows the transformation of an array of unrelated data into structured events suitable for use by models [87].

2. **Model execution and orchestration.** The DT platform is an environment for the execution and orchestration of heterogeneous models:

- physical models based on differential equations (for example, heat transfer models);
- data-driven models, for example, neural networks for predicting residual resource;
- hybrid models combining physical principles and machine learning methods (Physics-Informed ML).

The platform manages the life cycle of models: it loads them into memory, supplies current data, obtains predictions, and caches results [88]. Table 2.7 presents a comparison of the characteristics of the three main types of digital twin models.

Table 2.7

Comparison of the main types of digital twin models

Model type	Example	Features
Physical	Differential equations for heat transfer	High interpretability, requires process mathematics
Data-driven	Neural network for residual life prediction	High accuracy, low explainability
Hybrid	Physics-Informed ML	Compromise: accuracy + interpretability

3. **Visualization of results (advanced visualization & Human-in-the-Loop).**

Modern visualization tools are focused not only on displaying raw data but also on interpreting the results of model operation. This includes:

- prediction of residual resource in hours or cycles;
- explainable AI (XAI) - displaying factors that influenced the prediction;
- interactive "what-if" simulations, allowing the operator to set parameters and see the consequences.

Such mechanisms increase the expert's trust in the prediction and ensure the preservation of the human role in the decision-making loop [89].

To systematize the advantages of modern platforms, Table 2.8 provides a comparison of the functional capabilities of classical IMS and DT platforms.

Table 2.8

Comparison of the functional capabilities of classical IMS and digital twin platforms

Characteristic	Classical IMS	Modern DT platform
Data acquisition	Yes	Yes + cleansing and contextualization
Visualization	Simple HMI	XAI + predictions + what-if
Model execution	No	Yes (physical, ML, hybrid)
Architecture	Monolithic	Microservices + API-first

The architecture of the DT platform (Fig. 2.18) is built according to a multi-level principle. The lowest level represents the data layer (collection, cleansing, and normalization of information). The next level contains physical, data-driven, and hybrid models. Above it is located the orchestration and API layer, which ensures the interaction of components and external services. The top level forms the visualization layer, including tools for result interpretation (XAI), interactive dashboards, and simulation instruments.

The Role of API and microservice architecture.

Monolithic architectures of traditional IMS do not provide the required flexibility and speed of adaptation. Modern DT platforms are implemented based on the principles of microservice architecture and the API-first approach.

Microservice decomposition. Instead of a single application, the platform consists of independent services: data collection, execution of physical models, ML model inference, visualization, and notifications. This makes it possible to scale and update individual components without shutting down the entire system [90].

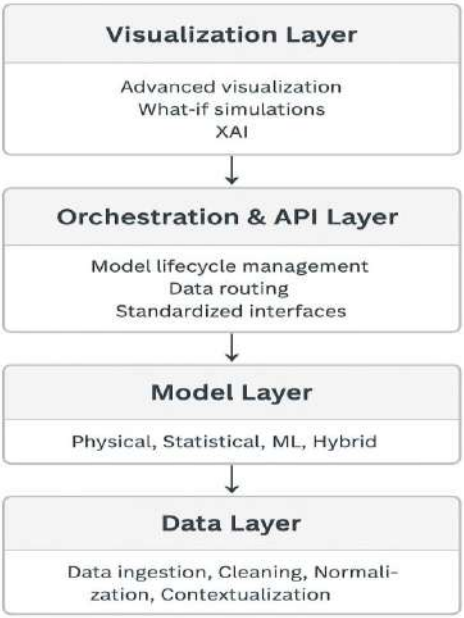

Figure 2.18. Architecture of the DT platform

API as universal contracts. Interaction between services and with external systems (ERP, CMMS, etc.) is implemented through formalized interfaces (REST, GraphQL, etc.). This ensures interoperability, flexibility, and the possibility of integrating third-party solutions [91].

Figure 2.19 presents the microservice architecture of the digital twin, including key services and API gateways for integration.

The differences between the monolithic approach and the microservice architecture of the digital twin are reflected in Table 2.9. It demonstrates the advantages of decomposition into independent services and the importance of API as a universal interaction interface.

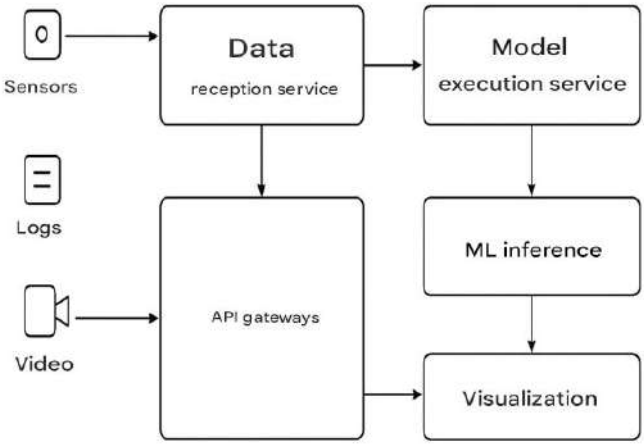

Figure 2.19. Microservice architecture of the digital twin platform

Table 2.9

Comparison of Monolithic and Microservice Architectures

Aspect	Monolith	Microservices
Scalability	Limited	Independent scaling of services
Flexibility	Low	High, easy to add new models
Interoperability	Problematic	High (API-first, REST/GraphQL)

Thus, the transition from classical IMS to digital twins provides not only a reduction in response time and an increase in diagnostic accuracy, but also forms a fundamentally new architecture for managing complex technical systems, combining predictive and diagnostic functions.

2.5 Integration of diagnostic information into forecasting and condition management of complex technical systems

Analysis of domestic and foreign works on the management of CTSs shows that, at present, their operation is predominantly based on a "rigid" maintenance strategy, relying exclusively on a priori data and scheduled repair intervals. Although this approach ensures a certain level of reliability, it does not take into account the actual condition of the equipment and, therefore, is unable to flexibly adapt to the dynamics of its degradation.

A more promising direction in this field is represented by “flexible” strategies for managing the technical condition of potentially hazardous facilities. They are based on the analysis and assessment of the actual condition of the system using diagnostic data, forecasting its changes during operation, evaluating residual life, and dynamically adjusting the parameters of the maintenance program. Such an approach requires the presence of a scientific and methodological framework that, on the basis of reliable diagnostic information about the existence of prerequisites for emergency situations, makes it possible to form timely management decisions aimed at preventing possible extraordinary events.

A key element of such a framework is the forecasting of the dynamics of changes in the hazard level of an object and the selection of adequate compensatory measures. Modern forecasting methods are based on the application of a wide range of mathematical tools: functional analysis, probability theory and mathematical statistics, the theory of random functions and processes, spectral analysis, as well as pattern recognition methods. The choice of a specific method is influenced both by the set of available parameters and by the target orientation of the task and the characteristics of the working algorithm.

With the development of DM technologies, software systems have emerged that provide the functioning of DSS. For effective analysis, such systems must have tools for accumulating, inputting, and storing information.

However, the continuous accumulation of diagnostic and operational data leads to an avalanche-like growth in their volumes. This imposes special requirements on DSS, which must guarantee reliable storage, high processing speed, and the ability for intelligent interpretation of information. As a result, research in the field of building intelligent DSS has intensified, where, according to the degree of “intelligence” in data processing, three classes of tasks are distinguished: information retrieval, operational-analytical, and intelligent.

A generalized architecture of a DSS can be represented in the form of three interconnected subsystems. The data input subsystem (OLTP — Online Transaction Processing) provides operational transactional processing, usually implemented by means of traditional database management systems (DBMS). The storage subsystem is built on the concept of data warehouses. The analysis subsystem can operate in three modes:

- information retrieval analysis, based on relational DBMS and SQL queries;

- operational-analytical analysis, implemented using OLAP (On-line analytical processing) technologies and multidimensional data representation;

- intelligent analysis (data mining), implementing DM methods and algorithms.

Thus, diagnostic data within the framework of digital infrastructure cease to be a passive source of information and turn into a basis for the formation of concrete prescriptive actions that close the management cycle. The evolution of maintenance methods shows that simple resource forecasting is no longer sufficient - it is necessary to form recommendations and action scenarios that make it possible to increase operational efficiency. In this context, a direct connection with Prescriptive Maintenance (RxM) arises.

Prescriptive maintenance (RxM) represents the highest maturity level in the evolution of maintenance strategies, following Reactive, Preventive, and predictive maintenance (PdM). While predictive maintenance answers the question “When will a failure occur?”, prescriptive maintenance (RxM) provides answers to the questions “What to do?” and “Why is this particular decision optimal?” [92].

The role of diagnostic data in this paradigm is crucial. Vibration characteristics, temperature indicators, and process technological parameters all serve as the raw material for building digital twins, which allow for:

1. Predicting (PdM): assessing residual useful life (RUL) and failure probability;

2. Simulating: conducting scenario analysis (“what-if”), modeling the consequences of different actions (for example, “what if the load is reduced by 15%?” or “what if the maintenance is postponed by 48 hours?”);

3. Optimizing: using optimization methods and artificial intelligence (including reinforcement learning, RL) to generate a set of recommended actions, ranked according to cost, risk, resource availability, and impact on production plans [93].

Figure 2.20 presents the multi-level architecture of a digital twin platform, enabling the transition from predictive to prescriptive maintenance.

The architecture includes four levels:

- data layer - collection of diagnostic and operational information from sensors and production processes;

- digital twin model layer - implementation of predictive models, conducting simulations, and performing optimization calculations;

- recommendation layer - transforming modeling results into prescriptive actions considering cost, risk, and production constraints;

- execution layer - implementation of decisions by an operator or an automated control system.

Thus, the platform architecture closes the management loop, turning diagnostic data into concrete prescriptions, which minimizes the risk of failures and optimizes the maintenance strategy.

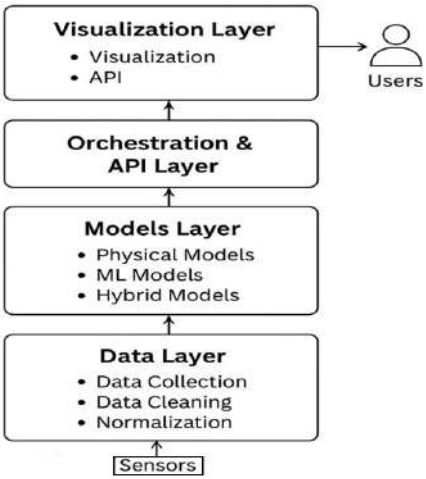

Figure 2.20. DT platform architecture (multi-level scheme)

Table 2.10

Optimization Criteria and Their Influence on Decisions in RxM

Criterion	Question	Influence on decision
Cost	How much will maintenance or failure cost?	Minimization of expenses while ensuring reliability
Risk	What is the probability and consequence of failure?	Priority elimination of critical risks
Downtime	How will downtime affect production?	Selection of repair timing with minimal impact on output
Resource availability	Are spare parts and personnel available?	Coordination of the decision with logistics and staffing
Production plans	How does maintenance align with the schedule?	Optimization in the context of business priorities

The result of applying RxM is that the operator receives not merely a warning about a potential failure, but a specific prescriptive recommendation.

Figure 2.21 presents the architecture of prescriptive maintenance (RxM) using a digital twin. The input data for the digital twin are diagnostic and technological parameters coming from sensors and production processes.

Based on these inputs, three key functions are implemented: prediction of residual useful life (RUL), scenario-based "what-if" simulations to analyze the consequences of different actions, and optimization of decisions using artificial intelligence methods and reinforcement learning. The output consists of concrete prescriptive actions, for example: "replace the bearing in 48 hours," which ensures the transition from prediction to the execution of management decisions.

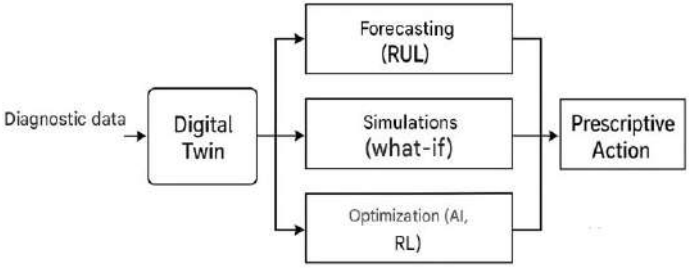

Figure 2.21. RxM architecture with a DT

The input data consist of diagnostic and technological parameters coming from sensors and production processes. Based on these inputs, three key functions are implemented: prediction of RUL, scenario-based "what-if" simulations, and optimization of decisions using AI methods. The system output consists of concrete prescriptive actions, ensuring the transition from prediction to execution of management decisions.

To understand the place of RxM in the overall evolution of maintenance strategies, Table 2.11 presents a comparison of different levels.

The real operation of complex technical systems requires continuous refinement and adaptation of decisions as new data arrives. This approach is possible only with a closed-loop management system that integrates all stages of the lifecycle.

Table 2.11

Evolution of maintenance strategies

Level	Characteristic	Question answered	Example
Reactive (Run-to-Failure)	Repair after failure	“What happened?”	Replacement of a burnt-out engine
Preventive (Planned)		“When to perform maintenance?”	Maintenance every 1000 operating hours
Predictive (PdM)	Failure prediction	“When will the failure occur?”	Prediction of residual life of a bearing
Prescriptive (RxM)	Prescriptive actions	“What to do and why?”	Optimal recommendation for component replacement

The concept of Closed-Loop Lifecycle Management becomes a key paradigm in modern management of complex technical system states, providing continuous integration of diagnostics, prediction, and prescriptive decisions into a unified adaptive process.

The cycle includes sequential stages:

1. Data: collection of diagnostic and operational data;
2. Diagnostics: detection and classification of anomalies;
3. Prediction: modeling of future states using a digital twin;
4. Control Action: implementation of decisions based on prescriptive recommendations;
5. New Data: monitoring the results of actions;
6. Validation and Learning: comparison of predictions with actual results and retraining of models to improve accuracy [94].

The significance of this cycle lies in transforming the digital twin from a static model into a self-learning system that continuously improves its predictions and recommendations.

For the implementation of a closed-loop management cycle in a digital twin, a microservice architecture of the platform is used, which allows the separation of data processing, modeling, optimization, and visualization into specialized services interacting through API gateways. Such an architecture provides scalability, fault tolerance, and flexibility for integration with industrial systems. Figure 2.22 presents the diagram of the microservice architecture of the digital twin platform.

Table 2.12

Stages of the closed-loop management cycle based on a DT

Stage	Content	Role of the DT
Data	Collection of diagnostic and operational data	Integration of data streams
Diagnostics	Detection and classification of anomalies	Identification of deviations from normal
Prediction	Forecasting future state	PdM models for failure risk assessment
Control action	Decision making and implementation	Prescriptive recommendations (RxM)
New data	Monitoring the result of action	Recording actual outcomes
Validation and learning	Comparison of prediction and fact	Self-learning and accuracy improvement

Thus, subsection 2.5 demonstrates the evolution of how diagnostic data, passing through the digital twin, is transformed from information into specific prescriptive actions, ensuring the closure of the control loop and the transition from prediction to adaptive management of the state of complex technical systems.

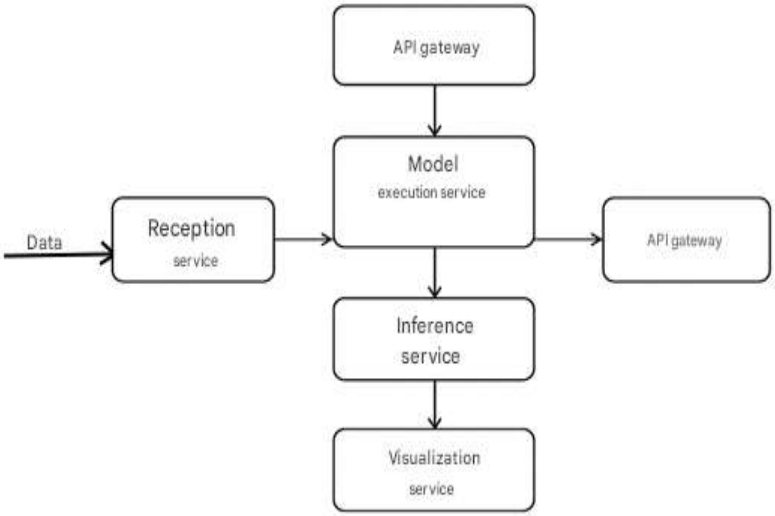

Figure 2.22. Microservice architecture of the DT platform

2.6 Diagnostic models and digital methods for state analysis of complex technical systems

Analysis of CTSs is impossible without the application of a systems approach, which implies consideration of numerous interrelated aspects. The studied object is regarded as a multi-component system, encompassing a hierarchy of models of the object, its subsystems, and elements. For such analysis, it is necessary to use a system of criteria that allows for the assessment of performance, reliability, safety, and other operational characteristics. An important element is the information support system for research, as well as the procedures for developing and making decisions at all stages of the CTS lifecycle. The breadth of applications of systems analysis largely determines the diversity of methods employed for modeling and diagnostics [13, 14, 15].

Key methods include mathematical modeling (optimization, probabilistic, simulation, fuzzy models), elements of information theory, and decision-making techniques. These methods are used to solve tasks such as failure risk assessment, analysis of structural properties of stability (reliability, robustness, survivability, safety), as well as operational efficiency and risk. Markov chains, queuing theory, statistical modeling, and the apparatus of semi-Markov processes are most widely applied. An approach using a probabilistic model of real phenomena, accounting for the likelihood of undesirable events and potential damage, has become the basis for quantitative risk assessment.

In modeling, it is necessary to account for the uncertainty of input data, associated with a limited number of parameters, finite measurement accuracy, and the complexity of external factors. Uncertainty can be described in various forms: deterministic, stochastic, interval-based, or fuzzy. The latter approach is especially important in expert evaluation of parameters expressed in natural language, where “fuzzy” terms are used. A diagnostic model (DM) is defined as a formalized description of the diagnostic object, adequately reflecting the structure, the process of changing technical states, and providing the ability to identify the object’s state with a given depth. The DM serves as the basis for constructing diagnostic algorithms and is, in fact, a mathematical model of the object, taking into account its design features. It can be specified explicitly (a set of formal descriptions of all possible object states) or implicitly (a mathematical description of physical faults and rules for constructing faulty-state models).

Traditionally, the literature emphasizes structural-analytical models, focused on checking operability at the production stage. However, in operation, functional models that describe the dynamic behavior of the control object are most relevant. For this purpose, logical and temporal

parameters, as well as structural indicators, are used [95]. Logical parameters define rules for converting input signals into outputs and internal states; temporal parameters describe dynamics (delays, transient processes, frequency of changes); the structure specifies the nature of element interactions. For describing diagnostic objects, analytical, structural, structural-analytical, and behavioral (automata-based) models are employed. An analytical model is built on a system of equations; a structural model is represented as a graphical depiction of topology; a structural-analytical model combines these approaches. An automata-based model is defined by transition and output tables, describing operational logic and faults.

The choice of a fault model is a key stage. The main types of models are:

- logical (discrete and multi-level “operational/faulty” scales);
- logical-parameter (for checking correct functioning);
- logical-dynamic (accounting for temporal characteristics of malfunctions).

Logical-dynamic models allow describing the behavior of faulty objects over time, thus expanding diagnostic capabilities. Therefore, the DM is characterized by component and interconnection parameters, as well as an adequately selected fault model. The DM component hierarchy includes modules (elementary or virtual structures), blocks, nodes, subnetworks, and networks. At each level, it is possible to determine the technical status of the object: operational, failure, or condition (non-failure defect). Transitions between states are conveniently described using graphs [95], which enable the development of decision-making algorithms in management system software.

Fault function tables (FFT), describing the behavior of both the operational object and the object with defects, have gained wide adoption [96]. They allow the construction of diagnostic tests—both verifying and fault-identifying.

Structural-logical modeling of reliability, survivability, and risk of CTS occupies an important place. These methods provide automation of model construction and calculation of stability indicators, which is especially important for complex systems.

Current trends are associated with the application of simulation modeling (SM), which allows experimentation with probabilistic models and the study of various operational scenarios. A typical SM scheme is shown in Fig. 2.23. This approach corresponds to the methodology of formalized safety assessment (FSA), based on comparing alternatives and selecting the optimal solution to reduce risk.

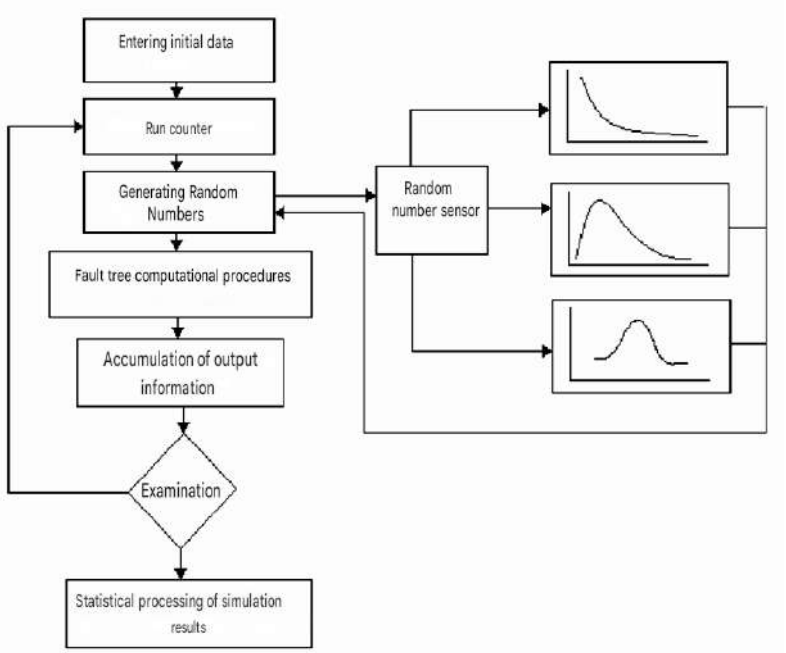

Figure 2.23. Block diagram of simulation modeling

The wide adoption of simulation modeling is supported by advanced software environments: Arena, AutoMod, AnyLogic, Extend, GPSS World, and others [97, 98]. However, the key difficulty remains—the collection and formalization of source information, which requires significant effort.

For failure risk analysis, the use of CSM is considered promising, as shown in Fig. 2.24. CSM is applied in the development of forecasting and decision-support systems under conditions of uncertainty [68, 99].

The main advantage of CSM is its ability to operate with partially reliable and incomplete data, which reflects the real situation in CTS diagnostics. Additionally, fuzzy logic methods are applied, enabling the construction of fuzzy-probabilistic models and the identification of pre-failure states [15]. Fuzzy cognitive maps make it possible to visualize the propagation paths of threats, although they have limitations in defining linguistic variables.

The conducted analysis confirms the necessity of developing cognitive-simulation and fuzzy modeling approaches for failure risk assessment of CTS. These methods provide integration of structural and functional

analysis while accounting for the interaction of elements and inter-element connections.

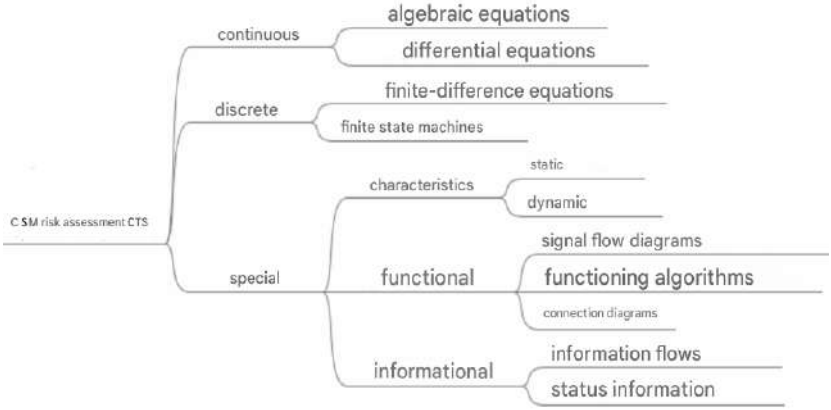

Figure 2.24. Cognitive-simulation models for failure risk assessment of CTSs

The next aspect is related to the integration of diagnostic models into the DT concept. In this case, all models are considered as complementary components rather than isolated solutions. Three main classes can be distinguished:

- Physics-Based - grounded in differential equations and physical laws; applied for accident mode analysis [100];
- Data-Driven - based on statistical and machine learning methods; effective in predicting remaining useful life (RUL) [101];
- Hybrid - combine physical laws and ML, ensuring a balance between accuracy and interpretability [102].

Their comparative analysis is presented in Table 2.13.

Figure 2.25 illustrates the classification of digital twin models, reflecting the interconnection between physics-based, data-driven, and hybrid approaches. The arrows indicate the flow of information: diagnostic and operational data are fed into the models, generating forecasts and recommendations, while control actions are fed back into the system.

This structure demonstrates the principle of complementarity of models: physics-based models provide validity and interpretability, data-driven models ensure adaptability and the ability to handle big data, and hybrid models achieve a balance between accuracy and physical consistency.

Table 2.13

Comparative analysis of digital twin model classes

Model class	Data volume	Interpretability	Extrapolation beyond training data	Computational complexity	Typical application areas
Physics-based	Low	High	Yes	High (solving equations)	Accident modes, critical assets
Data-driven	High	Low–Medium	No	Medium–High	RUL prediction, anomaly detection
Hybrid	Medium	Medium–High	Partial	Medium	Complex systems requiring accuracy and generalization

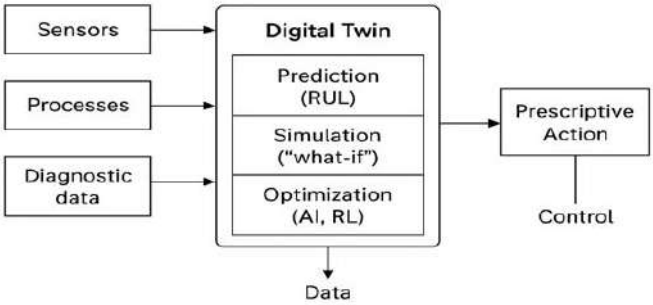

Figure 2.25. Classification of DT models

Models as a Service (MaaS - Model as a Service).

This is an architectural approach to deploying and utilizing models within industrial IoT platforms.

Essence: each model (physics-based, data-driven, hybrid) is encapsulated into a standardized microservice that provides its functionality through a unified API (most commonly REST or gRPC) [103].

Advantages: isolation and scalability: Models developed in Python (TensorFlow/PyTorch), Julia (for complex physics), or C++ can operate independently and scale under load; interoperability: any system (SCADA, ERP, dashboard) can easily consume model results simply by calling its API, without requiring knowledge of the internal implementation; lifecycle management: updating, versioning, and A/B testing of models are simplified.

The key advantages and potential challenges of the architectural approach "Model as a Service" are presented in Table 2.14.

Table 2.14

Lifecycle features of models in the MaaS paradigm

Stage	Traditional approach	MaaS	Effect
Development	Monolith	Service with API	Accelerated implementation
Deployment	Manual integration	CI/CD	Reduced timeframes
Operation	Rigid coupling	Scalability	Real-time operation
Maintenance	System-dependent	Independent updates	Risk minimization

The architecture of model deployment in the MaaS paradigm is presented in Figure 2.26. Each model (physical, data-driven, hybrid) is encapsulated in a microservice that provides its functionality through an API (REST or gRPC). These services are integrated via API gateways, ensuring access for external consumers (SCADA, ERP, visualization systems). Such a structure guarantees scalability, interoperability, and convenient lifecycle management of models.

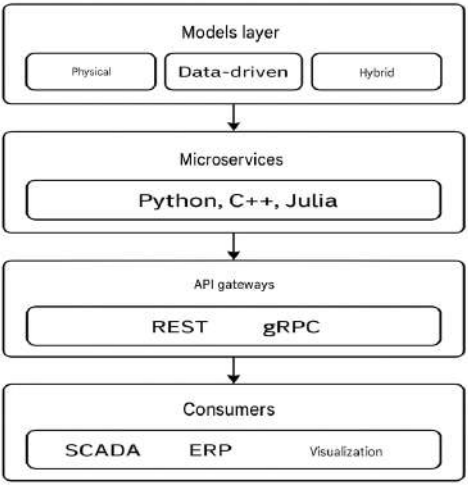

Figure 2.26. Architecture of model deployment as services (MaaS)

Thus, subsection 2.6 demonstrates the evolution of diagnostic methods from classical probabilistic models to cognitive, fuzzy, and hybrid approaches, as well as the integration of diagnostic models into the digital twin using MaaS. This ensures the formation of comprehensive solutions for analysis, forecasting, and management of the condition of complex technical systems.

CHAPTER 3. INFORMATION SUPPORT FOR MONITORING, DIAGNOSTICS, AND PROGNOSTICS OF THE TECHNICAL CONDITION OF COMPLEX TECHNICAL SYSTEMS

3.1 Processing of large-scale diagnostic data on the technical condition of complex technical systems

At present, to ensure the processes of Big Data processing and analysis, solutions based on the distributed computing model MapReduce are actively used, which makes it possible to provide parallel data processing and increase the processing speed [102, 103, 104, 105]. Due to the use of a distributed file system (HDFS), it becomes possible to flexibly organize the processes of data processing and reduction on the control server. The advantages of this approach include the automation of the distribution of computing nodes into clusters within a computer network, which makes it possible to utilize the computing capabilities of an unlimited number of hosts, ensure the implementation of data redundancy algorithms to guarantee reliable storage, and support software implementations for most modern high-level programming languages [106, 107, 108]. The disadvantages of the model include the laborious organization of real-time streaming data processing, the complexity of deploying the system under conditions of an unstable data transmission channel with low bandwidth, and the resource-intensive procedure of data visualization by iterations of their processing. As the length of SQL commands for aggregated search, selection, insertion, and data storage operations in relational DBMS increases, the complexity of forming transactional queries to databases (DB) also increases. Depending on the hardware resources used, this can significantly reduce the speed and efficiency of data processing in real time. Therefore, as Big Data repositories, it is advisable to use non-relational databases (NoSQL) such as MongoDB, Redis, HBase, Firebase, and others. The use of NoSQL in the context of Big Data processing is due to the following factors: a high level of flexibility in ensuring the required level of data scalability; no restrictions on the stored data types; support for the data representation model; and the presence of a document-oriented approach.

Large volumes of technical information are characterized by static and stable behavior since, for a significant part of the operation time of CTS under normal conditions, their parameters do not change sharply. This specificity necessitates the constant reorganization of data to exclude low-informative arrays and the application of statistical analytical procedures for summarizing and aggregating information, which requires additional computing resources and operations [82]. To solve such tasks, it is advisable to develop an analytical module that performs parsing and classification of data in order to compress them in the form of a separate node when

implementing a microservice architecture. The task in processing Big Data from sensors of system element parameters, data collection and transmission modules, and CTS control modules is the effective segmentation and clustering of information into separate sets [109]. The advantage of this approach lies in the ability to conduct more accurate data analysis through the parallel use of different types of models for each of the identified sets, which allows comparison and adequacy analysis of each of them, as well as contributes to a reduction in the number of computational operations performed when processing the data array.

Research on the possibilities of structuring and concise representation of incoming data to ensure their further processing and analysis is of current relevance. The obtained models can be integrated into the designed decision support systems based on the use of AI methods for extracting new useful knowledge from collected data and identifying hidden patterns (Data Mining) [110]. Such systems will reduce the time required for analytical operations, allowing for the assessment of various operating scenarios of CTS, their subsystems, and elements, the construction of analytical models, and the performance of prognostics of the technical condition of systems.

The efficiency of organizing the Data Mining process for Big Data significantly depends on the type of data representation, their consistency, quality of cleaning, integrity, orderliness, and granularity. In modern AI, there are a significant number of approaches that allow designing learning systems to automate the processes of searching, accumulating, and structuring data and knowledge [111]. The most promising among them are artificial immune and neural networks, deep machine learning, and hybrid adaptive learning algorithms, which are based on the principles and models of the functioning of the human brain. Such methods can significantly accelerate the processes of Big Data analysis; however, their application requires preliminary data structuring and the identification of significant features for forming the input and output data of the designed models [112].

Within the framework of the task being solved, it is possible to apply the Decision Tree (DCT) method to describe the key ontological meta-information about heterogeneous sets of large-scale data [113]. The visualization of DCT is implemented as the display of a connected directed acyclic graph [114, 115, 116]. The branches of the constructed DCT graph store attribute values representing the functional parameters of CTS elements on which the target function depends, while the leaves of the DCT display its resulting value. The conceptual scheme of the composition of the proposed system for structuring and processing Big Data is shown in Fig. 3.1.

Under the control of the data collection system, the collection and transformation of the technical parameters of the CTS elements from the sensors are carried out. The obtained data are aggregated, structured

according to their belonging to subsystems, and stored in files in the *.json format, after which they are sent in separate streams to the cloud IAAS infrastructure using a virtual working environment and containerization by means of Docker.

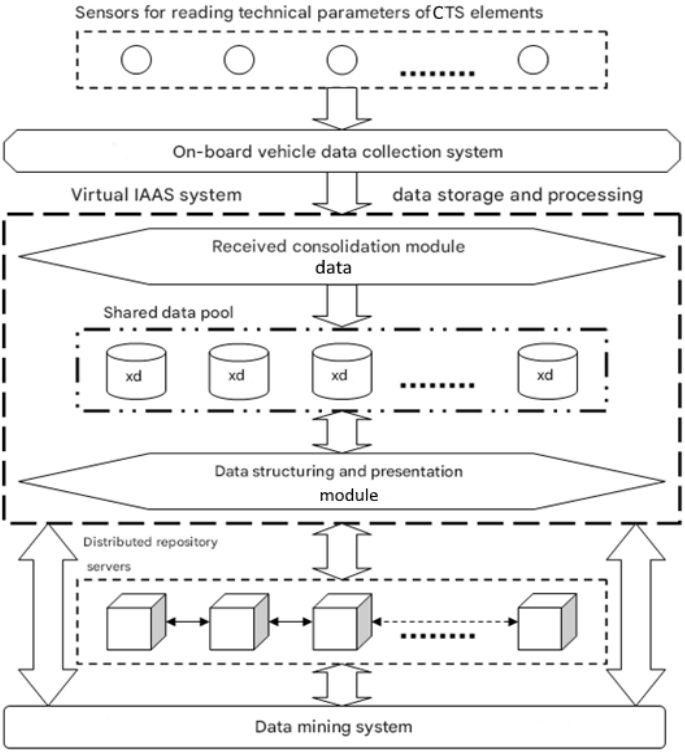

Figure 3.1. Conceptual scheme of the composition of the Big Data structuring and processing system

The data integrated within the IAAS system are processed by the consolidation module according to specified criteria in order to further harmonize and evenly distribute them within individual data storages (DS) in the form of nodes (Nodes) of the data space (pool).

The Data Representation and Structuring Module (DRSM) makes a request to extract the required data of a given volume into the pool, in response to which a dataset from the corresponding DS is returned. As a result of the DRSM operation, the obtained DCT models in the form of an

ordered metadata scheme in json and xml formats are sent for further storage in distributed repositories functioning under the control of the HDFS file system.

Further intelligent data analysis based on the stored metainformation becomes possible through the use of analytical software (Knime, SPSS Statistics, QIWare, or others) or a proprietary information system.

If it becomes necessary to perform additional data processing or analysis operations from the system, direct access to the IAAS system should be provided through the integration of the corresponding API or RDP support with the terminal access service.

The proposed approach to building the DCT model is based on the CART algorithm. The DCT model includes:

- a root node, which is incident to all outgoing edges of the graph;
- internal nodes, which are incident to a single incoming edge of the graph and to several outgoing ones;
- terminal nodes, which are incident to an incoming edge.

Logically, the construction of the DCT model is formed by performing the following stages:

1. Selection of a subset of the training sample of a specified size for building the tree structure;
2. At each iteration of forming a new split in the tree, a numerical evaluation of the frequency of the features is carried out;
3. Selection of the feature with the maximum frequency of occurrence and performing the operation of dividing the tree hierarchy into a new level.

This process is repeated until the entire training sample is exhausted or until the initially specified boundary conditions are reached, in particular, the maximum allowable tree depth.

To reduce the dimensionality of the stored data, it is necessary to evaluate the technical condition of each CTS element based on a set of several heterogeneous parameters, i.e., to solve a classification problem.

As an example for research, a ship power plant based on Wartsila 16V50DF was selected [91].

It is planned to perform two sets of operations (representation and structuring), containing stages (Fig. 3.2).

The main parameters that regulate the logic of the DRSM DCT model construction process are:

- CR, tree splitting criterion - heterogeneity index;
- DP, maximum tree depth level;
- CN, confidence level value of the model construction for assessing the pessimistic error of branch pruning;
- LT, minimum tree leaf value;
- LS, minimum sample size in the training subset;

- PP, number of alternative tree nodes for early stopping during DCT construction.

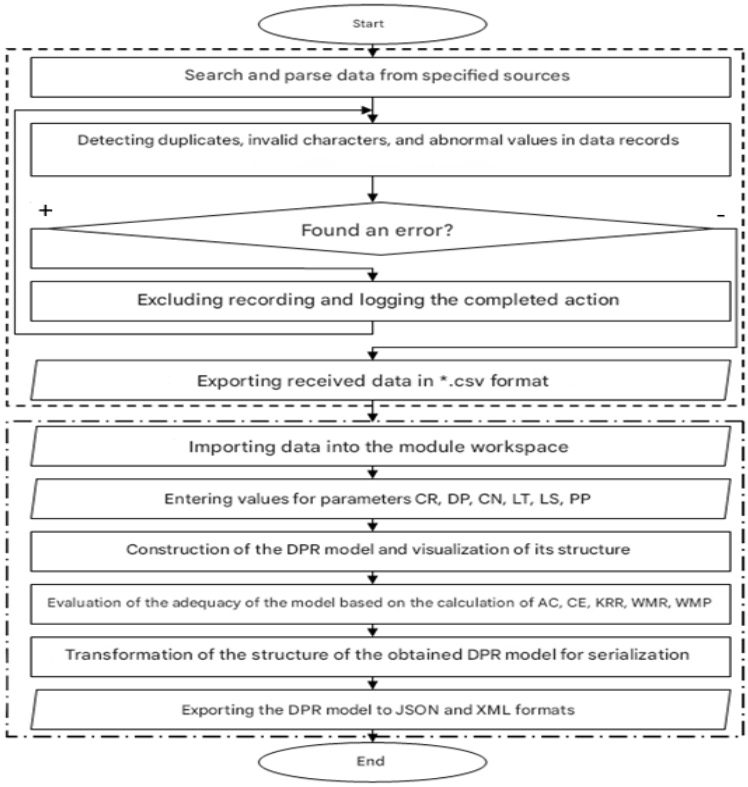

Figure 3.2. General algorithm for building the DCT model for Big Data description

As functional parameters (attributes) for building the DCT model, the following indicators are used for all CTS elements:

- structural risk of system element failure (R_{str}) - reflects the level of its vulnerability in the CTS topology, takes a real data type in the range [0,1];
- functional risk of system element failure (R_{func}) - reflects the level of its vulnerability in the dynamics of CTS functioning, takes a real data type in the range [0,1];

- operating mode of the system element (OM) - characterizes the intensity and load of its use, takes a string data type in the form of two possible options “Regular” and “Irregular”;

- operating duration of the system element (OD) - determines the period of its functional use within the system, takes an integer data type in the range [1,20];

- maintainability degree of the system element (MD) - formalizes the level of susceptibility of the element to functional modifications in the event of a failure for the prompt provision of its further functioning, takes a string data type in the form of three possible options “Low”, “Middle”, and “High”;

- degree of physical wear of the system element (W) - reflects the level of its damage affecting its target functioning, takes an integer data type in the form of a percentage from 0 to 100%;

- mean time between failures of the system element (MTBF) - characterizes the average period of time between the occurrences of failures in its operation, takes an integer data type in the range from 0 to 10^6;

- number of repairs performed on the element (RN) - describes the history of technical manipulations carried out to restore its operability, takes an integer data type in the range [1,10];

- average market cost of the element (EC) - characterizes the criticality of its failure from a financial point of view and the potential cost of replacement, takes an integer data type in the range [10^3,10^5];

- performance of the element (PR) - takes a string data type in the form of three possible options “Low”, “Middle”, and “High.”

The integral indicator for assessing a system element (the target variable for classification) is its technical condition (TC), i.e., the final value of the DCT leaf can be one of two classes - “Acceptable” and “Unacceptable.”

The resulting data functional for description based on DCT has the form

$$TS<Acceptable,Unacceptable>=\{R_{str},R_{func},OM,OD,MD,W,MTBF,RN,EC,PR\}$$

The values R_{str} and R_{func} are formed based on the approach of constructing a CSM for assessing the risk of CTS failures using damaging modeling impulses and normalizing effects. The operability and maintainability degree (MD) are evaluated by expert methods using the method of sequential comparisons, taking into account Spearman's rank correlation coefficient. The average market cost of the element (*EC*) is determined based on the analysis and comparison of prices from five different manufacturers by averaging the final values. The remaining statistical data for 20,000 elements are taken from the OREDA database [92].

To study the possibility of improving the efficiency of solving the classification problem when constructing the DCT model, it is proposed to use the following heterogeneity indices:

1. Information entropy (*IG*). The entropy of all model attributes (*C*) is calculated from the data sample (*X*), and the attribute with the smallest entropy value is selected to form further branches.

$$I\hat{G}(C) = -\sum p(X)\log p(X). \tag{3.1}$$

where *p(X)* is the proportion of examples in the given class of the DCT model, and *C* is the number of model attributes

2. Information Gain Ratio (*GR*). A modified version of *IG*, extending information entropy for each model attribute to ensure breadth and uniformity in data processing and to reduce bias toward multi-valued attributes.

GR is defined through the Split Information (SP) indicator, which represents the potential information generated by splitting the training data set *X* into *v* partitions corresponding to the outcomes of attribute *C*.

$$SPC(X) = -\sum\left(\left|X_j\right| \Big/ X\right)\log 2\left(X_j \Big/ X\right), j = 1..v \tag{3.2}$$

The final value of *GR* is determined as

$$GR(C) = G(C) \Big/ SP(C) \tag{3.3}$$

3. Gini index (*GI*). A measure of inequality between the distributions of the characteristics of a data set. Splitting by the selected model attribute leads to a reduction of the average *GI* in the resulting subsets. If the data set *X* contains data from n classes, then the Gini index is defined as follows:

$$GI(X) = 1 - \sum p_i^2, i = 1..n, \tag{3.4}$$

where *pi* is the probability (relative frequency) of class *i* in *X*

The best split is considered to be the one for which *GI*(*X*) is minimal. If ***N*** is the number of examples in the parent node, *L* and *R* are the numbers of examples in the left and right child nodes, and l_i and r_i are the numbers of

instances of class *i* in the left/right child, then the split quality is evaluated using the following formula:

$$GI(X) = N - \left(\left(1/L\right)\sum l_i^2\right) + \left(\left(1/R\right)\sum r_i^2\right)\right) \to \min, i = 1..n \tag{3.5}$$

4. Standard accuracy (*AC*). At each step, such an attribute is selected for splitting that maximizes the overall accuracy of the model classification. For the purpose of assessing the quality of model construction, it is advisable to employ and implement the following numerical metrics:

– *ACC* – the relative number of correctly classified examples in the data sample, i.e., the percentage of correct classifications:

$$ACC = (t_p + t_n) / (t_p + f_p + f_n + t_n), \tag{3.6}$$

where t_p (true positive) – a true positive assessment by the DCT model of the technical condition of the CTS, f_p (false positive) – a false positive assessment by the DCT model of the technical condition of the CTS, f_n (false negative) – a false negative assessment by the DCT model of the technical condition of the CTS, t_n (true negative) – a true negative assessment by the DCT model of the technical condition of the CTS

CE – the relative number of incorrectly classified examples in the data sample, i.e., the percentage of incorrect classifications. The classification error *CE* of the DCT model depends on the number of incorrectly classified samples and is evaluated by the formula:

$$CE = \left(F/n\right) \cdot 100, \tag{3.7}$$

where *F* is the number of incorrectly selected samples, and *n* is the total number of samples

KPP – kappa statistic, which makes it possible to account for random correct classification:

$$KPP = (P_0 - Pe) / ((\max P_o) - P_e)', \tag{3.8}$$

where P_0 is the initial characteristic of class proximity, and P_e is a random variable in the range from 0 to P_0

WMR – the value of the weighted mean recall of classification, i.e., the proportion of correctly classified records relative to the total number of all relevant records in the sample.

WMP – the value of the weighted mean precision of classification, i.e., the proportion of correctly classified records relative to the total processed sample.

For the purpose of testing the concept of structural representation of the ontological schema of meta-information about data, a simulation of the operation process of the MPSD was carried out by constructing the DCT according to the designated functional TS of the CTS using the cross-platform software RapidMiner Studio. After data collection and preprocessing based on their automatic (manual) statistical analysis using distribution boxplots, their consolidation was performed using the standard tools of the system. The data were imported into the system's workspace using the functional block Retrieve (Fig. 3.3).

The data imported from the Retrieve block are sent to the set role block, which is necessary to assign logical roles to the required attributes of the analyzed data sample. Using this block, the target output variable is selected for the formation of the DCT structure, which in our case is the technical condition of the CTS element.

The configured dataset is fed into the Split Data block, which splits the sample into subsets according to the selected ratio. The block is configured as follows: for the training subset, the sample proportion is in the range [50%; 70%], and for the test subset, the range is [50%; 30%]. The split is performed as follows: if the label is nominal, stratified sampling is carried out (random subsets are allocated so that each subset contains approximately equal proportions of the two values of the target variable); otherwise, a simple random shuffle of the data is performed.

The separated data are then sent to the Decision Tree model configuration block and the apply model block, which is required for further evaluation of the model's performance.

The decision tree block implements operators for setting all previously designated parameters and constraints during the construction of the DCT simulation model (Fig. 3.4), allowing the selection and configuration of various combinations of structuring the processed data by CTS elements in order to identify the most informative representation of the data in graphical form.

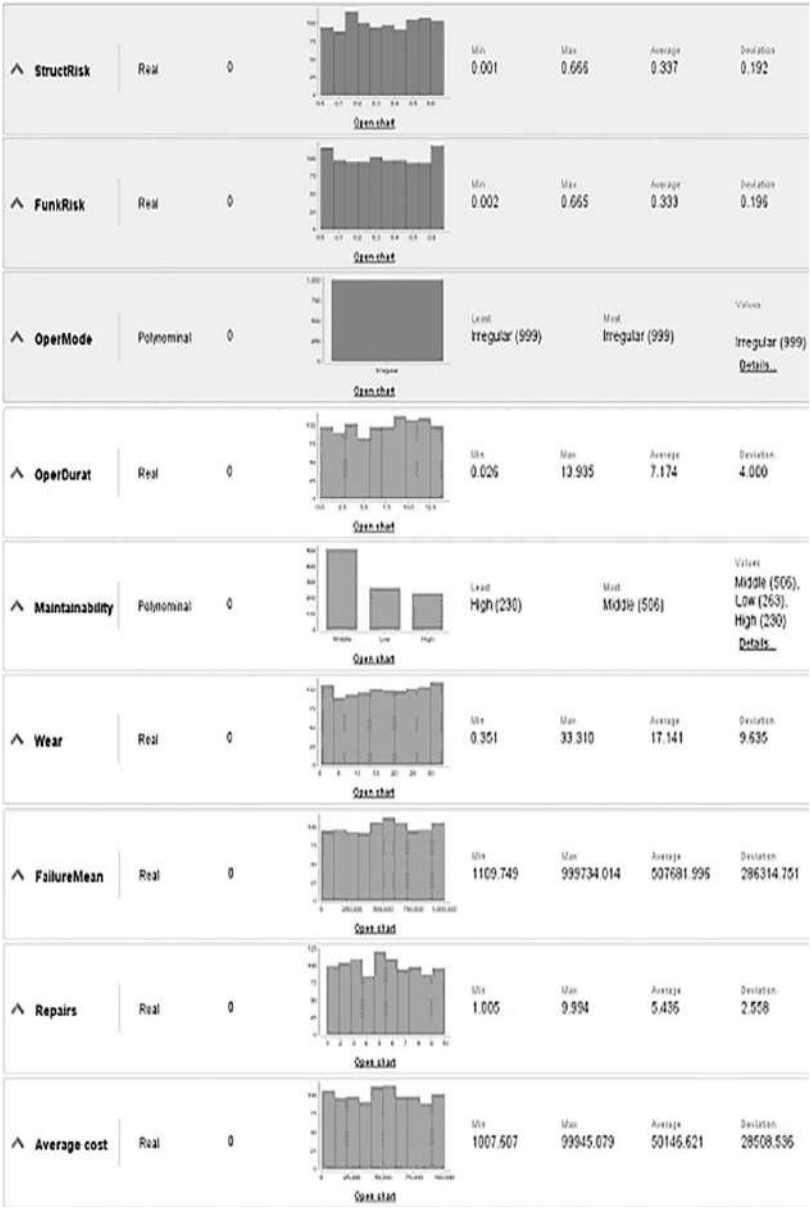

Figure 3.3. Fragment of the statistical processing results of the presented data for DCT model formation

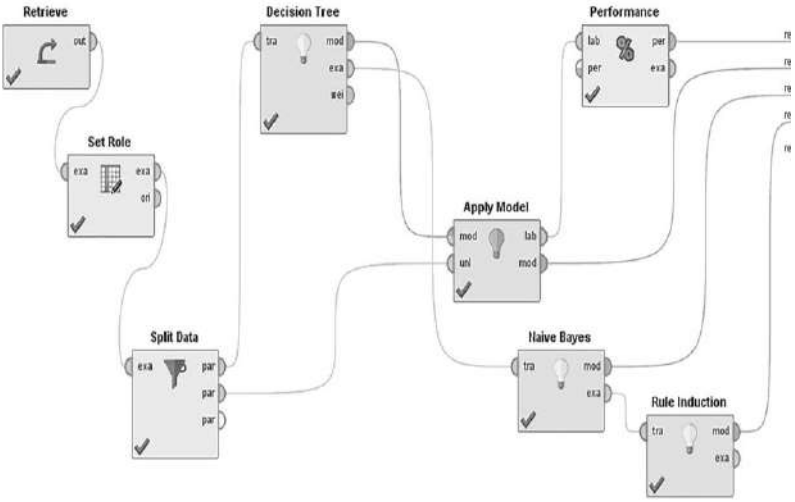

Figure 3.4. Block composition of the DCT simulation model

The determination of numerical values of metrics for assessing the quality of the constructed DCT model is carried out in the performance block. The evaluation results are output from the Performance block for model visualization in the results analysis mode of RapidMiner Studio.

For the informational enhancement of the structured DCT model, it is necessary to define and formalize logical production rules capable of simplifying the representation and interpretation of data in the naive bayes and rule induction blocks. Data from the Decision Tree block are sent to Naive Bayes. After the processing is completed, the data are transferred to the Rule Induction block. The obtained results are then visualized.

The structure of the DCT model based on the use of the GR index is shown in Fig. 3.5.

The root of the tree is the attribute of structural failure risk (R_{str}). If its value is below 0.49, the model classifies the system's operability level as low. If R_{str} exceeds 0.49, classification proceeds according to the remaining attributes in descending order of their priority. The next attribute of the DCT model is the regulated mean time between failures (MTBF); exceeding the value of 180,000 leads to splitting the data according to the operating duration of the element (value less than or equal to 3.5 – high operability level), degree of physical wear (value less than or equal to 7.5 – high operability level), and average market cost (value less than or equal to 17,500 – high operability level).

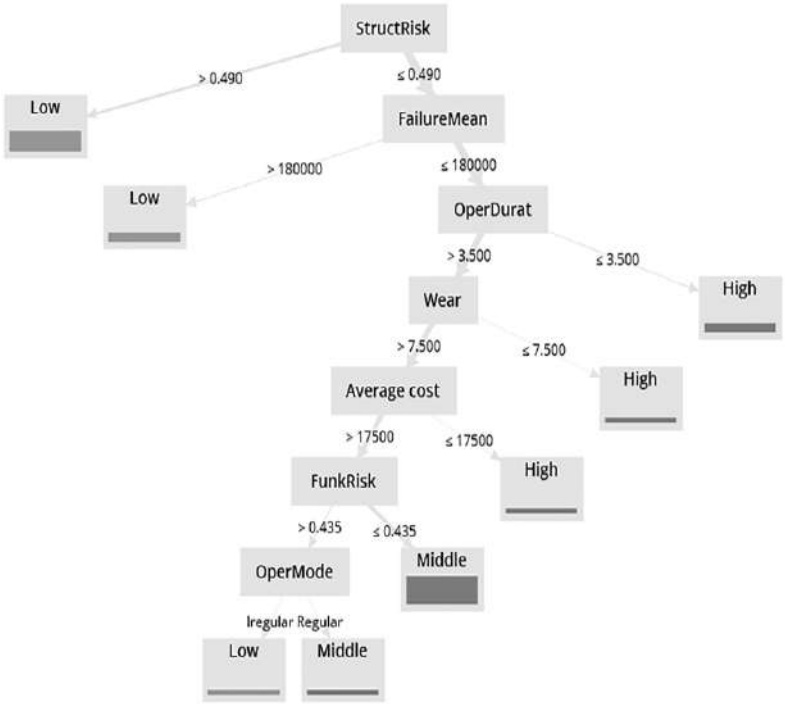

Figure 3.5. Structure of the DCT model based on the use of the GR index

With further splitting of the tree by the functional failure risk attribute, the operability class is either medium (functional failure risk value less than or equal to 0.435) or low (if the operating mode is irregular).

For the DCT model based on the use of the *GI* index (Fig. 3.6), a characteristic feature is the different order of attribute splitting, in particular, the higher priority of the degree of physical wear attribute. If the system parameter takes a value greater than or equal to 16, further splitting is performed according to the degree of maintainability and the structural risk at a value of 0.26. If the value of the degree of physical wear is above 16, the tree is split according to the functional failure risk (medium operability level corresponds to R_{func} less than or equal to 0.32) and the structural failure risk (low operability level corresponds to R_{str} greater than 0.305).

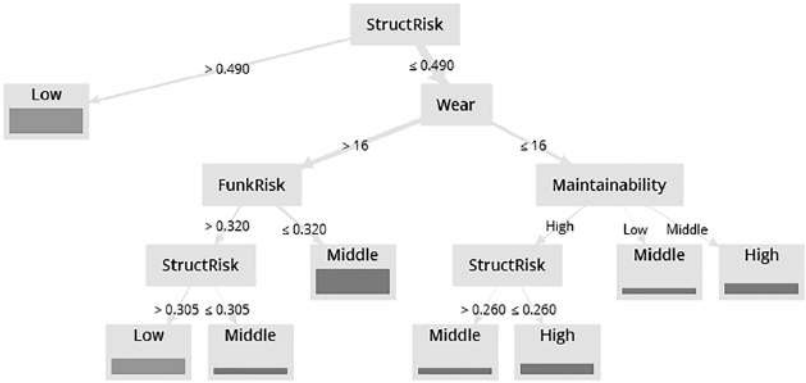

Figure 3.6. Structure of the DCT model based on the use of the IG index

According to the model, a low operability level of the CTS corresponds to values of the most prioritized parameter – structural failure risk – in the range (0.305; 1], medium [0.305; 0.26), and high [0.25; 1]. For the DCT model based on the *GI* index (Fig. 3.7), the attributes of average market cost and degree of physical wear are equal in priority after the structural failure risk.The structure of the DCT model based on the use of standard accuracy (*AC*) is shown in Fig. 3.8. The more significant attributes after structural failure risk are functional failure risk and degree of physical wear. In DCT models, the results of solving the data classification problem according to the CTS operability level are structured and displayed with different levels of detail and flexibility in interpretation. Therefore, it is necessary to perform numerical evaluation of the used metrics for a comprehensive analysis of the results.The summary results of the metric evaluation for the obtained DCT models based on the *GR*, *IG*, *GI*, and *AC* indices are presented in Table 3.1.

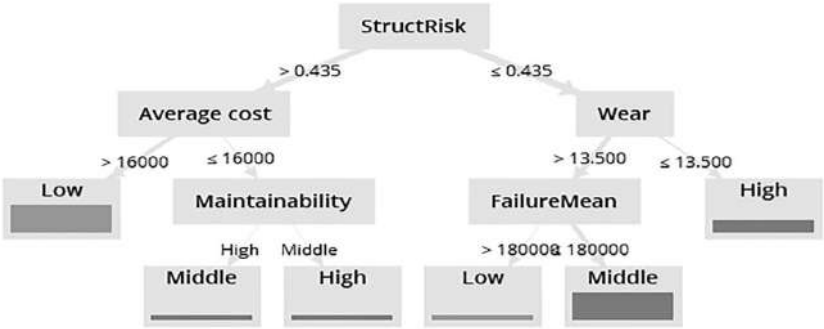

Figure 3.7. Structure of the DCT model based on the use of the *GI* index

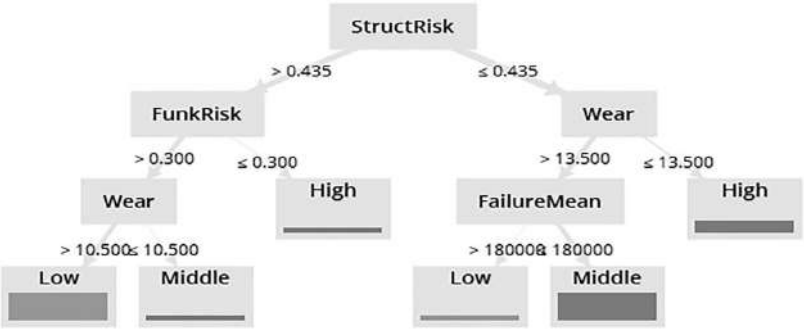

Figure 3.8. Structure of the DCT model based on the use of standard accuracy

Table 3.1

Summary results of metric evaluation for DCT models

		Used indices			
		GR	***IG***	***GI***	***AC***
Metric values	***ACC***	98.01%	96.79%	91.24%	93%
	CE	1.99%	3.21%	8.76%	7%
	KPP	0.966	0.95	0.89	0.9
	WMR	98.48%	97.57%	92.91%	91.73%
	WMP	97.22%	95.70%	91.26%	89.95%

Classification performed based on the use of the *GI* and *AC* indices is less accurate due to the low values of the calculated metrics. For the purpose of additional analysis of the reliability level of data classification by DCT models, a confusion matrix was compiled for the used indices (Table 3.2). Analysis of the matrix shows that the precision and recall indicators are highest for models based on the *GR* index. DCT models based on *IG*, *GI*, and *AC* have lower indicators, which confirms their lower accuracy in adequately representing the analyzed dataset. Since the attribute PR is a value obtained by expert evaluation, it is advisable to perform a statistical analysis of the classification results with a modified target function (Table 3.3). The obtained mean values fall within ranges close to the statistically determined values in the originally imported data sample, which provides additional confirmation of the adequacy of the obtained results. A fragment of the logical rules obtained from the DCT model based on the *GR* index for data classification is presented below. The resulting array of logical rules can be used to supplement metadata arrays and for further construction of neuro-

fuzzy models to evaluate the relationships between technical parameters of CTS elements and to conduct a deeper search for and identification of hidden associative rules among them.

StructRisk > 0.490: Low {High=0, Middle=0, Low=5}
StructRisk ≤ 0.490
| FailureMean > 180000: Low {High=0, Middle=0, Low=2}
| FailureMean ≤ 180000
| | OperDurat > 3.500
| | | Wear > 7.500
| | | | Average cost > 17500
| | | | | StructRisk > 0.435
| | | | | | OperMode = Iregular: Low {High=0, Middle=0, Low=1}
| | | | | | OperMode = Regular: Middle {High=0, Middle=1, Low=0}
| | | | | StructRisk ≤ 0.435: Middle {High=0, Middle=7, Low=0}
| | | | Average cost ≤ 17500: High {High=1, Middle=0, Low=0}
| | | Wear ≤ 7.500: High {High=1, Middle=0, Low=0}
| | OperDurat ≤ 3.500: High {High=2, Middle=0, Low=0}

Table 3.2

Confusion matrix of DCT models

GA			
	True allowable TC	**True non-allowable TC**	**Classification accuracy**
Classified as allowable TC	10200	600	94.44%
Classified as non-allowable TC	0	19100	100%
Classification recall	100%	96.95%	
IG			
	True allowable TC	**True non-allowable TC**	**Classification accuracy**
Classified as allowable TC	17000	1600	91.40%
Classified as non-allowable TC	3	31300	95,84%
Classification recall	95,84%	95.14%	
GI			
	True allowable TC	**True non-allowable TC**	**Classification accuracy**
Classified as allowable	13600	1100	92.52%

TC			
Classified as non-allowable TC	0	25200	100%
Classification recall	100%	92.82%	
AC			
	True allowable TC	**True non-allowable TC**	**Classification accuracy**
Classified as allowable TC	6800	600	91.86%
Classified as non-allowable TC	0	12600	100%
Classification recall	100%	95.45%	

Table 3.3

Results of statistical evaluation of classification by attribute model

Attribute	Parameter	Maximum PR value	Mean PR value	Minimum PR value
Rstr	Mean	0.243	0.318	0.523
	Standard deviation	0.179	0.136	0.13
Rfunc	Mean	0.303	0.287	0.527
	Standard deviation	0.076	0.148	0.17
OM	value=Regular	0.958	0.495	0.333
	value=Irregular	0.042	0.505	0.667
OD	Mean	5	9.333	8.667
	Standard deviation	3.464	4.885	6.088
MD	value=High	0.347	0.181	0.022
	value=Middle	0.633	0.489	0.649
	value=Low	0.02	0.33	0.33
W	Mean	12	26.5	33.5
	Standard deviation	3	12.406	16.622
MTBF	Mean	35000	20500	126167
	Standard deviation	35000	10728.5	120330

RN	Mean	2.667	4.333	4.333
	Standard deviation	0.577	2.944	1.966
EC	Mean	37333.3	67166.7	40500
	Standard deviation	19399.3	44024.6	17762.3
TS	value=Acceptable	0.749	0.413	0.186
	value=Unacceptable	0.113	0.873	0.291

The highest classification accuracy values (ACC) obtained by the data model were for GR and IG, 98.01% and 96.79% respectively. For these indices, the values of KPP, WMR, and WMP exceed 0.95, indicating high accuracy and reliability in solving the classification task. From the perspective of convenience in representing metadata about the data and its subsequent interpretation, the DCT model implemented based on GR is more detailed and flexible. The resulting density distribution of the R_{str} attribute for acceptable and unacceptable TC of the CTS based on the averaged data sample is shown in Fig. 3.9. The peak growth of values for the acceptable TS class occurs in the range 0.25–0.33 (density approximately 2.3) and is narrower in shape compared to the range of values for the unacceptable TC class (0.31–0.41, density close to 2). This indicates a predominance of the analyzed datasets with negative outcomes (the CTS TC is unacceptable and requires prompt intervention). The scatter plot showing the correlation between structural failure risk and the technical state of CTS elements is presented in Fig. 3.10.

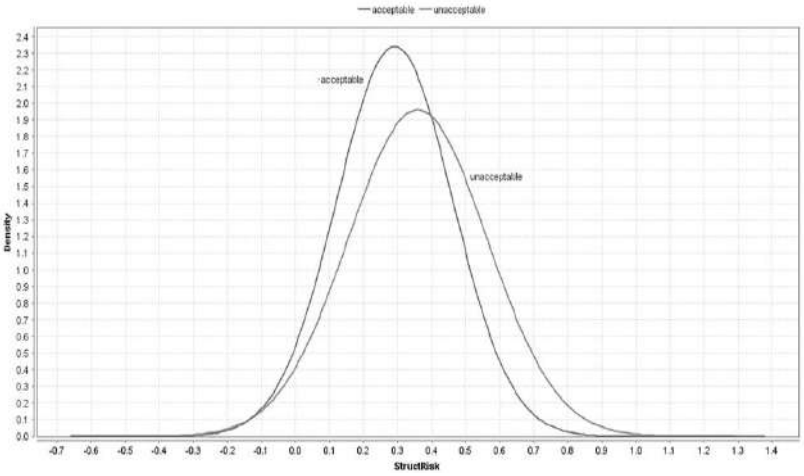

Figure 3.9. Density distribution of the structure risk attribute

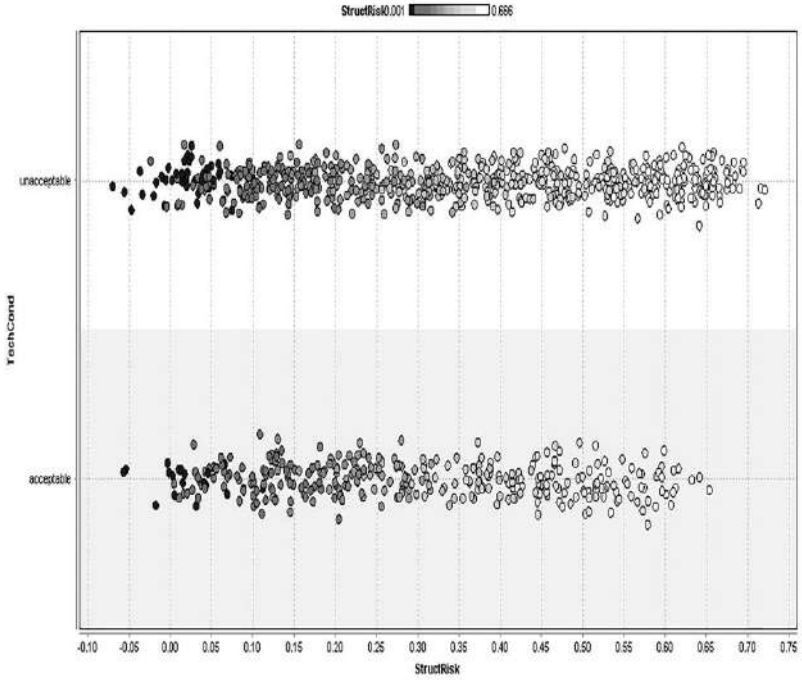

Figure 3.10. Scatter plot showing the correlation between structural risk and the technical state of elements

In the diagram, each record of the analyzed dataset corresponds to a point, the Cartesian coordinates of which represent the classified values of structural risk on the x-axis and CTS TC on the y-axis. Analysis of the constructed scatter plot confirms the previously stated predominance of cases in which TC is unacceptable. Data visualization also allows the identification of small outliers at the boundary (initial and final) ranges, which can be filtered during further analysis.

It is useful to compare the level of correlation between structural and functional failure risks relative to the target variable TC. The scatter plot showing the correlation between functional risk and the technical state of CTS elements is presented in Fig. 3.11. The diagram indicates uniformity in the displayed data and a low degree of outliers caused by errors and inaccuracies during data processing

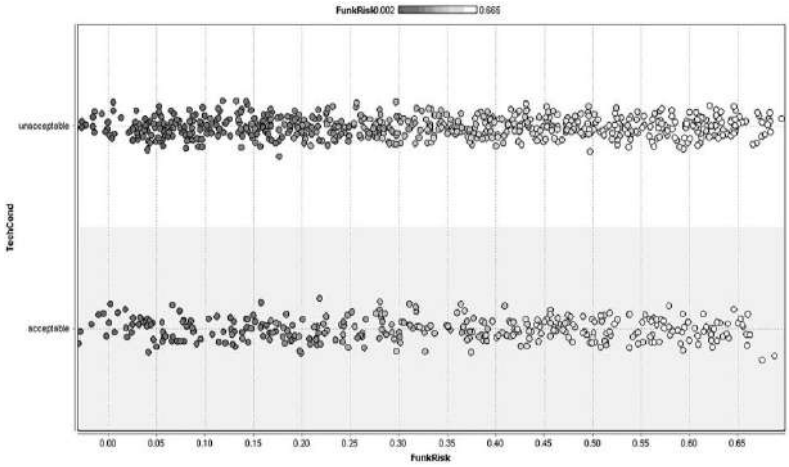

Figure 3.11. Scatter plot showing the correlation between functional risk and the TC of elements

The density distribution of CTS technical state data depending on wear and completed repairs by system elements, for assessing their impact on technical condition in the form of a heat map, is shown in Fig. 3.12. Points indicate values of acceptable and unacceptable technical states of CTS elements.

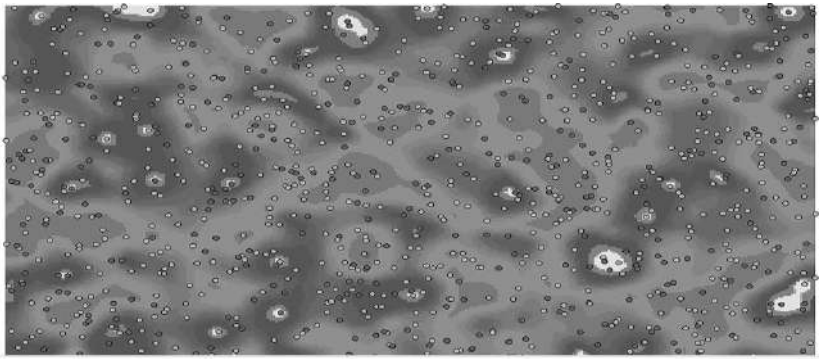

Figure 3.12. Fragment of heat map of the density distribution of wear and completed repairs by system elements

Analysis of the formed areas allows us to assert a more pronounced correlation between low operability and unacceptable technical state values of CTS elements, and a less pronounced correlation when operability is medium or high. This can be used to form weighting coefficients for the predominance of certain parameters characterizing TC over others in the

context of further refinement of the DCT model, aiming to expand the presented metadata for better interpretation and understanding by the analyst. From the analysis of DCT models and graphical diagrams, it follows that a higher level of correlation is observed among structural failure risk parameters, the degree of technical wear, and mean time between failures. The values of these parameters exert the highest influence on the technical state and operability level of CTS elements, which determines the criticality of their operational monitoring and analysis for the functioning of the entire transport system. The resulting DCT structure is subsequently necessary to accelerate the data analysis process over the formed metadata arrays by reducing computational costs for data processing and avoiding the need for operations related to sending and transforming additional samples.

It should be noted that the final size of the constructed data representation models is quite large (about 30 megabytes for 20,000 records), which can later be reduced by applying logical convolution operations according to individual criteria, archiving algorithms, and using gradient boosting through composing from other existing machine learning algorithms. Analysis of the research results of DCT models established that the most suitable approach is the tree construction based on the GR index. The resulting model is 8–10% more effective in terms of results compared to other constructed and investigated models, which determines its suitability for solving Big Data structuring tasks and its convenience for hierarchical visualization of interrelationships between selected data segments. The model is used as a DSS module, performing evaluation and prognostics of the technical state of CTS elements under various operating conditions.

In solving the tasks of Big Data transmission and processing, formalization of tasks based on their software implementation is required. The tasks to be solved include:

- development of an algorithm for intercepting and intermediate analysis of the transmitted CTS information flow to the result storage server;
- calculation of the initial metadata in the package;
- development of an algorithm for distributing packages according to the read metadata;
- development of a data archiving algorithm according to transmission standards via satellite lines;
- construction of a data transmission model based on minimizing the transmitted package size;
- development of an algorithm for performing data analysis necessary to build a prognostic model of CTS state;
- prognostics of CTS state based on the data obtained from SCADA systems.

For the software implementation of Big Data transmission and processing tasks, it is necessary to create a modular software solution, allowing its use as part of the functional module of a SCADA system. The initial stage of implementing a modular software solution is the formulation of software requirements, executed using software development tools. For this, design tools based on CASE systems are used.

The construction of a use-case diagram reflecting the software interaction of CTS subsystems in the modular software solution is shown in Fig. 3.13.

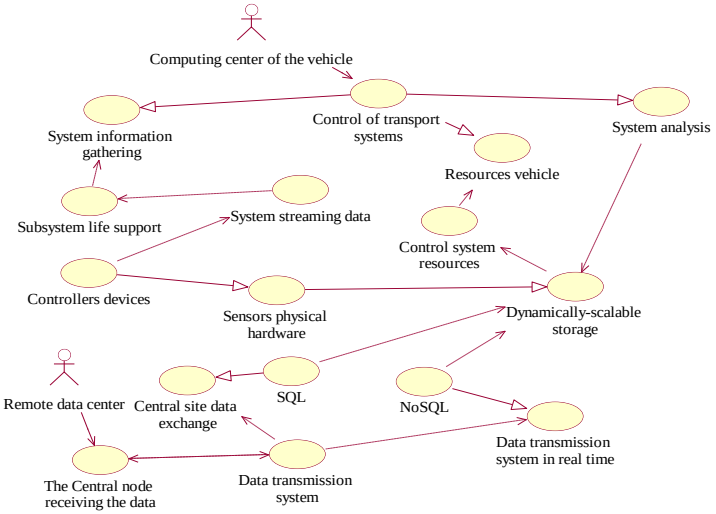

Figure 3.13. Interaction diagram of CTS subsystems in the modular software solution

To determine the sequence of using the modules of the software solution, a generalized algorithm for processing large volumes of information in real time was developed (Fig. 3.14).

For data that is of priority importance, a separate storage built on a non-relational architecture (NoSQL) should be provided. This will allow achieving maximum write speed, as well as minimizing the compressed data archive for further transmission via satellite channels. As a NoSQL storage, it is proposed to use a “bigtable-like” database employing column families as information storage. The use of a “bigtable-like” database is beneficial for handling large volumes of information, making the modular system a sufficiently sophisticated tool for processing big data arrays. As a method for distributing metadata, it is proposed to apply direct classification, allowing sorting by the types of information collection sensors.

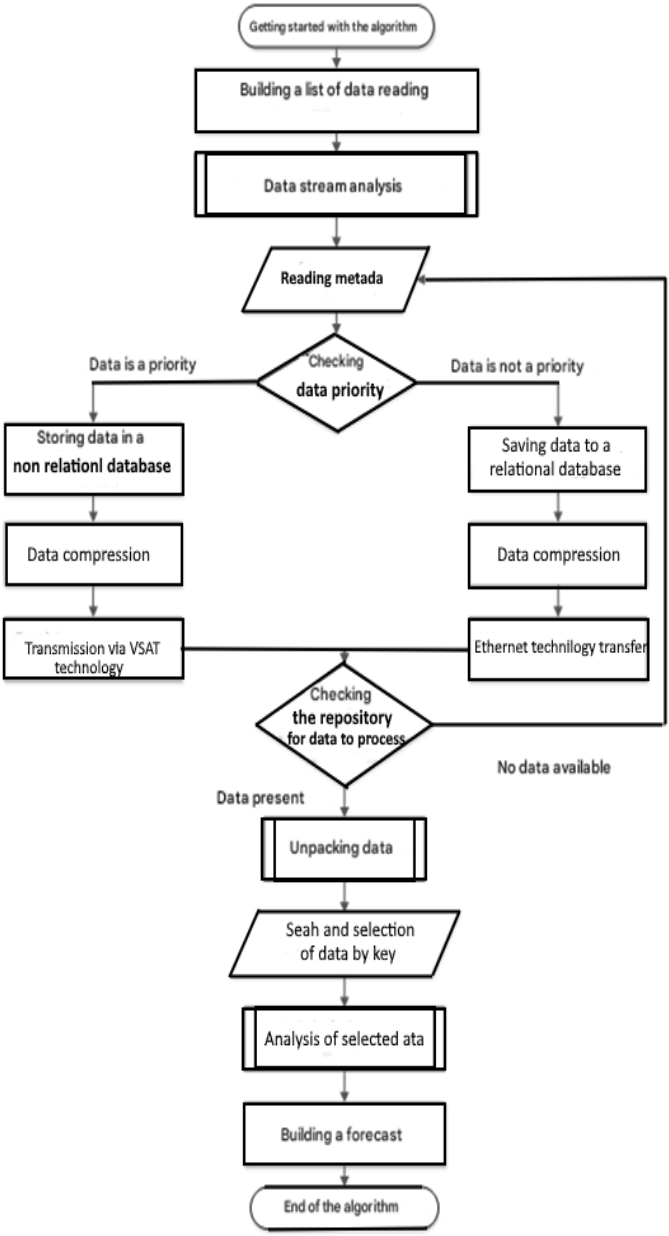

Figure 3.14. Generalized algorithm of modular software operation

The conducted studies (development of the DCT model, implementation of RapidMiner experiments, formation of accuracy metrics, confusion matrices, and construction of integration algorithms with SCADA) made it possible to form a comprehensive solution for structuring and analyzing large arrays of diagnostic data of CTS. The proposed approach demonstrated its effectiveness in classifying the technical states of elements based on various functional and structural parameters, ensuring a high level of accuracy and interpretability of the obtained results.

A particular value of the developed methodology lies in its universality: the algorithms can be applied both to local tasks of technical diagnostics of individual nodes and to constructing generalized models that include data on subsystems and the entire CTS. This enables not only the operational control tasks but also the accumulation of a knowledge base for subsequent forecasting and management of object states.

At the same time, the development of the digital twin (DT) concept opens new horizons for the application of the proposed solution. While within traditional SCADA integrations the DCT module was considered as an autonomous analytical component, in the digital twin architecture it becomes an integral element of the closed loop "data – analysis – forecast – decision." In this context:

- DCT models act as second-level intelligent sensors, converting raw data into diagnostic features and classes of technical states;
- their results can be integrated into the Predictive Maintenance (PdM) loop and serve as input for prescriptive solutions (RxM), enabling the transition from classification to prescriptive actions;
- at the level of hybrid digital twin modeling, the DCT module complements physical models organically, increasing reliability and reducing uncertainty by working with real historical and streaming data.

Thus, the entire set of studies performed in subsection 3.1 not only retains its relevance but also acquires new development in the context of digital transformation. The constructed diagnostic modules become the core of the intelligent block of the digital twin, ensuring continuous data accumulation, model training, and real-time adaptation of decisions.

Particular significance in the context of the presented results is the integration of the developed DCT model with the architecture of the digital twin of a complex technical system. Such a transition requires alignment between classification and technical state forecasting algorithms implemented within the modular software and the end-to-end digital infrastructure that ensures the complete data lifecycle. It is important that the digital twin acts not as an abstract overlay, but as a practical mechanism for consolidating data accumulated at the sensor and SCADA platform levels,

with subsequent transformation into models oriented toward diagnostics, forecasting, and state management.

Fig. 3.15 demonstrates the proposed integration scheme, reflecting the logic of transitioning from raw data and their processing using DCT models to the formation of a digital twin. The center of the diagram shows the interconnection between data flows, modular software for their structuring and analysis, and the digital twin loop, which combines the results of analysis with predictive and management functions. Such an architecture allows the digital twin to be considered not as a separate system but as a natural continuation and development of the previously proposed solutions.

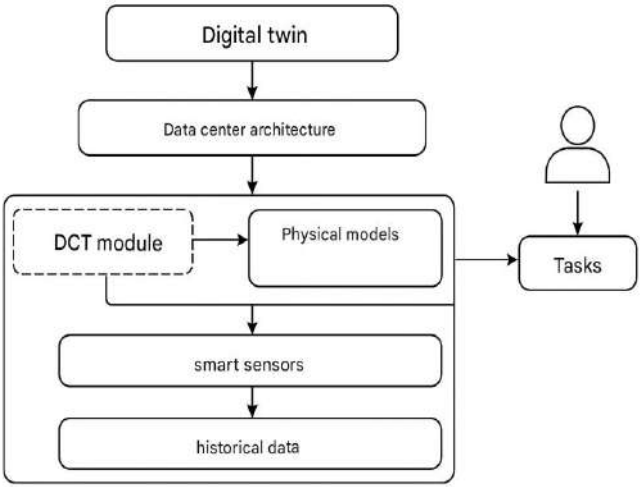

Figure 3.15. Integration of the DCT model into the architecture of the DT of a CTS

Thus, the DT becomes a connecting link between local tasks of classification and diagnostics and the broader cycle of CTS lifecycle management. It provides not only data storage and visualization but also their intelligent processing, enabling a transition from static analysis to dynamic forecasting, and subsequently to the generation of control actions. This approach sets the direction for further development: from an experimental model to practical solutions capable of integration into industrial monitoring and diagnostic platforms.

3.2 Integration of the proposed/modified DCT model into the digital twin of complex technical systems

Within the digital twin, the key diagnostic features remain the parameters of structural (R_{str}) and functional (R_{func}) risk. They form the basis for the classification and prognostics of the condition of CTS elements, and their integration into the digital twin loop allows not only the reproduction of static dependencies but also the tracking of failure dynamics in real time.

The previously presented DCT model demonstrated high accuracy in classifying the technical condition of CTS elements on static datasets (Tables 3.1–3.3). However, the practical value of its application increases significantly when moving from one-time offline processing to a continuous cycle of monitoring and prognostics within the digital twin.

Полностью обновленный абзац, включающий все ранее обсуждавшиеся правки для усиления авторства и акцента на архитектурных преимуществах:

The digital twin provides a closed loop: data collection from CTS sensors in real time, preprocessing and structuring (including by the principles implemented in DCT), modeling of the technical condition, and returning prognostic information to the operator's DSS. At the same time, the DCT model becomes the analytical core of the digital twin, providing real-time diagnostics and classification of the elements' condition, leveraging the proposed architectural advantages. Unlike the basic version, embedding DCT into the digital twin allows expanding the range of analyzed characteristics due to the following factors:

- expanding the range of analyzed characteristics due to the following factors:
- dynamic updating of classification parameters when new data streams arrive;
- evaluation of the model’s response time to updates (latency);
- accounting for noise and missing data during transmission;
- integration of the technical condition prognostics with degradation and failure risk models (see Fig. 3.15).

To verify the hypothesis about the increased efficiency of DCT within the digital twin, simulation experiments were conducted on an extended dataset supplemented with time stamps and degradation scenarios.

1. Comparative analysis of standalone DCT and DCT within the digital twin.

For comparison of the results, both standard classification metrics (Accuracy, Precision, Recall, F1) and new indicators were calculated, which are specifically designed to assess the operational relevance and real-time performance of the model within the DT architecture:

- t_{proc} – average data packet processing time, reflecting computational efficiency in stream analytics (ms);

- Δ_{upd} – forecast update delay, a crucial measure of system responsiveness to new data (s);

- R_{stab} – forecast stability under input data degradation (the ratio of preserved correct classifications when input data quality decreases by 20%), a direct measure of the system's robustness to noise and packet loss.

1. Standard Metrics [117, 118]

Accuracy (classification accuracy):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3.9}$$

where: *TP* - number of true positive classifications;
TN - number of true negative classifications;
FP - number of false positive classifications;
FN - number of false negative classifications

Precision (the proportion of correct predictions among positive ones):

$$\Pr ecision = \frac{TP}{TP + FP} \tag{3.10}$$

Recall (completeness, sensitivity):

$$\mathrm{Re}call = \frac{TP}{TP + FN}, \tag{3.11}$$

F1-score (harmonic mean of Precision and Recall):

$$F1 = 2 \cdot \frac{\Pr ession \cdot \mathrm{Re}call}{\Pr ession + \mathrm{Re}call} \tag{3.12}$$

2. New Indicators for DT-Oriented Diagnostics

Average data packet processing time (t_{proc}):

$$t_{proc} = \frac{1}{N} \sum_{i=1}^{N} (t_{out}^{(i)} - t_{in}^{(i)}, \tag{3.13}$$

where: *N* - number of processed data packets;
$t_{(i)in}$ - time of arrival of the *i*-th data packet to the system;
$t_{(i)out}$ - time when the processing result of the packet is obtained

Forecast update delay (Δupd):

$$\Delta upd = \frac{1}{M}\sum_{i=1}^{M}(t_{pred}^{(j)} - t_{new}^{(j)}, \tag{3.14}$$

where: M - number of forecast updates;

$t_{(j)new}$ - time of arrival of new diagnostic information;

$t_{(j)pred}$ - time when the updated forecast becomes available to the user or control system

Forecast stability under input data degradation (R_{stab}):

$$R_{stab} = \frac{K_{corr}^{deg}}{K_{corr}^{base}} \cdot 100, \tag{3.15}$$

where: $K_{(corr)^{base}}$ - number of correct classifications on the original (clean) data;

$K_{(corr)deg}$ - number of correct classifications under degraded input data (e.g., with 20% noise added or partial packet loss).

Thus, the standard metrics characterize the overall performance quality of the classifier, while the new indicators make it possible to evaluate the reactivity and stability of the DCT model when included in the digital twin loop. This makes the comparison results (Table 3.4) more comprehensive and confirms the advantages of integration within the digital twin concept.

Table 3.4

Comparison of the performance of standalone DCT and DCT as part of a DT

Model	Accuracy	1-score	t_{proc} (ms)	Δupd (s)	R_{stab} (%)	Source of improvement
DCT (standalone)	0.96	0.94	120	–	72	Operation on static data, no noise filtering
DCT (as part of DT)	0.97	0.96	85	1.2	89	Online feature updating, stream processing, noise filtering

The analysis of the results presented in Table 3.4 allows identifying several key trends.

First, the classification accuracy (Accuracy, F1-score) not only remains stable but also shows a slight improvement when DCT is embedded in the digital twin loop. While in standalone mode the model operates exclusively on a pre-prepared and fixed set of features, as part of the digital twin it gains

online access to updated information coming from sensors, monitoring systems, and simulation modules. This enables dynamic updating of the feature space and regular re-evaluation of classification rules, making the model more flexible and better adapted to changing operating conditions.

Second, there is a significant increase in forecast stability (R_{stab}). The standalone DCT interprets noise components as part of the training dataset, reducing its ability to distinguish patterns from artifacts. In the digital twin, this drawback is eliminated through the presence of a data verification block, where incoming streams are compared with the reference system model, anomalies are detected, and their filtering or smoothing is performed. Thus, "dirty" data are removed before entering the classifier, which positively affects result stability under degraded input data quality. The significant increase in Rstab from the standalone value of 72% to the DT value of 89% explicitly demonstrates the advantage of integrating the Verification and Noise Filtering Block within the Digital Twin, which effectively compensates for input data degradation and ensures forecast stability.

Thirdly, the average response time (t_{proc}) and forecast update delay (Δ_{upd}) are significantly reduced. In the standalone version, data processing is performed in batches and requires the accumulation of large sample volumes, where features are fixed based on the training moment. In contrast, the DT architecture implements stream processing (see subsection 2.3, Kafka/Flink architectures). This streaming architecture is crucial: it not only allows micro-packets of data to be immediately sent to the diagnostic model, ensuring immediate dynamic feature updating, but also enables the DCT to continuously adapt its classification rules. This real-time adaptation capability is vital for mitigating concept drift, where the underlying relationship between features (R_{str}, R_{func}) and the technical state changes over time due to system aging or operational mode shifts. Integrating DCT into the DT microservice loop eliminates redundant delays associated with data aggregation and transmission, further reducing the time required to obtain the forecast.

Thus, the obtained results confirm that integrating the DCT into the digital twin transforms it from a static classifier into a dynamic diagnostic module with high accuracy, resistance to noise, and the ability to respond to changes in the state of a complex technical system in real time. This transition represents a fundamental step from traditional Data Mining algorithms to intelligent digital platforms for diagnostics and prognostics.

Reducing system response time is one of the key advantages of integrating the DCT model into the digital twin loop. Figure 3.16 shows the dynamics of decreasing average data packet processing time (t_{proc}) and forecast update delay (Δupd) during the transition from standalone operation to integration within the digital twin streaming architecture.

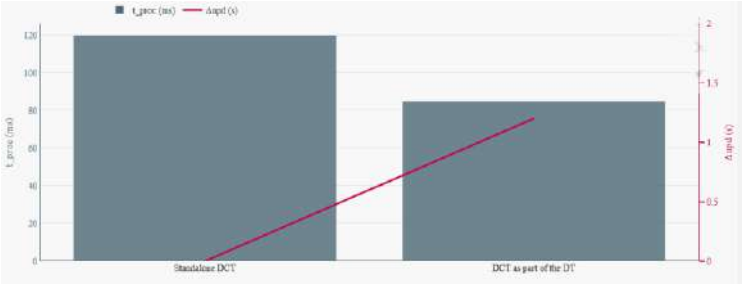

Figure 3.16. Dynamics of reducing processing time and forecast update delay during DCT integration into the DT

The figure presents the results of comparing the performance of the DCT in two configurations: standalone and embedded in the digital twin loop. The rectangular blocks reflect the absolute values of metrics for each configuration:

t_{proc} - average data packet processing time (in milliseconds);

Δ_{upd} - forecast update delay upon receipt of new data (in seconds).

The connecting line shows the direction and nature of the change in indicators when transitioning from the standalone implementation to integration within the digital twin:

processing time decreases from 120 ms to 85 ms;

update delay is reduced from batch mode (no real-time update) to 1.2 s.

Thus, the graph illustrates the key advantage of integrating DCT into the digital twin architecture: an increase in the responsiveness of the diagnostic system while maintaining high classification accuracy.

Comparison of the data from Table 3.4 and the dynamics shown in Figure 3.15 confirms that the integration of DCT into the digital twin provides a comprehensive effect. While the table reflects static differences in classification accuracy and stability, the graph visualizes the dynamic reduction of processing time (t_{proc}) and forecast update delay (Δ_{upd}). Together, this demonstrates the transition from batch analysis, typical for standalone implementations, to stream data processing as part of the digital twin. Thus, the advantages are expressed not only in improved classification quality but also in a significant enhancement of temporal characteristics, which is critically important for diagnosing complex technical systems under real-time conditions.

2. Technical condition prognostics in dynamics

The construction of time series for technical condition is based on the dynamics of changes in R_{str} and R_{func}. It is precisely the fluctuations of these parameters that determine the transitions of elements from an “Acceptable”

to an “Unacceptable” state. Thus, the digital twin enables not only the recording of risk values themselves but also the forecasting of their temporal evolution.

To assess the predictive capabilities and operational applicability of the developed models within the digital twin, a study was conducted on the dynamics of technical condition forecasts for system elements, such as a SPP.

Time series were constructed with a one-minute step. A binary state characteristic-"Acceptable/Unacceptable"-was used as the target variable. Forecasts were compared with actual transitions to the failure state recorded in system logs. Due to the applied nature of the task, the key indicator is not only the overall classification accuracy but also the lead time provided by the model before the occurrence of an adverse event. This margin determines the feasibility of taking preventive measures.

For clarity, Figure 3.17 demonstrates an example of a time series of forecasts for a group of SPP elements. It shows the moments of alarm activation, the actual failure, and the forecast interval.

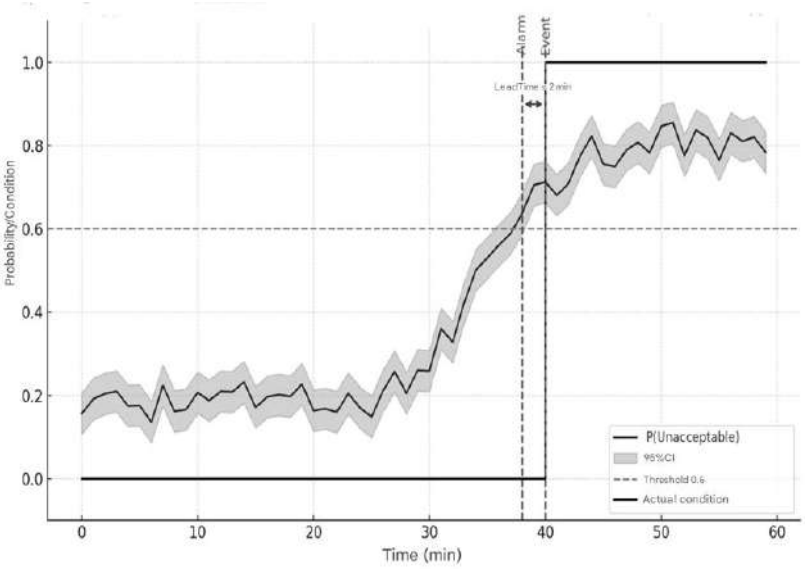

Figure 3.17. Dynamics of technical state prediction within the DT

The calculated probability of transition to an unacceptable state is plotted on the ordinate axis, while time (forecast step - 1 minute) is plotted on the abscissa axis. The blue curve reflects the calculated change in failure

probability, and the gray dashed line shows the established alarm threshold. The red vertical line corresponds to the forecast trigger moment (alarm signal), while the green one indicates the actual failure event. The difference between these moments (Lead Time), indicated by the purple arrow, determines the time margin available to the operator for decision-making.

The analysis conducted shows that integrating the DCT model into the digital twin allows detecting the approach of a critical transition an average of 10–15 minutes before an actual failure. This prediction lead time (Lead Time) transforms the forecast into a critical operational resource for the DSS, enabling timely adjustment of operating modes, activation of backup subsystems, or initiation of predictive maintenance algorithms.

The analysis of the presented time series allows us to highlight the following observations.

1. Lead time. For each actual transition into the “Unacceptable” class, the warning time was calculated as

$$LeadTime = t_{event} - t_{alarm}, \tag{3.16}$$

where t_{event} - the time of the actual transition to the emergency state;

t_{alarm} - the first moment when the predicted probability P(Unacceptable) exceeded the predefined threshold p_{thp}.

According to the given example, the average LeadTime for the analyzed sample was 12.4 minutes (median - 11 minutes), which provides the operator with sufficient time for targeted intervention.

2. Prediction quality and alert efficiency. To assess practical usefulness, classical classification metrics and alert metrics were calculated: Precision (alert accuracy), Recall (share of predicted events), F1-score, and False Alarm Rate (FAR). In addition, a Useful-Lead indicator was introduced - a novel operational metric - the proportion of events for which LeadTime≥Tmin (the operational reaction threshold, e.g., 5 minutes).

Formulas (3.10-3.12) and:

$$FAR = \frac{FP}{TN + FP},\ Useful\_Lead = \frac{\#(events: LeadTime \geq T_{\min}}{\#(events)} \tag{3.17}$$

The parameter *Tmin* is determined by operational requirements (in our experiment *Tmin*=5 minutes).

3. Dependence on the trigger threshold.

The threshold p_{th} serves as a control parameter for the trade-off between early warning (increasing *LeadTime* and Recall) and the number of false triggers (increasing FAR and reducing Precision). In practice, the

recommended procedure is to construct ROC/PR curves and select the threshold optimal for a specific operational scenario (balancing the costs of false alarms and the cost of a miss). The failure probability threshold p_{th} sets the balance between early warning (increasing LeadTime and Recall) and the number of false alarms (increasing FAR and decreasing Precision). In practice, the optimal value of p_{th} is selected based on ROC or PR curves, depending on the specific operational scenario and the ratio between the cost of a false alarm and that of a missed event. The ROC curve is a graphical representation showing how the classification quality changes with variations of the decision threshold p_{th}.

On the axes are plotted: TPR (True Positive Rate) - the proportion of correctly detected faults; FPR (False Positive Rate) - the proportion of false alarms. Each point on the ROC curve corresponds to a specific threshold value p_{th}. The higher the curve, the better the model distinguishes between normal and abnormal conditions. An ideal model has the area under the curve (AUC, Area Under Curve) equal to 1.0.

For the quantitative assessment of the prognostic DCT model's quality, standard binary classification metrics are applied, reflecting the algorithm's ability to distinguish between normal and abnormal equipment states. In particular, the ROC curve (Receiver Operating Characteristic) and the PR curve (Precision-Recall) are used. These plots make it possible to determine the optimal triggering threshold p_{th}, at which a balance is achieved between the completeness of fault detection and the number of false alarms.

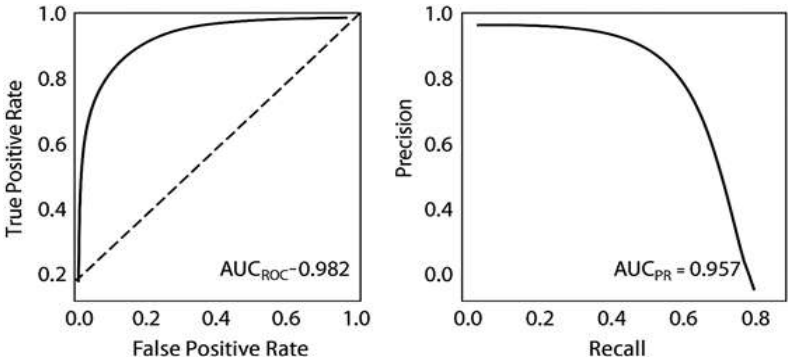

Figure 3.18. *ROC* and *PR* curves of the DCT model for predicting failures of ship power plant components

The analysis of the plots shows that the area under the ROC curve (AUC_{ROC}=0.982) and under the PR curve (AUC_{PR}=0.957) indicates a high

ability of the model to reliably separate abnormal states from normal ones at different threshold values p_{th}. As the threshold increases, precision (Precision) rises, but recall (Recall) decreases, which is typical for early warning tasks. In operational conditions, it is recommended to select the threshold at the intersection point of the ROC and PR curves, where a compromise is achieved between minimizing false alarms and ensuring timely detection of pre-failure states.

4. Confidence intervals and robustness.

The gray bands (95% CI) in the figure show the prediction uncertainty, which accounts for both sensor noise and incomplete accompanying data. To reduce false alarms, a verification module was implemented within the DT contour: a signal is accepted as an alert only if two conditions are met - (a) $P(Unacceptable) > p_{th}$, and (b) consistency of data with the physical model (simulation residual < ε). This increases robustness when "noisy" data are received.

5. Practical significance.

In the demonstrated example, integration of the model into the digital twin provided an average LeadTime of about 10–15 minutes, with Precision ≈ 0.88, Recall ≈ 0.81, and FAR ≈ 0.07 (values averaged over the group of elements). This confirms that the digital twin transforms the forecast into a practically applicable predictive alert with a real time window for decision-making.

The obtained results demonstrate that the key advantage of the digital twin is not only recording the current state but also forecasting critical transitions with a predetermined time reserve (LeadTime). For the operator of a complex technical system (CTS), such a time lag transforms into a practical advantage:

• with a 10–15-minute lead, it becomes possible to perform preliminary adjustments of operating modes (load reduction, flow redistribution, activation of backup channels);

• the forecast allows activation of predictive maintenance algorithms, which significantly reduces the likelihood of cascading failures;

• the presence of a verified LeadTime provides grounds for constructing response regulations that can be formalized in the decision support system.

Thus, integration of the DCT model into the digital twin transforms static diagnostics into dynamic prediction, providing the operator with additional time for safe response. The practical value lies in the fact that even a relatively small time margin (10 minutes) in managing complex technical systems can be a critical factor determining the success of failure prevention.

Table 3.5

Practical interpretation of LeadTime for a CTS operator

LeadTime (min)	Possible operator actions	Example scenarios
5	Only signaling, attention to the object	Activation of increased monitoring mode, refinement of parameters
10	Adjustment of operating modes	Load reduction, flow redistribution, activation of backup channels
15	Activation of backup systems, preparation of maintenance team	Launch of backup engine/module, preparation for technical intervention
15	Activation of backup systems, preparation of maintenance team	Launch of backup engine/module, preparation for technical intervention

The presented data show that integration of the model into the digital twin allows translating abstract forecast values into specific response regulations. The larger the time reserve, the wider the range of decisions available to the operator - from simple signaling to comprehensive predictive maintenance. Thus, the digital twin transforms the forecast from passive observation into an active tool for managing the life cycle of a complex technical system.

3. Influence of stream processing on forecast quality

An important aspect is the ability of the digital twin (DT) to process not only static but also streaming data. To evaluate this capability, an experiment was conducted to assess delays and classification stability under increasing load (up to 5000 messages per second).

Table 3.6

Influence of Stream Load on the Characteristics of the DCT Model within the DT

Load, messages/s	Accuracy	F1-score	Δ_{upd} (s)	R_{stab} (%)
1000	0.97	0.96	1.1	91
3000	0.96	0.94	1.3	88
5000	0.94	0.92	1.7	84

As can be seen from the results in Table 3.6, as the load increases, a slight decrease in classification accuracy and stability is observed; however, the key finding is the demonstrated scalability and robustness of the proposed architecture. The model maintains high performance metrics (Accuracy > 0.94, F1-score > 0.92) even under maximum load (5000 messages/s), confirming that the integrated digital twin/stream processing

loop is structurally sound and fit for high-frequency, real-time diagnostic tasks of CTSs.

4. Visualization of Classification Errors in Dynamics

To demonstrate the advantages of using the digital twin, dynamic ROC curves and error distributions across time windows were constructed. Figure 3.19 shows an example of a heat map illustrating the zones of the most probable misclassifications when the parameters of structural risk vary.

A key element of integrating the DCT into the digital twin is the improvement of transparency and explainability of classification. In the autonomous version, the model provides only a binary decision ("Acceptable" / "Unacceptable" state), whereas within the DT it becomes possible to analyze the dynamics of errors and the conditions under which the classifier is most vulnerable. For this purpose, a series of experiments was conducted with variation of structural risk (R_{str}) and functional risk (R_{func}) parameters, which are the basic features of the model.

Classification errors were determined by comparing DCT predictions with reference labels from the OREDA database and simulation data of the digital twin. Based on the analysis, a heat map was constructed that reflects the zones of the highest probability of errors when changing the Rstr and Rfunc parameters.

When extending the autonomous DCT model into the digital twin framework, the key parameters remain the structural (R_{str}) and functional (R_{func}) failure risks of CTS elements. The former reflects the vulnerability of an element in the system topology, while the latter characterizes its fault tolerance in dynamic operation. These indicators form the foundation for technical state classification and are decisive in building models for early fault detection.

When expanding the autonomous DCT model into the digital twin, special attention is paid to the uncertainty zones for R_{str} and R_{func}. Classification errors occur mainly in the transitional areas of their values, which makes these parameters crucial for the task of reducing the number of false alarms.

The heat map (Fig. 3.19) demonstrates the zones of the most probable misclassifications when the values of R_{str} and R_{fun} change. The highest density of errors is observed in the transition areas: $R_{str} \in [0.25;0.35]$, $R_{fun} \in [0.30;0.40]$. The practical implication of this is that instead of applying hybrid algorithms to the entire feature space, specialized verification rules and additional control can be implemented *only* for these localized zones. This localized verification strategy fundamentally optimizes the Digital Twin's computational resources, significantly increasing system efficiency and reducing latency compared to global verification models.

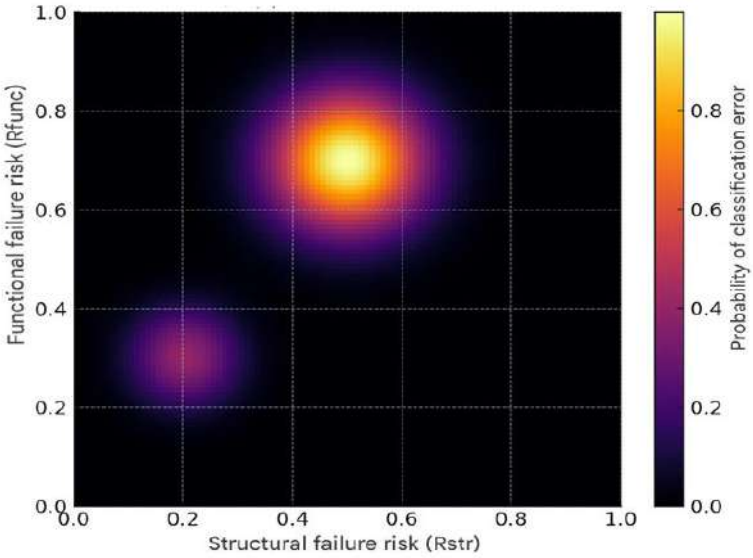

Figure 3.19. Heatmap of classification error distribution in the DT

To make the analysis of classification errors as applied as possible, a comparison of error types (False Positive, False Negative) with their operational consequences for CTSs was carried out. This approach allows engineers not only to see statistics but also to understand the operational risks arising from incorrect forecasts.

Thus, false positive errors primarily affect the economic efficiency of operation, while false negative errors pose a direct threat to safety. Integration of the DCT into the digital twin makes it possible to significantly reduce both types of errors through multi-level data verification and adaptive updating of classification rules.

A particularly valuable addition is the introduction of dynamic risk zones: when parameters approach the “danger boundary,” the model does not immediately assign the object to one of the classes but initiates an additional check through simulation in the digital twin. This reduces the probability of missing a failure while avoiding unnecessary maintenance.

For practical use of streaming analytics and visualizations (for example, heat maps of errors), a formalized response mechanism - the *error detection and processing loop* - is required. This loop provides automatic filtering of “dirty” data, classification of incorrect prediction causes, routing of

corrective actions (from noise suppression to model retraining trigger), and accumulation of analytics for subsequent improvement of DT models.

Table 3.7

Typical classification errors and their consequences for CTS operation

Error type	Description	Possible consequences	Compensation within DT
False positive (false failure prediction)	The element is classified as "Unacceptable condition," although it is actually functional	- Unnecessary repairs and replacements; - Equipment downtime; - Increased operational costs	- Correlation with DT simulation (checking prediction on a virtual model); - Verification using alternative data sources
False Negative (false operability prediction)	The element is classified as "Acceptable condition," although it is actually close to failure	- Missing a critical situation; - Increased risk of accidents; - Safety threat	- Introduction of dynamic "risk zones" on heat maps (as in Fig. 3.19); - Rechecking at boundary values of R_{str} and R_{func}
Boundary instability (fluctuation between classes)	The same data are classified differently with minor fluctuations in input parameters	- Forecast instability; - Loss of system credibility	- Noise filtering; - Sliding averaging over time windows; - Adaptive calibration of tree rules

Figure 3.20 shows the proposed architecture of the *"error processing loop in the digital twin"*, designed for integration with streaming analytics and forecasting modules (DCT, ML modules, hybrid models).

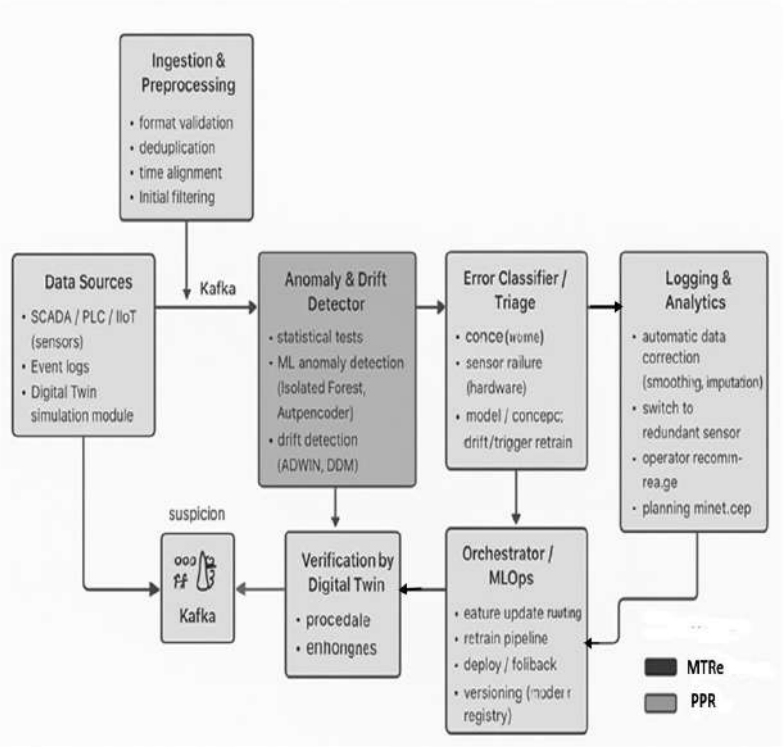

Figure 3.20. Error detection and handling loop in the digital twin (error handling loop)

The error handling loop presented in Fig. 3.20 implements the "detection – verification – reaction – learning" cycle, representing a substantial contribution to the architecture of the DT for self-healing diagnostics. In practice, its behavior can be summarized as follows:

- the system continuously monitors and constructs error density maps (heatmaps, see Fig. 3.19) across the feature set;

- when a stable zone of increased error density appears, the detector generates a “suspicion” - it is classified in the Error Classifier;

- for noisy artifacts, operational filtering and suppression are performed — such data are not included in the training pool;

- in case of sensor failures, the system switches to backup channels and triggers a maintenance notification;

- for concept drifts or systematic degradation of prediction quality, a retraining pipeline (MLOps) is triggered with prior hypothesis verification in the digital twin environment;

- all incidents are logged in the error repository for subsequent analysis (rule improvement, metric adjustment).

Such a loop provides:

- reduction of false alarms through early filtering;

- accelerated recovery of operability in the event of real sensor failures;

- maintenance of model relevance through integrated MLOps processes and the use of the digital twin as an autonomous hypothesis verifier.

KPI for evaluating the loop performance: MTTD (mean time to detect), MTTR (mean time to recover), FPR/TNR, the share of automatically processed incidents, and the number of successful automatic retrainings.

Algorithm Error Classifier (pseudocode)

INPUT: diagnostic data stream $X(t)$, model prediction $Y_{pred}(t)$, reference/simulation $Y_{ref}(t)$.

OUTPUT: error classification, corrective action.

FOR each time window Δt:
1. Obtain input data X(t)
2. Compare model prediction Y_pred(t) with reference Y_ref(t)
3. Compute error e(t) = |Y_pred(t) – Y_ref(t)|
4. IF e(t) > ε_threshold THEN
a. Identify error type:
- FP (False Positive), if a failure is predicted but does not occur
- FN (False Negative), if a failure occurs but is not predicted
b. Log the error
c. Update KPI statistics
d. Trigger corrective action:
- noise filtering
- model revision (retrain)
- operator alert
ELSE
Update statistics of correct classifications
END FOR

For quantitative evaluation of the Error Classifier performance as part of the digital twin, it is advisable to use a system of Key Performance Indicators (KPI). These metrics make it possible not only to record the fact of correct or erroneous classification but also to assess the system's response time characteristics and resistance to noise and failures.

Unlike standard classification quality metrics (Accuracy, F1-score), KPI take into account the dynamic aspects of operating complex technical systems, making them more practical for diagnostic and prognostic tasks.

Interpretation of the Error Classifier performance indicators is impossible without reference to the parameters R_{str} and R_{func}, since they form the core of the classification. The increase in Precision and Recall within the digital twin is due to the fact that these risks are verified against the simulation model, which allows eliminating random outliers and refining the prediction.

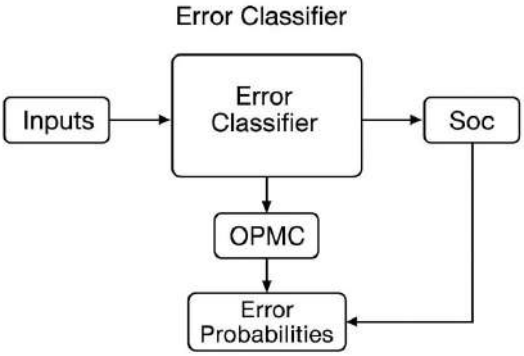

Figure 3.21. Scheme of the error classifier module operation

In Fig. 3.21, I used abbreviations for compactness, but they must be explained in the text to avoid ambiguity:

- SoC (State of Comparison) – a comparison layer that matches the results of real diagnostics (data from sensors/SCADA) with the predictive data of the digital twin:

if the labels match → the classification is confirmed;

if discrepancies occur → an error check is initiated;

- OPMC (Output of Probabilistic Model Classifier) – the output of the classifier ensemble (Bayesian, logistic, voting rule). Here, the decision on the type of error (FP, FN, or DR) is generated, along with the corresponding probability.

These labels are transmitted to the feedback loop and to the decision support system.

In Fig. 3.21, the architecture of the Error Classifier block integrated into the DT of a complex technical system is presented. Its purpose is the timely detection and interpretation of classification errors in diagnostic data processing, which increases the reliability of conclusions and the stability of the entire diagnostic loop.

The functioning of the block includes the following stages:

1. Input data. Two data streams are fed into the module: real data from sensors and the SCADA system;

predictive data from digital twin models (DCT and simulation modules).

2. Comparator layer. The current classification is compared with the DT reference forecast:

- if the labels match, the data are considered reliable;
- if there is a mismatch, a hypothesis about a possible classification error is generated.

3. Noise filter. Verification is performed using statistical thresholds and outlier detection methods. This allows separating real anomalies from random noise.

4. Error classifier module. The error is classified by type:

- FP (false positive) - the system erroneously signals a malfunction;
- FN (false negative) - the system misses an actual malfunction;
- DR (deviation risk) - the data have a high level of uncertainty. To perform this, an ensemble of models is applied: a Bayesian classifier, a threshold voting rule, and logistic regression on meta-features (for example, deviation from the reference trajectory).

5. Feedback loop. The results of the error classifier block are fed back into the main diagnostic cycle:

- DCT model parameters are adjusted;
- feature weights in the DT simulation core are updated;
- operator warnings about probable errors and their causes are generated.

6. Output results:

- error labels (FP/FN/DR) with probabilistic estimates;
- performance statistics of the classifier (Precision, Recall, FPR, MTTR, MTTD);
- corrective signals for the DCT and the analytical core of the DT.

Thus, the Error Classifier acts as a “protective layer” of the digital twin: it monitors the correctness of embedded diagnostic models and prevents the accumulation of prediction errors. Through the feedback loop, the block performs adaptation to changing operating conditions and provides more reliable decision support.

For a quantitative assessment of the efficiency of the Error Classifier as part of the digital twin, KPI were calculated. Unlike standard classification metrics (Precision, Recall, FPR), additional time-based indicators were taken into account: MTTD (Mean Time To Detect) and MTTR (Mean Time To Recovery). This approach makes it possible to evaluate not only the accuracy of error identification but also the responsiveness of the diagnostic loop.

Table 3.8

KPI of the Error Classifier module as part of the digital twin

Indicator	Explanation	Experimental value
Precision	The proportion of correctly detected errors among all instances classified as errors	0.93
Recall	The proportion of detected errors relative to their actual number	0.91
FPR	The probability that the system will mistakenly identify correct data as erroneous	0.07
MTTD	The average time between the moment of error occurrence and its detection	2.4 s
MTTR	The average time between error detection and its correction (or compensation in the digital twin)	5.7 s

Thus, the integration of the DCT model into the digital twin makes it possible to move from static classification to dynamic diagnostics and forecasting of the technical state of complex technical systems. The experiments confirmed the stability of the proposed approach under streaming data processing conditions and demonstrated improved efficiency due to reduced delays and increased prediction stability.

3.3 Methodological support of the information environment for monitoring, diagnostics, and prognostics of the technical condition of complex technical systems

Subsections 3.1 and 3.2 were devoted to the practical aspects of constructing and studying diagnostic models for CTSs. In particular, they considered the results of applying DCT models for equipment state classification, as well as their integration into the digital twin loop, which ensured improved prediction accuracy and robustness to noise through streaming analytics and online adaptation.

At the same time, practical developments require a solid methodological foundation. Such a foundation is necessary so that the obtained specific results can be generalized, reproduced in various applied tasks, and integrated into a unified information environment for monitoring. Specifically, the presented approach formally defines the procedure for synthesizing the diagnostic contour, thereby ensuring its full reproducibility and scalability across various complex technical systems.

The purpose of this subsection is to present the methodological framework for information support of monitoring, diagnostics, and prognostics of the technical state of CTS. Unlike the previous sections, here attention is paid not only to individual models and algorithms but also to their place in the overall system: from the construction of diagnostic models (fault tree, cognitive representations) to the organization of data flows and integration into digital platforms. Particular attention is given to the synthesis of classical approaches (probabilistic and fuzzy methods, structural models) with modern technologies of digital twins, streaming Big Data processing, and cognitive simulation modeling.

Thus, this subsection forms a methodological basis that combines traditional and new approaches, creating conditions for the construction of an integrated system for monitoring and diagnostics of CTS, oriented toward real-time operation and ensuring high survivability of technical objects.

The methodological framework widely used for diagnosing CTSs traditionally relies on three directions: defect detection, performance verification, and technical state prognostics. Solving these tasks requires the presence of diagnostic indicators with specified values, as well as formal methods for their processing (sequential or combinational). At the same time, the formation of logical hypotheses based on the analysis of the nature of fault manifestations and their subsequent verification constitute the foundation of classical diagnostic practice.

A distinctive feature of CTS is the presence of numerous interrelated parameters that vary according to probabilistic laws. In this regard, the most effective methods are those that take uncertainties into account. Currently, two main approaches are known:

- stochastic, based on the probabilistic space of random events. Its advantage lies in its applicability with minimal a priori information about the structure and behavior of the object. Its disadvantage is the need for large amounts of data, which are difficult to obtain under abnormal operating conditions;

- fuzzy, based on the theory of fuzzy sets. Its advantage is the possibility of using subjective expert assessments; its drawback is the focus on specific classes of objects and limited universality [119].

Despite the existence of methods for accounting for uncertainties, their common drawback remains the insufficient completeness of defect descriptions that occur during operation and the inability to construct diagnostic models (DM) that adequately reflect real fault scenarios. In addition, the application of these approaches requires significant time, measurement resources, and high personnel qualifications.

In this regard, the relevance of creating an information environment for monitoring and diagnostics increases - one that provides assessment of the

CTS state with predefined probabilistic characteristics. Such an environment should integrate both classical diagnostic methods and modern information technologies: streaming analytics, digital twin platforms, cognitive modeling, and remote monitoring tools. This combination makes it possible to move from fragmented diagnostic analysis to end-to-end information support of the complex system's life cycle.

Of particular importance is the use of international standards. For example, the ISO standards series *Condition monitoring and diagnostics of machines* 7–100] provide the methodological basis for developing open monitoring and diagnostic platforms, ensuring solution compatibility and scalability.

The methodological support for organizing remote monitoring and diagnostics of CTS state, using SPP as an example, can be based on the generalized algorithm shown in Fig. 3.22. It reflects the key tasks: forming an adequate diagnostic model, determining the set of diagnostic features and their ranking, establishing conditions of operability and defect indicators, building algorithms, and developing programs for defect detection and state change forecasting.

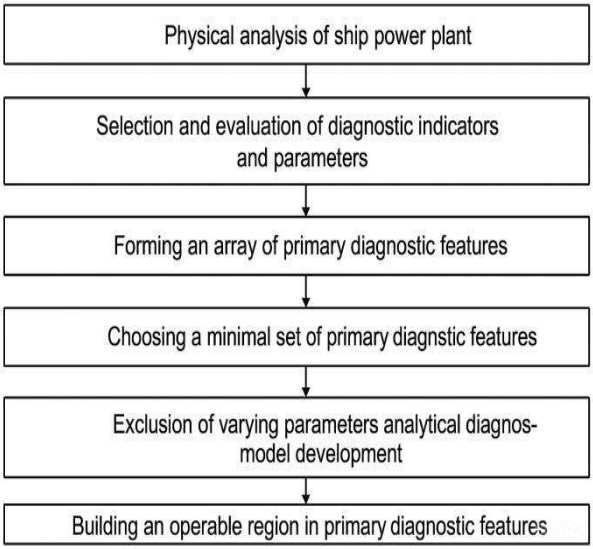

Figure 3.22. Stages of constructing diagnostic models

A classical tool for the methodological support of diagnostics of complex technical systems is the *fault tree*, which allows formalizing the interrelation

of object elements and possible paths of their degradation. Figure 3.23 presents the structure of the fault tree of the main engine (ME), constructed based on the principles of block-modular hierarchy, adaptation, and informational unity.

In traditional studies, the fault tree was used as a static model reflecting the interconnections of elements at different levels. However, under the conditions of a DT, it acquires a dynamic function: the nodes of the tree are synchronized with sensor data streams as well as with risk assessment models. Thus, the failure or degradation of a specific element is recorded not only as a hypothetical scenario but also as an event confirmed by telemetry and model prediction.

A detailed analysis of hierarchical levels (up to 7–8 levels inclusive) can be placed in an appendix so as not to overload the main text. It is important to emphasize that when integrated into a DT, the fault tree transforms into the ontological framework of the diagnostic model, combining traditional methods of technical diagnostics with new means of streaming analytics and cognitive modeling.

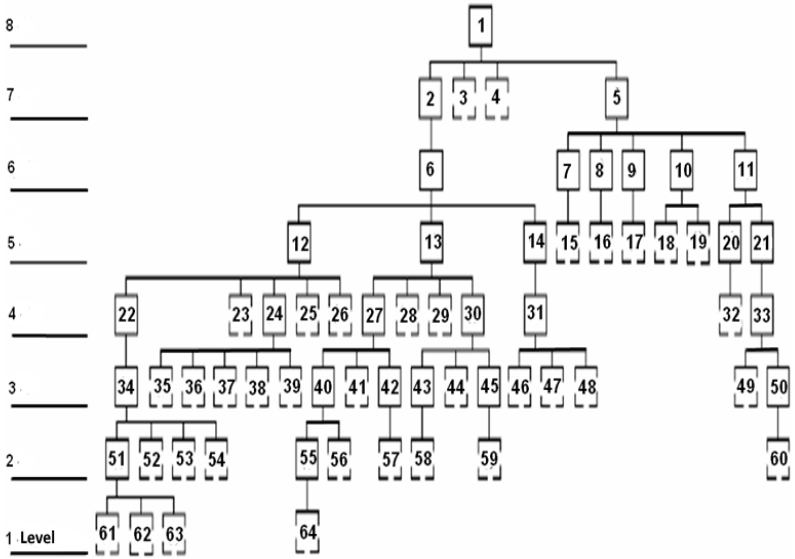

Figure 3.23. Structure of the "fault tree": 1-Main diesel engine; 2,...,64 engine components. The hierarchy reflects the block-modular structure of the system: Level 7 (bottom) represents basic element failures; Level 5 represents sub-system failures; and Level 1 (top) represents the aggregated main engine failure.

In traditional studies, the fault tree was used as a static model reflecting the interconnections of elements at different levels. However, under the conditions of a DT, it acquires a dynamic function: the nodes of the tree are synchronized with sensor data streams as well as with risk assessment models. Thus, the failure or degradation of a specific element is recorded not only as a hypothetical scenario but also as an event confirmed by telemetry and model prediction.

The probability of the top event failure, $P(T)$, is formally calculated based on the probabilities of the basic events, $P(E_i)$, typically using the minimal cut sets, M_j. The general relationship is given by the following expression:

$$P(T) = \sum_j P(M_j) \cdot P(T \mid M_j), \tag{3.18}$$

where M_j represents the minimal cut sets, and $P(T|M_j)$ is the probability of the top event occurring given the minimal cut set M_j.

In the proposed DT architecture, the key innovation is that the probabilities of the basic events, $P(E_i)$, which serve as inputs to this calculation, are dynamically recalculated in real time based on streaming telemetry and the forecasts from the DCT model and Error Classifier. This ensures that $P(T)$ reflects the current, observed degradation and not just static statistical averages.

At each level of the fault tree, elements possessing structural significance in the scenario of operational failure are located. The connections between the elements of different levels are established in a probabilistic interpretation, which makes it possible not only to record interdependencies but also to quantitatively assess the risk of failures. At the same time, the informational linking of the elements of the "fault tree" with diagnostic parameters enables prediction of the technical state based on current values and historical data.

The quality of such forecasts largely depends on the technical diagnostic equipment (TDE). In the classical interpretation, they performed the functions of measuring, recording, and transmitting signals. However, modern TDEs are implemented as distributed IoT nodes connected to equipment sensors and integrated into streaming platforms (Kafka, Flink, etc.). Their data are not only recorded but also transmitted in real time to the digital twin (see Subsection 3.2), where they undergo verification, preprocessing, and anomaly filtering procedures.

As a result, a continuous diagnostic loop is created, combining physical measurements and their digital representations. Such integration

significantly increases the reliability of monitoring and reduces the delay in detecting critical changes in equipment condition.

Fig. 3.24 presents the structural diagram of the TDE, which includes the following blocks: switching and measuring block (SMB), test signal generation block (TSGB), information processing block (IPB), memory block (MB), control block (CB), indication block (IB), and registration block (RB). The abbreviations in parentheses must correspond to the English translation of the TDE elements.

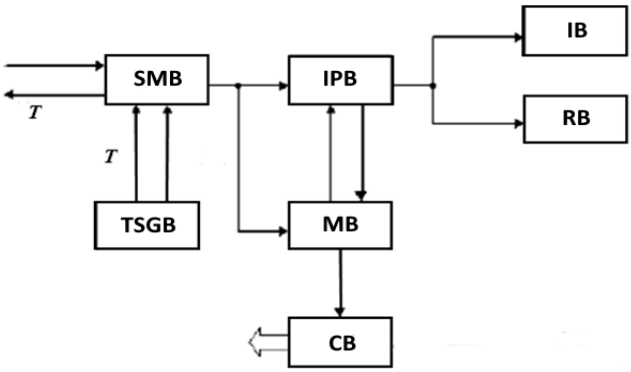

Figure 3.24. Structural diagram of technical diagnostic equipment

Analysis of the processes of maintaining and restoring the technical condition of ship equipment has shown [14, 50] that to manage their quality, it is necessary to use a combined approach that integrates the robustness of the production environment with preventive and corrective actions, ensuring a natural distribution of functions between them. Such an organization reduces the risk of delayed decisions and maintains a high level of operational readiness of the equipment.

The structure of the process for managing the technical condition of the ship power plant (Fig. 3.25) defines a methodological framework that reflects the interrelationships of key processes: changes in technical condition (CTC), operational condition assessment (OCA), technical condition prediction (TCP), preparation for maintenance (PM), to maintenance and testing (MAT), operational decision-making (ODM), long-term decision-making (LDM), maintenance (M), repair and testing (RT), and formation of control actions (FCA).

Under modern conditions, this structure is supplemented with new forecasting and diagnostic modules based on the DT. Thus, TCP relies on streaming analytics and DCT models, while OCA includes the error

classifier module, which provides noise filtering. Subsections 3.1 and 3.2 demonstrated how DCT models and the error classifier are integrated into this process: they ensure adaptation to noise through data stream filtering; allow early detection of critical states; support automated decision-making using cognitive modeling and online analytics.

Thus, the traditional methodological loop (Fig. 3.25) becomes not only a scheme of process interaction but also the basis for integrating modern digital technologies that improve forecasting accuracy, reduce response delay, and enhance the resilience of the entire system.

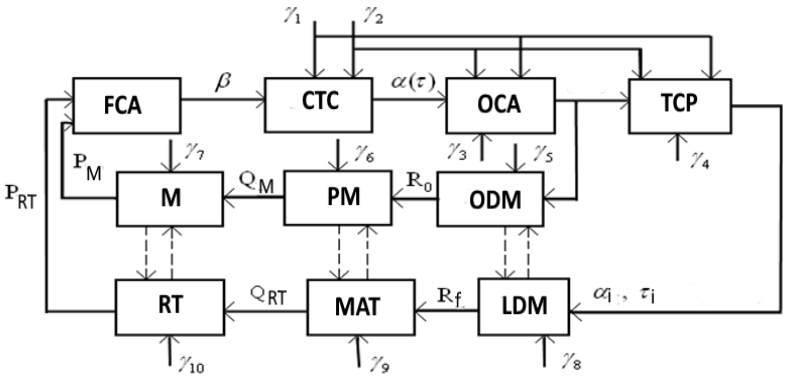

Figure 3.25. Structure of the process of managing the technical condition of the SPP equipment

In Fig. 3.25: $\alpha(t)$ – a set of technical condition indicators; β – a set of control actions; α_i, τ_i – values of condition indicators and operating time at which the residual resource will be exhausted (forecasting result); R_O – operational management decision; R_f – long-term management decision; signals: Q_M – readiness for maintenance; Q_{RT} – readiness for repair and testing; P_M – completion of maintenance activities; P_{RT} – completion of repair work and testing; γ_1 – operating conditions of objects; γ_2 – operating modes of the object; γ_3, γ_4 – regulatory bases of operational assessment; criteria: γ_5 – for making an operational management decision; γ_6 – for preparation for maintenance; γ_7 – for maintenance quality; γ_8 – for making a long-term management decision; γ_9 – for preparation for repair; γ_{10} – for repair and testing quality.

Below is a summary of the key performance metrics of the diagnostic loop obtained from experimental studies (Subsections 3.1–3.2). Table 3.9 demonstrates the effect of integrating DCT into the digital twin streaming platform: comparing autonomous (batch) execution with the option featuring streaming analytics and online adaptation.

Table 3.9

KPIs of diagnostic loop functioning

Indicator	Designation	Autonomous DCT (batch mode)	DCT as part of DT (stream analytics)
Classification accuracy	Accuracy	0.87	0.94
Precision of positive detections	Precision	0.85	0.95
Recall	Recall	0.82	0.93
F1-score	F1	0.83	0.94
Average packet processing time	t_{proc} (s)	125	68
Forecast update delay	Δ_{upd} (s)	5.0	2.1
Stability under input data degradation	R_{stab} (%)	74	91
Mean time to detect	MTTD (s)	4.7	2.6
Mean time to recovery	MTTR (s)	7.8	3.9

As can be seen from the table, integration provides simultaneous improvement in accuracy and reduction of response delays—this is explained by the presence of preprocessing mechanisms, data verification, and continuous feature updating within the DT loop (see Section 3.2). To clearly demonstrate the quantitative advantage of integration, Figure 3.26 graphically compares the key performance indicators (KPIs) of the diagnostic loop, highlighting the percentage change when moving from autonomous (batch) DCT processing to integrated stream analytics within the DT loop. The chart confirms simultaneous improvement in classification quality (Accuracy, F1-score) and critical reduction in temporal delays (Δ_{upd} (s), MTTD).

Analysis of Figure 3.26 confirms the methodological effectiveness of synchronizing the diagnostic model (DCT) with stream analytics and the Digital Twin environment. Critically, the integration yields a simultaneous, systemic improvement across metrics of quality and temporality: classification metrics (Accuracy, F1-score) increase by 8-13%, while time-

critical indicators (t_{proc}, Δ_{upd}(s), MTTD) are reduced by 40-58%. This simultaneous improvement demonstrates that the integrated architecture not only achieves higher diagnostic fidelity but, crucially, transforms the diagnostic loop from a periodic, latency-prone operation into a resilient, near-real-time

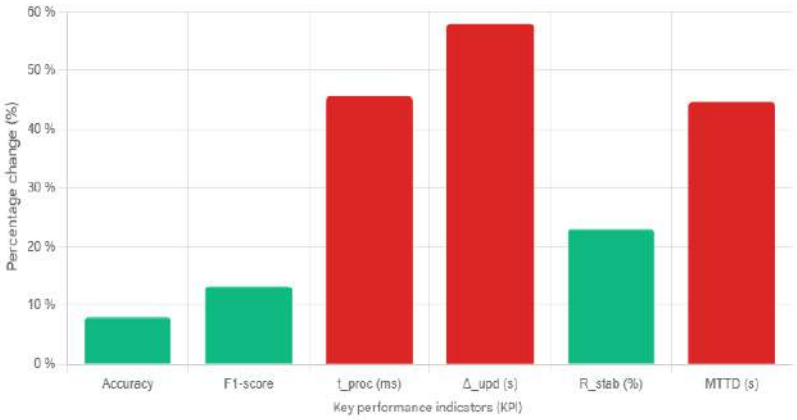

Figure 3.26. Comparison of KPIs: percentage improvement due to DCT integration into the DT Loop

The development of the ideas presented in Subsections 3.1 and 3.2 makes it possible to propose a comprehensive architecture of an information environment that integrates monitoring, diagnostics, and forecasting of the technical condition of CTSs. This architecture is not limited to the use of individual methods (for example, decision trees or fuzzy logic), but forms a methodological foundation that synchronizes:

- data collection and preprocessing;
- construction and updating of diagnostic models;
- implementation of predictive algorithms;
- automation of decision-making processes.

The main feature of the developed approach is the integration of classical diagnostic tools (“fault tree,” probabilistic models, fuzzy logic methods) with new technologies of the DT, stream analytics, and cognitive modeling. This ensures the transition from discrete fault analysis to dynamic monitoring, where prediction and diagnostics are formed in real time.

Unlike traditional diagnostic systems, where data processing was performed in batch mode, the proposed information environment is built on the principles of multi-level architecture:

1. Data layer - sensor nodes, distributed IoT devices recording operating parameters;
2. Preprocessing layer - noise filtering, normalization, data enrichment using the digital twin;
3. Diagnostic layer - DCT models, error classifier, and dynamic fault tree;
4. Predictive layer - algorithms for residual life prediction and critical transition detection;
5. Decision-support layer - automated generation of recommendations (operational and strategic).

For a quantitative illustration of the advantages of integrating the DCT model into the digital twin loop, consider the dynamics of reducing diagnostic data processing time and monitoring result update delay. The baseline scenario uses an autonomous DCT implementation, where data are processed in batches without continuous synchronization with sensor streams.

In the integrated version, stream processing (Kafka → Flink) is applied, providing immediate data transfer to the digital twin.

Figure 3.27 shows the dependencies of two key indicators:

- t_{proc} - average diagnostic query processing time (ms);
- Δ_{upd} - monitoring result update delay (s).

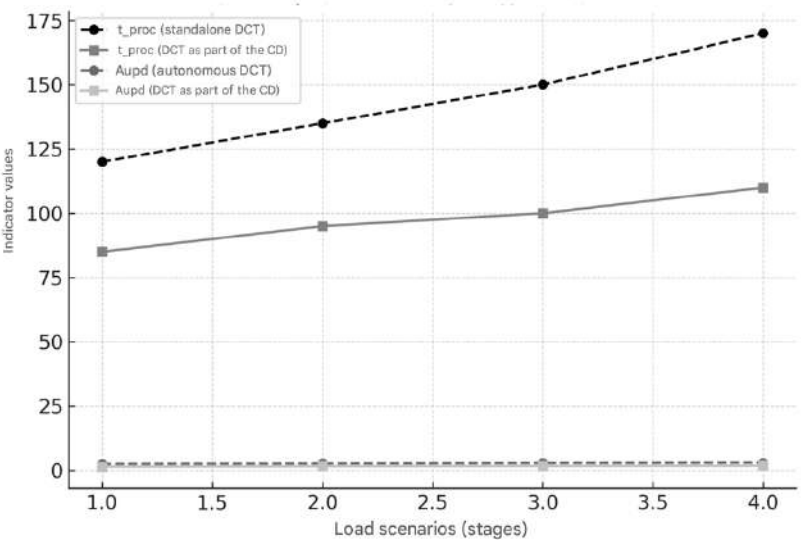

Figure 3.27. Dynamics of processing time (t_{proc}) and update delay (Δ_{upd}) reduction during DCT integration into the digital twin loop

The indicator values are given in relative units (100% — baseline level of autonomous DCT). The abscissa axis shows load scenarios corresponding to increasing data inflow intensity:

1. S1 (nominal mode) - data from 100 sensors with a frequency of 1 Hz;
2. S2 (increased load) - 500 sensors, 2 Hz;
3. S3 (emergency mode) - burst load up to 1000 sensors, 5 Hz;
4. S4 (limit mode) - 1500+ sensors, 10 Hz.

In the autonomous DCT, the processing time grows linearly with increasing load, and the update delay reaches tens of seconds in emergency modes. When embedded in the digital twin, due to distributed stream processing, the growth of these indicators slows down: t_{proc} decreases by 40–60%, and Δ_{upd} - by 3–4 times.

Thus, the transition from an autonomous DCT to integration into the digital twin provides:

- more stable functioning of the diagnostic loop under high loads;
- the ability to operate in modes close to real time;
- a reduced probability of delayed decisions during critical changes in equipment condition.

It is especially important to emphasize that the improvement in update delay (Δ_{upd}) has a systemic effect: residual life prediction and early defect detection become practically continuous processes rather than periodic operations. This enables more accurate planning of maintenance and repair activities.

The dynamics of reducing processing time (t_{proc}) and update delay (Δ_{upd}) during DCT integration into the digital twin loop are shown in Fig. 3.20. As can be seen from the graph, with increasing input stream intensity, the autonomous DCT demonstrates growing delays and processing times, leading to the risk of delayed response. At the same time, integration of the DCT into the digital twin ensures more stable behavior: the values of t_{proc} and Δ_{upd} grow much more slowly due to data preprocessing, stream filtering, and distributed computation mechanisms.

For quantitative illustration, Table 3.10 presents numerical values of the indicators for four load scenarios:

- S1 – baseline mode (100 streams/s);
- S2 – medium mode (500 streams/s);
- S3 – high mode (1000 streams/s);
- S4 – limit mode (2000 streams/s).

In scenarios S3–S4, the integration advantage is particularly pronounced:

- processing time decreases by more than 2 times;
- update delay - almost 2 times compared to the autonomous DCT.

Table 3.10

Influence of DCT integration into the DT loop on processing time and update delay

Load scenario	Load characteristics	Processing time t_{proc}, s (Autonomous DCT)	Processing time t_{proc}, s (DCT as part of DT)	Update delay Δ_{upd}, s (Autonomous DCT)	Update delay Δ_{upd}, s (DCT as part of DT)
S1	Baseline load (100 data streams/s)	50	30	100	60
S2	Medium load (500 streams/s)	95	55	180	110
S3	High load (1000 streams/s)	160	90	280	170
S4	Limit load (2000 streams/s)	240	95	400	180

Thus, the integration of the DCT into the digital twin not only improves forecasting accuracy but also significantly enhances the temporal characteristics of the diagnostic loop, enabling operation under high-load conditions without critical delay growth.

The next stage of methodological support for the information environment was the inclusion of the Error Classifier module, designed for filtering erroneous classifications in dynamics. Its key task is to minimize the influence of noise and incorrect data received from sensors and intermediate algorithms. Unlike standard smoothing procedures (such as moving average or simple time filtering), the Error Classifier operates not only at the data processing level but primarily at the model and semantic level, comparing class probabilities against the digital twin's predictive forecasts and ontological data to identify abnormal deviations. This capability to integrate high-level contextual information is the foundation of its original design. This makes it possible to significantly increase diagnostic reliability and reduce the delay in error detection.

Table 3.11

Performance indicators of the error classifier as part of the digital twin

Indicator	Symbol	Value
Classification accuracy	Precision	0.95
Classification completeness	Recall	0.93
False positive errors	FPR	0.07
Average error detection time	MTTD (s)	2.5
Average error correction time	MTTR (s)	4.1

Analysis of Table 3.11 shows that the Error Classifier enables the system to respond to incorrect data within seconds (MTTD < 3 s, MTTR < 5 s). At the same time, a balanced ratio of Precision and Recall is achieved, minimizing both false-positive and false-negative decisions. Thus, the Error Classifier becomes an important methodological component of the digital twin, ensuring the stability of the entire architecture against noise and sensor data failures.

For visual representation of the operation of the developed Error Classifier block, Fig. 3.28 shows its loop within the information environment of the digital twin. Unlike autonomous solutions, the Error Classifier is integrated directly into the diagnostic data flow:

• data from sensors and IoT nodes enter the verification block, where they undergo primary filtering and comparison with the reference parameters of the digital twin;

• the classification block (DCT model) generates an object state prediction based on preprocessed data;

• the error classifier analyzes the classifier's results, identifies potentially erroneous decisions, and corrects them based on time series, probabilistic features, and historical information;

• the decision support system receives an already verified and refined forecast, which is used for operational and strategic management decisions (see Fig. 3.25).

Thus, the Error Classifier serves as a "dynamic reliability filter" that reduces the likelihood of incorrect decisions and significantly decreases the system's response delay when anomalous data are received.

The structural diagram of the error classifier loop within the DT environment (Figure 3.28) defines a critical methodological component designed to ensure diagnostic robustness against sensor noise and data degradation. Unlike standard data smoothing techniques, the Error Classifier operates at the model and semantic level, integrating contextual information from the DT to validate the classification output.

The information flow within this specialized diagnostic contour is organized as follows:

1. Data Layer & Ontological Layer: raw telemetry enters the preprocessing layer, while the ontological layer / DT simultaneously supplies contextual data and verification rules based on the system's reference model and historical operation trends;

2. Preprocessing Layer: this layer performs data cleaning and initial validation by comparing incoming features with the DT's physical limits, resulting in clean preprocessed features;

3. Diagnostic Model (DCT): The DCT generates classification results (probabilities of states) from the clean features, which are then passed to

two distinct paths: directly to the error classifier (as direct probabilities) and to the DT Feedback Loop;

4. DT Feedback Loop: this module is the core of the validation process. It dynamically compares the current classification probabilities with historical trends and predicted probability changes modeled by the DT. This comparison identifies potential probabilistic anomalies that cannot be explained by normal degradation;

5. Error classifier (ECL): the ECL receives the initial classification, alongside the contextual feedback from the DT Loop. Its unique function is to use time-series analysis and probabilistic features to confirm or reject potentially erroneous classification decisions flagged by the DT feedback. This process minimizes false positives resulting from temporary signal spikes or minor sensor failures;

6. Decision support layer: only after the classification result has been validated and, if necessary, corrected by the error classifier, is the verified state/correction passed to the final decision-making process.

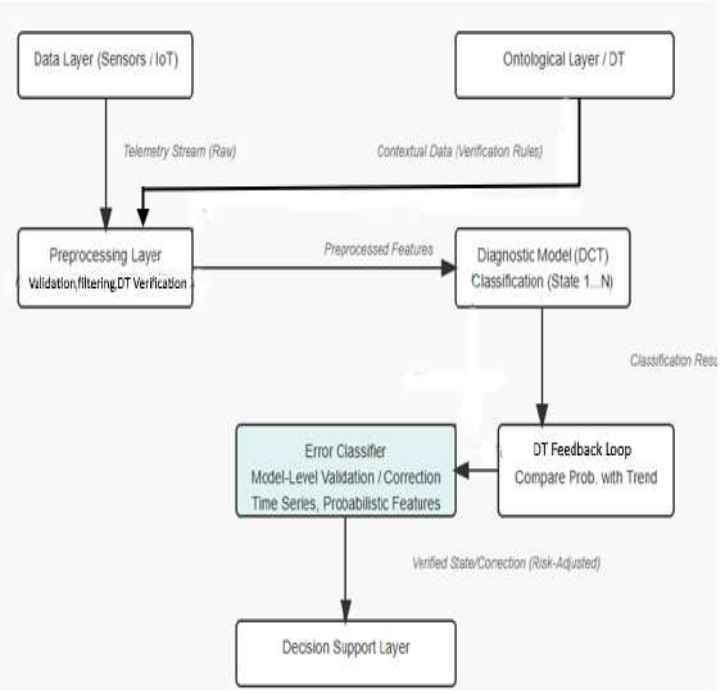

Figure 3.28. Error classifier loop in the information environment of the DT

This integrated approach transforms the Error Classifier into a dynamic reliability filter, ensuring that diagnostic decisions are based not merely on instantaneous sensor readings but on a validated consensus between the analytical model (DCT) and the contextual, historical data provided by the DT.

Practical implementation has shown that including the Error Classifier in the diagnostic loop allows:

- reducing the number of false-positive triggers by almost 20% compared to the standalone DCT;
- achieving an average error detection time (MTTD) of less than 3 seconds;
- increasing forecast stability under input data degradation by 15–18%.

These results confirm the necessity of the error classifier as a mandatory element of the methodological support for the information environment of monitoring, diagnostics, and forecasting of complex technical systems.

An important element of the methodological support is the regulation of the life cycle of diagnostic and prognostic models within the digital twin. Unlike static algorithms used in traditional systems, in the proposed architecture the models operate in the format of services (MaaS Model-as-a-Service). The adoption of the Model-as-a-Service (MaaS) architectural pattern for the diagnostic components introduces a crucial organizational and technological layer to the methodology. MaaS dictates that individual models are encapsulated, versioned, and accessed exclusively via stable APIs. This organizational principle is essential for establishing a continuous Machine Learning Operations (MLOps) cycle for diagnostic models, encompassing automated monitoring, rapid retraining, seamless deployment (A/B testing), and model lifecycle management. This means that each model (DCT, Error Classifier, residual life prediction) undergoes the following sequential stages: development and testing on a training dataset; deployment and integration into the DT; performance monitoring using KPI metrics (Accuracy, Recall, Latency, etc.); retraining on streaming data and parameter updating; versioning and publication of a new iteration. This integration ensures that the diagnostic system remains operationally relevant and continuously adapts to the evolving technical state of the CTS.

As shown in Fig. 3.29, the digital environment provides a continuous model development cycle, where each update is based on monitoring results and real operational data. This process is integrated into the overall information environment, creating a methodological foundation for the stable operation of diagnostic and prognostic modules throughout the entire life cycle of CTSs.

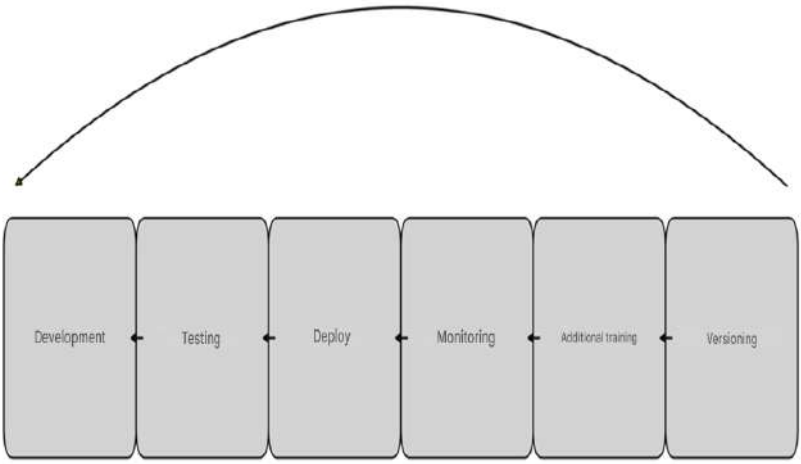

Figure 3.29. Model life cycle in the MaaS paradigm

The implementation of the error classifier module and the integration of DCT models into the digital twin have confirmed that the information environment cannot be viewed as a set of isolated tools but must function as a multi-level system.

This is illustrated, in particular, in Table 3.11, where the transition from a standalone DCT to a DCT integrated into the digital twin results in an increase in classification accuracy (from 0.87 to 0.94) and a reduction in forecast update delay (from 5.0 to 2.1 seconds). Additionally, Fig. 3.26 shows the dynamics of decreasing processing time and latency when the diagnostic loop is included in the digital twin, confirming the effectiveness of streaming analytics.

The performance indicators of the Error Classifier (Table 3.11) demonstrate an improved balance between classification precision and recall and a significant reduction in the system's response time to incorrect data. Finally, the use of a dynamic "fault tree" (Fig. 3.23) has shown the feasibility of automatically linking faults with telemetry, which forms the basis of the ontological level of the digital twin.

Thus, it is the integrated architecture implementing the end-to-end cycle "data collection → preprocessing → diagnostics → forecasting → decision support" that ensures not declarative but quantitatively confirmed improvement of monitoring and diagnostics characteristics for complex technical systems.

The set of presented results confirms the necessity of transitioning from fragmented diagnostic solutions to an integrated architecture that ensures

stable operation under conditions of uncertainty and large data streams. The key principle here is the multi-level structure of the information environment, where each level performs strictly defined functions and passes results to the next stage.

Thus, the data collection level is represented by distributed sensor nodes and IoT devices forming the telemetry stream. At the preprocessing level, noise filtering, normalization, and parameter verification are carried out based on the digital twin. The diagnostic level combines DCT models, the error classifier, and the dynamic "fault tree," enabling real-time defect identification. The prognostic level is responsible for calculating the remaining life, predicting degradation scenarios, and identifying pre-failure transitions. Finally, the decision support level provides automated generation of control actions and recommendations for operators.

This approach allows implementing a closed-loop "physical system ↔ digital model ↔ control action," where the results of diagnostics and forecasting not only reflect the current state but also serve as the basis for proactive management. This is the fundamental difference between the proposed architecture and traditional systems, in which data processing was batch-based and did not allow maintaining the required level of responsiveness.

Below is a diagram of the multi-level information environment that reflects the logic of the structure and interaction of the described levels (Fig. 3.30). The analysis of the developments presented above (DCT models, Error Classifier, prognostic algorithms, dynamic "fault tree") shows that each contributes independently to improving the quality of monitoring and diagnostics of complex technical systems. However, their real effectiveness is manifested only when they are combined into a single information environment.

The results of experimental studies have made it possible to identify the requirements for such an environment: the need for streaming data processing, integration with the digital twin, and the presence of diagnostic, prognostic, and decision support levels. Therefore, the multi-level architecture shown in Fig. 3.30 is not considered an initial theoretical scheme but rather a generalized conclusion of the conducted work. It consolidates specific results and demonstrates their place within a coherent system that closes the full cycle "data collection→preprocessing→ diagnostics→forecasting→decision support."

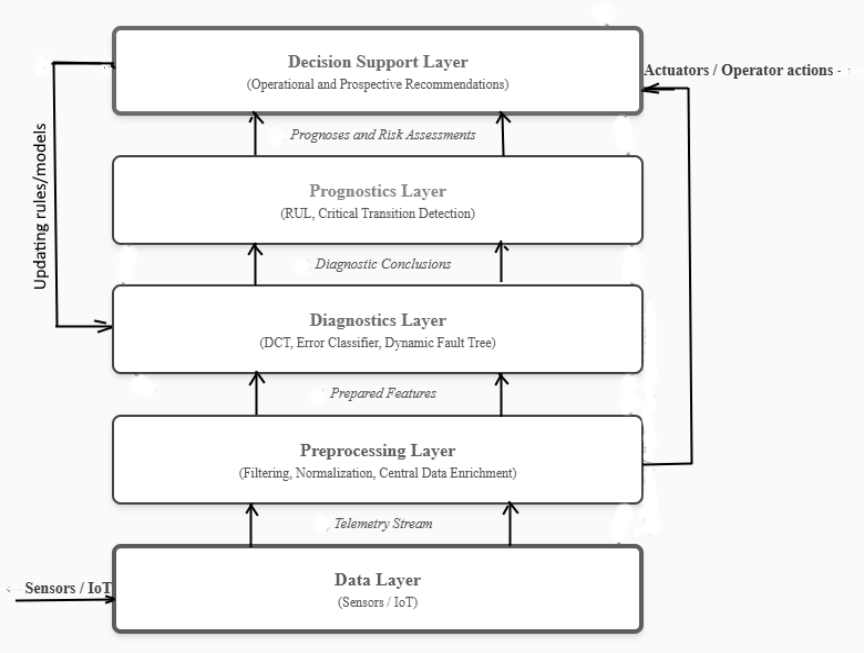

Figure 3.30. Multilevel architecture of the information environment for monitoring and diagnostics of CTSs

Figure 3.30 demonstrates the fundamental organization of the proposed information environment. The key components and their interactions are briefly described below.

1. Data Level (Sensors / IoT). Telemetry, events, and metadata are collected by distributed IoT nodes and sensor complexes. At this level, important issues include synchronization, time-stamping, local preprocessing, and ensuring transmission to the streaming bus (Kafka, MQTT, etc.).

2. Preprocessing level. This level performs data validation and verification (matching measurements with digital twin models), noise and outlier filtering, normalization, and enrichment (feature engineering) using auxiliary data from the digital twin and metadata (ontology descriptions). For streaming operation, engines such as Flink/Streams are used; for batch processing — separate ETL channels.

3. Diagnostic level. This level hosts diagnostic modules: DCT (Decision-Tree-based Classifier), error classifier (module for detection and classification of distortions/anomalies), and the dynamic version of the "fault tree," presented as an ontology. Diagnostic outputs are generated both in

online mode (streaming events) and as results of periodic retraining/validation.

4. Prognostic level. At this level, algorithms are employed for residual useful life (RUL) estimation, detection of pre-failure transitions, and calculation of probabilistic event development scenarios (what-if simulations, Monte Carlo, hybrid physical-statistical models). Within the digital twin, accelerated simulations and scenario calculations can be performed to assess the consequences of control actions.

5. Decision support level. Resulting recommendations (prescriptive actions) are generated considering criteria of cost, risk, downtime, and resource availability and are transmitted to both operators and automated control systems. If an automated mode is selected, the decisions are integrated with controllers/actuators via secure interfaces.

The scheme includes direct "bottom-up" telemetry flows and feedback mechanisms: forecasting results and control decisions are used to update diagnostic rules, revise features, and retrain models (feedback loop). This closes the "target cycle" of the digital twin, allowing the system to adapt to changing operational modes and improve the quality of recommendations over time. Technical notes and implementation recommendations: the streaming loop is recommended to be built on a distributed event bus (Kafka) and a stream processing engine (Flink or Kafka Streams) - see Section 2.3; it is important to maintain offset retention and replay capabilities for model retraining; for preprocessing and diagnostic levels, a microservice architecture (Model-as-a-Service) should be used: individual models must be encapsulated and invoked via API, ensuring versioning and A/B testing; a quality metrics module (KPI) must be provided: LAT (processing latency), t_{proc} (average packet processing time), Δ_{upd} (forecast update delay), R_{stab} (stability under input data degradation), Accuracy/Precision/ Recall/F1 for diagnostics; the ontological layer (link between the "fault tree" and digital twin metadata) should store the CTS topology and the mapping rules between sensor parameters and tree nodes - this is crucial for automated defect localization and estimation of impact on related elements.

Thus, the multilevel architecture of the information environment (Fig. 3.29) integrates all the solutions proposed in this subsection - from DCT and Error Classifier to prognostic modules and the ontological fault tree - into a unified system for monitoring, diagnostics, and forecasting support. This architecture serves as the methodological foundation for transitioning to technical condition assessment models, which will be examined in detail in Chapter 4.

CHAPTER 4. METHODS AND MODELS FOR ASSESSING THE TECHNICAL CONDITION OF COMPLEX TECHNICAL SYSTEMS

4.1 Concept of analysis and risk assessment of failures in a complex technical system

Analysis and assessment of failure risk in CTS under modern conditions require the integration of classical reliability methods with contemporary digitalization approaches. DTs, streaming analytics, and intelligent diagnostic models form the informational foundation for monitoring and forecasting the technical state, expanding the capabilities of traditional methods.

The foundation remains a conceptual model that ensures the formalization of interdependencies between elements, processes, and factors influencing failure risk. It is based on a systemic representation of a CTS as an ensemble of interconnected elements interacting under uncertainty. In this approach, risk is considered as a function of failure probability and potential damage, while the assessment is performed with consideration of both structural and functional relationships between subsystems.

To formalize these interdependencies, a CSM is used, representing the system as a directed graph where the nodes correspond to aggregates, subsystems, and functional states, and the arcs correspond to cause-effect relationships. This approach makes it possible to describe the interaction structure and simulate the propagation of disturbances and failures across the hierarchy of elements.

In the context of digitalized monitoring, diagnostics, and forecasting of technical states of complex systems, the CSM can utilize data from digital platforms and remote control systems to refine the parameters and relationships between elements. However, its primary purpose is to formalize the cause-effect dependencies between failures, impacts, and consequences, enabling scenario-based risk development analysis and forecasting of their temporal dynamics.

The source of input data for the cognitive simulation model is digital monitoring platforms and the digital twin of the complex technical system. The digital twin accumulates diagnostic, prognostic, and residual life estimation results, forming a stream of parameters used to update the nodes and connections of the cognitive model. Thus, the CSM does not duplicate the functions of the DT but uses it as a source of reliable data on the condition of objects and external impacts, thereby ensuring the realism of risk analysis scenarios for CTS equipment failures.

To ensure the practical applicability of the CSM, it is important to establish a connection between the abstract elements of the model and real operational data. Under digitalization, this role is fulfilled by the DT, which

provides the cognitive model with reliable information about the actual technical state of the system. The DT accumulates telemetry streams, results of diagnostic and prognostic models (e.g., DCT, RUL estimates), forming a unified digital representation of the dynamic state of the object. This data is supplied to the cognitive model in real time, where it is used to update cause-effect relationships, revise factor weights, and refine risk indicators. The integration of the DT and CSM thus creates a closed loop of risk analysis: from diagnostics - to consequence assessment and the generation of risk management recommendations.

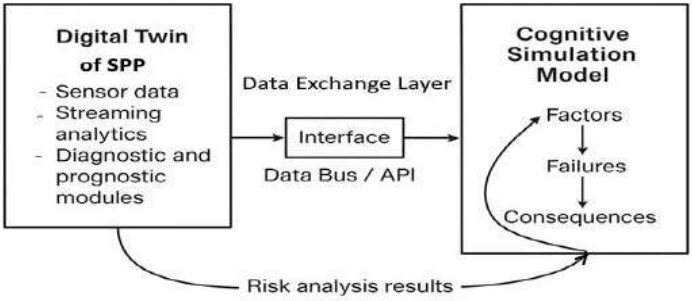

Figure 4.1. Interaction between the DT and the CSM for risk assessment

The presented diagram illustrates the conceptual architecture of interaction between the digital twin and the cognitive model. The DT block generates a stream of operational data reflecting the actual state of the ship's power plant and its subsystems. Through the Data Exchange Layer, data format unification, temporal synchronization, and mapping of parameters into the ontological space of the cognitive model are carried out. In turn, the CSM transforms the incoming data into a system of cause-and-effect relationships, enabling the assessment of the impact of individual factors on failure risk, the identification of critical nodes, and the calculation of integrated resilience indicators. The results of cognitive analysis are returned to the DT in the form of risk metrics and scenario-based assessments, which allow refining maintenance strategies, adjusting model parameters, and improving the predictive reliability of the DT. Thus, a dynamic "data → CSM → feedback" system is formed, ensuring adaptive and continuous risk assessment throughout the lifecycle of the CTS.

The integrated reliability indicator of CTS elements - technical risk - is a combination of the probabilities of occurrence of hazards of a certain class (P_{wi}) and the losses resulting from accidents and undesired events caused

by technical imperfections or violations of the operational rules of technical systems (Y_{wi}).

$$R \equiv [(P_1, Y_1), (P_2, Y_2), ..., (P_{wi}, Y_{wi})] \tag{4.1}$$

The probability assessment of a node (edge) failure in the CTS CSM is determined based on the Bayesian method of CTS analysis [9]

$$P(C|D) = \frac{P(D|C) \cdot P(C)}{P(C)}, \tag{4.2}$$

where $P(C)$ - the prior probability of hypothesis *C* of node (edge) failure;

$P(C|D)$ - the probability of hypothesis *C* given event *D* occurs (posterior probability);

$P(D|C)$ - the probability of event *D* given hypothesis *C* is true;

$P(D)$ -the total probability of occurrence of event *D*.

To evaluate the functional operability and the corresponding damage of a CTS, a normalizing impact (NI) is applied to the nodes and edges of the CTS CSM digraph. The NI is defined in accordance with the Birnbaum criterion [113]. The NI in terms of evaluating the functional operability of the system $F(t)$ and an individual edge (node) $f_{a(v)}(t)$ is defined as follows

$$\Theta b_{a(v)}(i \mid t) = \frac{\partial F(t|)}{\partial f_{a(v)t}(t)} \tag{4.3}$$

The evaluation of the CTS functional operability using NI is carried out in three stages. At the first stage, the operability of the CTS in a fully functional (fault-free) state is assessed. At the second stage, the sequential failure of individual elements and inter-element links (IL) of the CTS CSM is simulated, and the assessment of a partially faulty system is determined for each structural component (SC). At the third stage, quantitative estimates of the functional damages of an element and IL are performed. The quantitative estimate of the functional damage of a CTS element under ***F*** (nominal operability), *Fi* (system operability under failure of the *i*-th element (v_i)) is determined as follows

$$Y_{fV_i} = F - F_i \tag{4.4}$$

Quantitative assessment of the functional damage of the subsystem under F_j – the operability of the system with the *j*-th subsystem being faulty

(a_j)

$$Y_{fA_i} = F - F_j \tag{4.5}$$

For the CTS CSM, the risk assessment of the failure of the affected vertex (edge) of the model is determined as the product of the probability values of failure of the vertex (edge) of the CTS CSM and the corresponding assessment of the functional damage of the affected vertices (edges) of the digraph. The CTS digraph $G(V, A)$ consists of n vertices (nodes) and m arcs (directed edges). The set of graph vertices is V – ($V = \{v_i\}, i = 1, n$). The set of (ordered) pairs of vertices , called arcs (directed edges) of the graph, is A – ($A = \{a_j\} = \{(v_i, w_j)\}, \mathrm{j} = 1, \mathrm{m}$), where vertex v is called the start, and W the end of the arc. The reliability value of the CTS aggregate corresponding to vertex v_i is determined by

$$v_i(t) = P_{v_i}(t \angle T) \tag{4.6}$$

If it is possible to distinguish constructive or functional aggregates of the CTS, the graph vertices correspond to the structural aggregates of the system, and the graph edges correspond to the interconnections (IC). If it is difficult to distinguish aggregates, the graph vertices correspond to the system parameters, and the graph edges represent the cause-and-effect relationships between the parameters. Multiple edges may enter or leave a node. In this case, one speaks of a set of edges incident to a given node of the graph. Each edge is incident to two nodes located at its ends.

The qualitative representation of the state of the CTS aggregates and ICs is expressed by a functional dependency between the states of the aggregates or ICs, as well as by a certain type of loading from external or internal influences.

Assessments in emergency scenarios of structural and functional failure risk of the CTS, taking into account the interconnection and interaction of its elements, include the following stages:

1. Identification of interconnection and interaction of SC within the hierarchy and topology of the CTS, taking into account the used resource of ESI (energy, substance, information);
2. Construction and analysis of the CTS CSM;
3. Assessment of structural and functional damages of the CTS;
4. Assessment of structural and functional risks of the CTS.

Functional of the states of CTS elements

$$v = \left[F_{v.n.}; F_{v.1.}; a_{zi}; a_{zj}; H_m^v(t); K_{V.d.b.}\right], \quad (4.7)$$

where $F_{v.n.}$ - nominal operability of the element;

$F_{v.1.}$ - operability of the element in the case of its partial loss;

a_{zi}, a_{zj} - incoming and outgoing influences for the CTS element;

i, j - ordinal number of the incoming and outgoing influence for the CTS element;

k, m - total number of incoming and outgoing influences for the IL element;

$H_m^v(t)$ - transfer coefficient of the change in the amplitude of the destructive modeling impulse (DMI);

$(H_m^v(t) = \frac{m_{imp_r}^v(t+1)}{m_{imp_k}^v(t)}$, $m_{imp_k}^v(t)$, $m_{imp_k}^v(t=1)$ - value of the DMI amplitude at the moments of time *t* and *t*+1);

$K_{V.d.b.}$ - coefficient of the degree of damage of the element

Functional of the states of the IC of the CTS

$$a_z = [F_{a.n}; F_{a.l}; v; w; H_m^a(t); K_{A.d.b.}], \quad (4.8)$$

where $F_{a.n}$ – nominal operability of the IC;

$F_{a.l}$ - operability of the IC under partial degradation;

z – type of ECI resource;

$H_m^a(t)$ – amplitude variation transmission coefficient IC

$(H_m^a(t) = \frac{m_{imp_r}^a(t+1)}{m_{imp_k}^a(t)}$, $m_{imp_k}^a(t)$, $m_{imp_k}^a(t=1)$ - value of the amplitude of the DMI at moments in time *t* and *t*+1);

$K_{A.d.b.}$ – coefficient of the damage degree of the IC

To assess the functional damage of the CTS, a normalizing influence is applied to the vertices and edges of the directed graph of the IC CTS.

When assessing the structural damage of the CTS, it is assumed that the initial state of an element (CSM) corresponds to 0 if the DMI does not pass through the element (CSM), and to 1 if the IC passes through the element (CSM). Resetting the state values of the elements (IC) before each subsequent iteration of DMI propagation through the IC is performed at.

$1-imp_j(t)$

The influence of the DMI on a vertex (edge) of the directed graph of the CSM CTS at a discrete time moment *t* is defined as

$$1-imp_j(t)=\frac{w_{a(v)j}(t)}{w_{a(v)j}(t-1)\cdot K_{V.d.b.}\cdot K_{A.d.b.}}, \tag{4.9}$$

where $imp_j(t)$ – impulse vector for the edge with index *j*;

$w_{a(v)j}(t)$ $w_{a(v)j}(t-1)$ – weight value of the edge (vertex) at time moments *t, t-1*.

When passing through the IC from v_i vertex to vertex w_j , the impulses imp_j and imp_i are related by the relation

$$imp_j(t+1)=imp_i(t)\cdot H_m^a(t) \tag{4.10}$$

When passing through a vertex from the *i*-th IC to the *j*-th IC, the impulses imp_j and imp_i are related as follows

$$imp_j(t+1)=imp_i(t)\cdot H_m^v(t)\ . \tag{4.11}$$

The completion of IMI propagation (completion of simulation) over the directed graph corresponds to the equality

$$S=\sum_{j=1}^{M} imp_j(t)=0 \tag{4.12}$$

The DMI is generated at a conventionally defined affected vertex (edge), propagates to subsequent vertices (edges), sequentially disabling the interconnected structural components (SC) of the CSM. The degree of damage from the DMI to an element (CSM) of the CTS is determined by the damage degree coefficient of the element (IC).

$$K_{V.d.b.}=\frac{w_v(t+1)}{w_v(t)\cdot(1-m_{imp_k}^v(t))}, \tag{4.13}$$

$$K_{A.d.b.}=\frac{w_a(t+1)}{w_a(t)\cdot(1-m_{imp_k}^a(t))}, \tag{4.14}$$

where $w_v(t)$, $w_a(t)$, $w_v(t+1)$, $w_a(t+1)$- the value of the weight of an element (CSM) at time moment before *t* and *t+1* after the impact of the DMI.

A gradation of damage degree coefficient values by the level of damaging influence of each element on the structure of the CSM is assumed as follows: greater than 0.7 – maximum emergency; from 0.7 to 0.3 – pre-emergency; less than 0.3 – non-emergency.

The values of structural damages resulting from the affected *i*-th vertex, *j*-th edge of the directed graph for the total number of affected vertices (*b*), edges (*c*) of the CSM

$$Y_{s.(V_i).} = \frac{b}{N}, \qquad (4.15)$$

$$Y_{s.(A_j).} = \frac{c}{M},$$

The quantitative assessment of structural damage from an affected vertex (edge) of the CSM CTS is determined by the damage due to loss of connectivity of the CTS topological structures as the ratio of the affected aggregates to the total number of aggregates (ICs) of the CTS under a single aggregate (IC) failure and unobstructed propagation of the CSM through the CTS. The quantitative assessment of functional damage from an affected vertex (edge) of the IC CTS is determined by the damage due to the disruption of the functioning of aggregates (ICs) as the ratio of the CTS operability under its partial degradation by an aggregate (IC) to the nominal operability of the CTS. The failure risk assessment of an affected vertex (edge) of the IC CTS is defined as the product of the probability values of failure of the vertex (edge) of the IC CTS and the corresponding assessments of the structural and functional damages of the affected vertices (edges) of the directed graph.

To assess the functional damage of the CTS during the propagation of the harmful influence (HI) along the directed graph of the IC CTS, a matrix of values is formed, where each vertex and each edge is assigned a numerical value of the HI magnitude. Initially, all HI magnitude values are assumed equal to 1. During the propagation of the HI along the directed graph, they change proportionally to the weighted values of the SC. Based on Birnbaum's criterion, the HI magnitude value for a selected vertex (edge) of the directed graph of the IC CTS is expressed as the product of the HI magnitude value for the preceding vertex (edge) and the weight value of the selected vertex (edge). For a vertex of the directed graph of the IC CTS, into which several edges enter, the HI magnitude values of

$$m_i(t) = \sum_{i=1}^{N} m_j(t), \qquad (4.16)$$

$$m_i(t) = s_i \cdot m_t(t-1) + s_j \cdot m_i(t-1), \quad (4.17)$$

where s_i, s_j - the weight of the i - th vertex and the j -th edge;

$m_i(t)$, $m_i(t-1)$,–the magnitude of the HI passing through a vertex or edge at time moments *t* and *t-1*

Quantitative assessments of functional damages are determined for: a failed element (v_i) as the difference between the nominal operability of the CTS and the operability of the system under failure of element (v_i); a failed IC (a_j) as the difference between the nominal operability of the CTS and the operability of the system under failure of IM (a_j). Structural failure risk of the i - th element and the j - th IC of the CTS is determined as

$$R_{sv_i} = Y_{sv_i} \cdot p_{v_i}(t), \quad (4.18)$$

$$R_{sa_j} = Y_{sa_j} \cdot p_{a_j}(t), \quad (4.19)$$

where $p_{v_i}(t)$, $p_{a_j}(t)$-, are the probabilities of failure of the affected *i*- th element and *j* - th IC of the CTS.

The structural failure risk of all elements and ICs of the CTS is

$$R_s^a = \sum_i^N R_{sv_i} \cdot p_{v_i}(t), \quad (4.20)$$

$$R_s^s = \sum_j^M R_{sa_j} \cdot p_{a_j}(t), \quad (4.21)$$

The failure risk assessment under structural damage from the affected *i*- th vertex of the CSM CTS is

$$R_{sv_i} = k_{sv_i} \cdot p_{v_i}(t), \quad (4.22)$$

where k_{sv_i} is the structural damage assessment from the affected th vertex of the CSM CTS;

$p_{v_i}(t)$ is the probability of failure of the affected *i*-th vertex of the CSM CTS

The failure risk assessment under structural damage from the affected *j*- th edge of the CSM CTS is

$$R_{sa_j} = k_{sa_j} \cdot p_{a_j}(t), \quad (4.23)$$

where k_{sa_j} - is the structural damage assessment from the affected *j* – th edge

of the CSM CTS;

$p_{a_j}(t)$ -is the probability of failure of the affected th edge of the CSM CTS

The functional failure risk of the *i*-th element and *j*-th IC of the CTS is

$$R_{fa_j} = Y_{fa_j} \cdot p_{a_j}(t), \quad R_{fv_i} = Y_{fv_i} \cdot p_{v_i}(t) \tag{4.24}$$

The functional failure risk of all elements and ICs is

$$R^a_{F\,fa_j} = \sum_i^N R_{fv_i} \cdot p_{v_i}(t), \; R^S_{F\,fa_j} = \sum_j^M R_{fa_j} \cdot p_{a_j}(t) \tag{4.25}$$

The failure risk assessment under functional damage from the affected *i*-th vertex of the CSM CTS is

$$R_{fv_i} = k_{fv_i} \cdot p_{v_i}(t), \tag{4.26}$$

where k_{fvi} is the functional damage assessment from the affected *i*- th vertex of the CSM CTS.

The failure risk coefficient under functional damage from the affected *j* - th edge of the CSM CTS is

$$R_{fa_j} = k_{fa_j} \cdot p_{a_j}(t), \tag{4.27}$$

where k_{faj} is the functional damage assessment from the affected *j* - th edge of the CSM CTS

The probability of failure of elements and ICs is determined as

$$P_{v_i} = \frac{n_{v_i}}{\tau}, \; P_{a_j} = \frac{n_{a_j}}{\tau}, \tag{4.28}$$

where P_{v_i} is the probability of failure of the *i*-th element of the CTS;

P_{a_i} is the probability of failure of the *j*- th IC of the CTS;

n_{v_i} is the number of failures of the *i*-th element of the CTS;

n_{a_i} is the number of failures of the *j*-th IC of the CTS;

$\tau = h$ - the duration of statistical testing

As a basis for determining the values of failure probabilities of elements and ICs of the CTS, the data from [92] may be used. The dependencies of the total structural and functional failure risks on the probabilities of complete and partial failures of all elements and ICs of the CTS servicing systems are determined by the total values of failure risks and the total probabilities of failure of elements and ICs of the CTS as

$$P_v = \frac{\sum_{i}^{N} P_{v_i}}{N}, \quad P_A = \frac{\sum_{j}^{M} P_{a_j}}{M}, \tag{4.29}$$

where P_v is the sum of failure probability values of the CTS elements;

P_A is the total failure probability value of the CSM ICs;

N is the number of CTS elements;

M is the number of CTS ICs

To rank the obtained assessment of structural and functional failure risks of the CTS in emergency scenarios, the generalized Harrington desirability function is used to evaluate the failure risk level:

0–0.2 – minimal, the consequences of the accident are minimal and do not have a significant impact on the operation of the CTS;

0.2–0.37 – acceptable, the consequences of the accident are insignificant, allowing the CTS to operate without repair;

0.37–0.63 – maximal, the consequences of the accident allow CTS operation after repair works are performed;

0.63–1 – critical, the consequences do not allow the CTS to be operated.

The study of the effects of the DMI and HB on the CSM CTS was carried out in a developed software package based on the cross-platform Python language. To represent the input data of the elements and ICs when modeling emergency scenarios in the CSM CTS, the JSON format was used. The visualization of the graphs was performed using the Graphviz software product. MS Office and OpenOffice were used to analyze the research results. To test the method for assessing the structural and functional failure risks of the CTS in emergency scenarios, a system consisting of 12 elements and 26 ICs (Fig. 4.2) was investigated as an example, 17 of which are energy carriers (indicated by solid lines), 9 are matter carriers (indicated by dashed lines). The modeling process for assessing the functional failure risks of the CTS elements and IMs was carried out in accordance with the algorithm shown in Fig. 4.3.

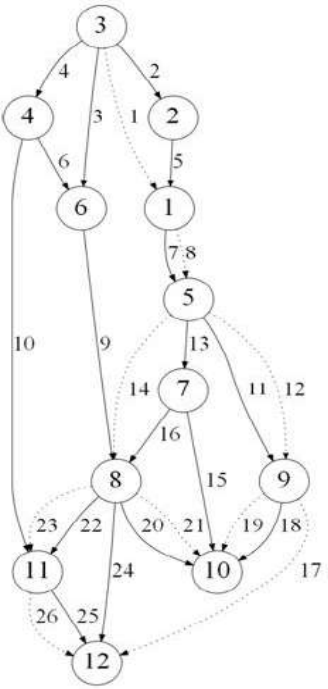

Figure 4.2. Orgaph of the CSM CTS

The results of the calculated values of the structural and functional risk assessment for the vertices and edges of the CSM CTS are presented in Figs. 4.4–4.7. The studies confirmed that if the point of application of the DMI and NR does not belong to the strongly connected elements of the CTS, then the propagation of the DMI and NR is less associated with the risk of CTS failures. In the studied CTS, the strongly connected components are elements 3, 8 and IMs 2, 5, on the operability of which the risk of adjacent elements and IMs depends.

When IC 2 is affected, there is a risk of failures during the operation of elements 1, 5, 7–12, and when element 3 is affected, there is a risk of failures for all structural components of the CTS.

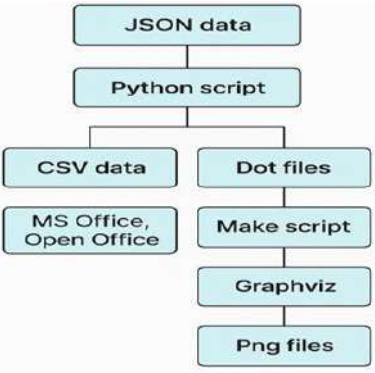

Figure 4.3. Algorithm of the CTS operating conditions simulation process

It has been established that the elements 2, 3, 5, 8 and IMs 2, 6, 9, 12, 16 have the greatest influence on the risk of CTS failures. To a lesser extent - the risk of failures during the operation of elements 1, 7, 9, 11 and ICs 4, 5, 7, 18, 24. The most critical elements are 3, 5 and 8, and IMs 9 and 16.

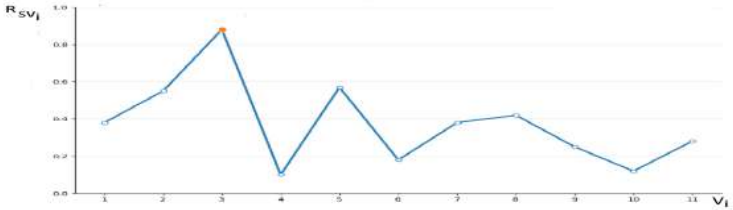

Figure 4.4. Calculated value of structural risk R_{sv_i} for the CTS elements v_i

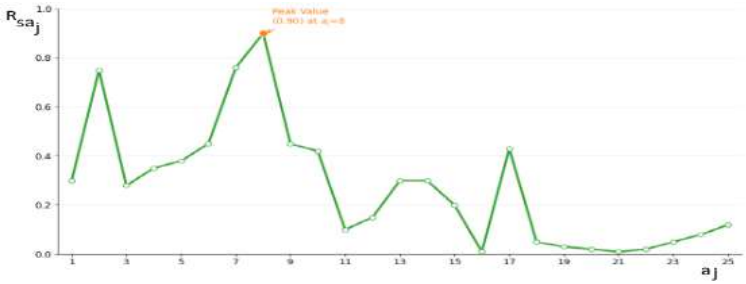

Figure 4.5. Calculated value of risk R_{sa_j} for the CTS inter-element a_j connections

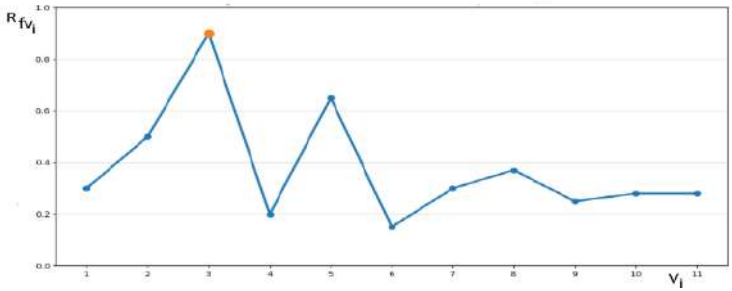

Figure 4.6. Calculated value of functional R_{fv_i} risk for the CTS v_i elements

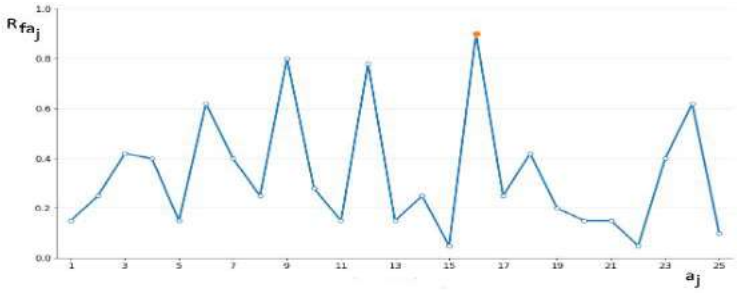

Figure 4.7. Calculated value of functional risk R_{fa_j} for the CTS a_j inter-element connections

The developed method, validated through cognitive simulation modeling, makes it possible to identify the interrelated and interacting elements, inter-element connections, as well as the degree of their influence on the assessment of the structural and functional risk of CTS failures, which in turn enables the automation of decision-making processes in emergency scenarios during CTS operation. The identified interdependencies and interactions of structural components within the hierarchy and topology of the CTS make it possible to determine the magnitude of the damaging influence of each system component in emergency scenarios on the structure of the CTS.

The method's procedures are easily formalized and transformed into a computational algorithm and model for the assessment of structural and functional risks, which is important for CTS with a large number of elements and inter-element connections.

4.2. Methodology for assessing the survivability of complex technical systems

The set of functional complexes of technical systems (FCTS), subsystems, and technical equipment that affect safe operation - for example, of ships - largely determines their survivability. The unpredictability of changes in most factors influencing the properties of the FCTS and their topology gives special importance to the assessment of survivability with consideration of the state of system elements and the interconnections operating between them. At present, in the theory of survivability of FCTS, there is no established methodological approach that makes it possible to solve the problem of a comprehensive assessment of the survivability of such systems from the standpoint of their structural vulnerability and functionality, taking into account the significance of existing intersystem and inter-element interconnections. Thus, methodological support for the assessment and assurance of the survivability of ship CTS in order to increase the level of safe ship operation and reduce accident rates is a relevant scientific and technical task.

The survivability of CTS is an integral concept [114, 115, 116], which implies consideration of the structure of the indicator of successful functioning of the system according to its purpose, consisting of a set of parameters with the following characteristics: the degree of system readiness to accomplish the assigned tasks; the internal topology and interaction of system elements; the context of system usage and of its individual elements both collectively and individually.

In the assessment of CTS survivability, an important part is constituted not only and not so much by local indicators, but by integral indicators and indicators that reflect the functioning of an individual element in the context of the entire system. An integrated assessment of the survivability of the FCTS requires a model that unifies the complexes of separate CTS into a single model reflecting their structural–functional interrelation and interaction. This makes it possible to obtain both an integral assessment of the overall vulnerability of the system as a whole to damaging factors and adverse impacts leading to failure and/or loss of operability of its individual elements, and an assessment of individual elements from the standpoint of their importance and criticality for the functioning of the entire system as a whole [117].

For constructing a cognitive map, the local indicators of survivability, operability, and reliability obtained according to the FCTS operational regulations are formalized and brought to a unified systemic non-scale and dimensionless evaluation. The criteria characterizing the survivability property of the system as a whole and of its individual components, in

accordance with the terminology and definitions used in the methodology, are taken as follows: greater than 0.67 - satisfactory survivability; in the range from 0.67 to 0.23 - the system is conditionally effective; below 0.23 - the system is inoperable. Similarly, criteria can be defined for individual system components at the level of subsystems, individual units, and elements of ITS.

The propagation of the DMI is modeled based on the following relationships. Let a directed graph be given $G = \{V, E\}$

$$V = v_i, v_i \in V, i = 1,2,\ldots,k, \qquad (4.30)$$
$$E = e_i, e_i \in V, i = 1,2,\ldots,k,$$

where *V* is the set of vertices of the system;

E is the set of arcs of the system, with v_i denoting the vertex with index *i*, v_j denoting the vertex with index *j*, and e_{ij} being the edge of the graph directed from vertex v_i to vertex v_j

In this directed graph, the vertices represent various components (complexes, systems, subsystems, units), and the directed arcs represent their interaction according to one of the types of interconnection (transfer of matter, information, or energy).

The impulsive impact of the DMI is defined by an impulse vector *imp* of the form $imp_j(t), j \in 1,2,..,k$ for discrete time t = 0, 1, 2, 3..., given by the relation

$$1 - imp_j(t) = w_j(t) / w_j(t-1) \qquad (4.31)$$

This relation defines the change in the weights of the directed graph, thereby determining the dynamics of the propagation of external influences throughout the system.

For an impact of *imp = 0*, the element is not affected at all, while for an impact with strength *imp = 1*, the element is rendered inoperable with 100% probability.

Thus

$$w_j(t) = (1 - imp_j(t)) w_j(t-1) \qquad (4.32)$$

During propagation through the system, the impulse is attenuated as it passes along the arcs. It is assumed that the impulse traverses an arc within one discrete time t period; therefore, when passing from element v_i to v_j, the impulses imp_i and imp_j will be related by the following expression

$$imp_j(t+1) = imp_i(t)e_{ij} \tag{4.33}$$

Propagation of the DMI, similar to the DMI, is defined using an impulse vector of the form: $imp_j(t), j \in 1,2,..,k$ for discrete time t = 0, 1, 2, 3... The weight of node *j* is

$$v_j = imp(t)/imp(t-1) \tag{4.34}$$

Thus,

$$imp(t) = imp(t-1)v_j \tag{4.35}$$

During propagation through the system, the impulse is attenuated as it passes through the nodes of the directed graph. The connections between nodes are assumed to be unitary, i.e., an impulse propagating along an edge passes through the arc within one discrete time t period without any attenuation. In the case when more than one impulse arrives at a node simultaneously, their values are summed.

Let us consider the propagation of the diagnostic impulse imp through sequentially connected elements with operability v_i and v_j at time moments *t* (before node v_i), *t*+1 (after node v_i), and *t*+2 (after passing through nodes v_i and v_j). After passing through the first node

$$\begin{gathered} imp(t+1) = imp(t)v_i, \\ imp(t+2) = imp(t+1)v_j = imp(t)v_i v_j \end{gathered} \tag{4.36}$$

For a unit diagnostic impulse

$$\begin{gathered} imp_1(t) = 1, \\ imp_1(t+2) = imp_1(t)v_i v_j = v_i v_j \end{gathered} \tag{4.37}$$

Let us consider the propagation of the diagnostic impulse imp through parallelly connected elements with operability v_i and v_j at time moments *t* (before passing through nodes v_i and v_j), *t*+1 (after passing through nodes v_i and v_j), and *t*+2 (after passing through the point of edge junction in the graph). In this case, both nodes receive the same impulse of magnitude imp(t), which when passing through the nodes splits into two impulses *imp*

$$imp_i(t+1) = imp(t)v_i, \quad imp_j(t+1) = imp(t)v_j \tag{4.38}$$

At the point of edge junction, we obtain the sum of the impulses imp_i and imp_j

$$imp(t+2) = imp_i(t+1) + imp_j(t+1) = imp(t)v_i + imp(t)v_j = (v_i + v_j)imp(t) \tag{4.39}$$

For a unit diagnostic impulse

$$imp_1(t) = 1, \quad imp_1(t+2) = (v_i + v_j)imp_1(t) = v_i v_j \tag{4.40}$$

In a similar way, the relationships for more complex series–parallel structures can be obtained.

Let us now consider the impulse impact on a system whose elements have survivability and connections equal to one, using a DMI with the impulse vector *imp* of the form

$$Imp = (imp_1 = 0, imp_2 = 0, \ldots, imp_i = 1, \ldots, imp_n = 0) \tag{4.41}$$

Such an impact models the damage of object number i by an impulse of strength 1. In this system, such an impact is equivalent to the complete destruction of element and its failure. In a unit system affected by a unit impulse, the connections between the nodes of the directed graph are also unitary, therefore the impulse will propagate through the system until it disables all accessible elements. This represents the “worst-case scenario” for a single damaging impact.

Consequently, the criticality of an element and the vulnerability of the system can be assessed by the degree of damage caused by its failure to the entire system. As an example of such an impact, let us consider a directed graph consisting of 17 elements (Fig. 4.8)

$<V> = V_i$, $i = 1,\ldots,17$. We sequentially apply a unit impulse to each of its vertices and trace the propagation of the impulse through the directed graph.

The structural threat coefficient (STC) for a given node is the ratio of the system’s affected elements to all elements of the system under the condition of a unit impact on this element and the unobstructed propagation of the impulse throughout the system.

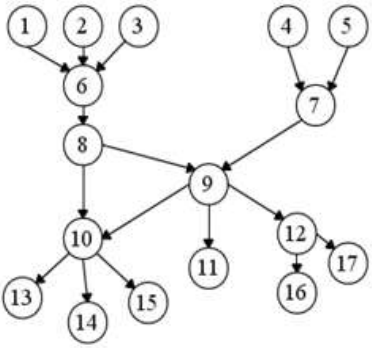

Figure 4.8. Directed graph of 17 elements

The structural threat coefficient (STC) for a given node is the ratio of the system's affected elements to all elements of the system under the condition of a unit impact on this element and the unobstructed propagation of the impulse throughout the system. Thus, if after the failure of element V_j the propagating impulse affects i elements, and the total number of elements in the graph is N, we obtain the following expression for the threat coefficient k_s:

$$k_s(V_j) = \frac{i}{N} \tag{4.42}$$

Let us consider the scenario of a unit impulse affecting the system's connections (assuming the nodes have unit conductivity). Similar to the previous case, the impulse will propagate through the system until it disables all accessible elements.

The criticality of an edge (and, correspondingly, the system's vulnerability) can be assessed by the extent of damage its failure inflicts on the entire system. Let us consider a directed graph of 17 elements (Fig. 4.9), for which $<V> = V_i$, $i = 1,...,17$, $<E> = e_{ij}$, $i = 1,...,17$, $j = 1,...,17$.

We sequentially apply a unit impulse to each of its edges and trace the propagation of the impulse through the graph. For example, Fig. 4.9 shows the final phase of impulse propagation under the failure of connection $E_{8,10}$ (the failed graph connections are indicated by dashed lines). The STC for this connection is the ratio of the system's failed connections to all system connections under a unit failure of this element and unobstructed impulse propagation throughout the system.

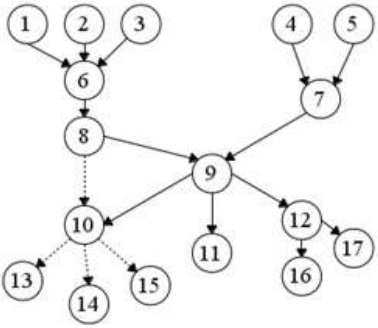

Figure 4.9. Failure of connection $E_{8,10}$ in the directed graph

If after the failure of element $E_{i,j}$ the propagating impulse disables i_e connections, and the total number of connections in the graph is N_e, we obtain the following expression for the threat coefficient k_{se}.

$$k_{se}\left(E_{ij}\right) = \frac{i_e}{N_e} \tag{4.43}$$

Structural threat estimates for nodes and connections make it possible, in a first approximation, to obtain a weighted assessment of the significance of a node or connection for ensuring the survivability of the system as a whole. Numerically, this assessment is greater the more vulnerable the element is to the system. In terms of survivability assurance, the assessment may be expressed with the following gradations:

above 0.7 - the object is structurally critical for the system and requires the highest priority in technical monitoring and diagnostics, and if its operational parameters decrease, it must be replaced first;

from 0.7 to 0.3 - the object is structurally significant for the system but has lower priority regarding survivability assurance;

0.3 and below - the object is not structurally important for the system as a whole, and its failure has minimal impact on the system's survivability.

Thus, the STC reflects the first level of system threat assessment, enabling the ranking of elements by structural significance and the identification of the most threatened ones.

The concept of functional vulnerability of the system is closely related to the concept of operability. In the general case, the operability assessment of the system F expresses the probability that the system will perform the assigned task within a given time Δt.

$$F_i = \frac{P_1^t}{P_0^t} = \frac{P(S_i \mid \Delta t)}{P(S_0 \mid \Delta t)} \tag{4.44}$$

Consider a basic operability CSM in the form of a directed graph consisting of two sequentially connected elements V_i and V_j, each possessing operability f_i and f_j. The total operability of the system, as a probabilistic characteristic, is defined by the following relations

$$F_{ij} = \frac{P_{1ij}^t}{P_{0ij}^t} = \frac{P_{1i}^t}{P_{0i}^t}\frac{P_{1j}^t}{P_{0j}^t} = f_i f_j, \tag{4.45}$$

$$F_{ij} = f_i f_j$$

Consider a directed graph consisting of two parallel-connected elements V_i and V_j, each of which individually possesses operability f_i and f_j, respectively. The total operability of such a system, as a probabilistic characteristic, is

$$F_{ij} = \frac{P_{1ij}^t}{P_{0ij}^t} = \frac{P_{1i}^t}{P_{0i}^t} + \frac{P_{1j}^t}{P_{0j}^t} = f_i + f_j \tag{4.46}$$

From the obtained relations, it follows that the change of a unit diagnostic impulse when passing through the nodes of the directed graph is numerically equal to the total operability of the system. Consequently, by simulating the passage of a unit diagnostic impulse through the system, its operability can be evaluated according to established criteria. The passage of a unit impulse through the system allows the operability of the directed graph to be assessed provided that it is “two-terminal” - it has one input node (source), one output node (receiver), and is loop-free (it contains no feedback loops). If feedback loops are present in the cognitive–simulation model, a condensation procedure described in [40] is applied, ensuring that the directed graph is modeled as loop-free and acyclic.

If the CSM contains multiple sources and receivers, the notions of a “supersource” and a “superreceiver” - special types of nodes - are introduced to simplify the algorithmic procedure. The “supersource” has unit operability ($f_{super} = 1$) and is connected via outgoing links to all sources. The “superreceiver” has unit operability and is connected via incoming links to all receivers. When a unit diagnostic impulse passes through the directed graph extended to include a “supersource”, a “superreceiver”, and a fully operational node v_i ($f_i = 1$), the change in impulse along the path from the “supersource” to the “superreceiver” can be obtained. This allows the functional operability of the system F_1 to be evaluated with node v_i fully

operational. By disabling the node ($f_i = 0$) and simulating the passage of the unit diagnostic impulse, we obtain the assessment of the system's functional operability F_0 with node v_i failed. Taking F_1 as the system state with a fully operational element f_i and F_0 as the system state with a completely failed element, we obtain for node v_i

$$\Theta b\langle i|t\rangle = \frac{\partial F(t)}{\partial f_i(t)} = \frac{F_1}{F_0} \tag{4.47}$$

Sequentially performing similar studies for each of the graph's nodes allows determining the values of the criteria for all nodes of the system. For example, by modifying the directed graph in Fig. 4.9, we obtain the scheme shown in Fig. 4.10. Node S_1 functions as a "supersource", and node S_2 as a "superreceiver". The passage of the diagnostic impulse between the nodes provides the overall functional assessment of the system - its operability.

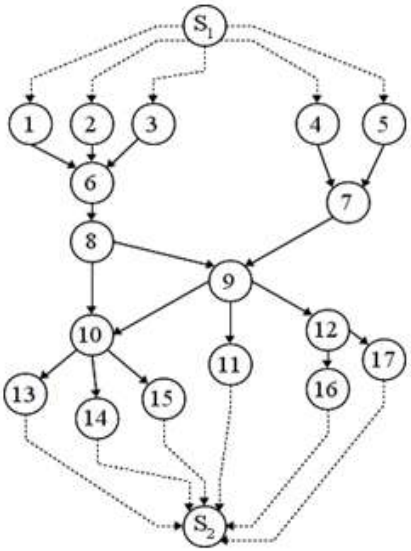

Figure 4.10. Directed graph with a "supersource" and a "superreceiver"

The proposed methodology makes it possible to: obtain a comprehensive assessment of the survivability of technical systems in terms of their functionality and structural vulnerability; determine the numerical

assessment of threats and failure risks for ranking vulnerability priorities; determine the significance of the interrelations existing in the system; perform simulation modeling of the propagation of external impacts through the system structure using a cognitive map.

The main objective of the developed method is to assess the influence of the state of an individual IC on the overall functional and structural survivability of the CTS, i.e., the assessment of the functional and structural threats of the IC. By analogy with the structural and functional threats of CTS aggregates, we use for assessing the degree of threats the coefficients of structural and functional threat, numerically expressing the degree of these threats on the interval (0–1). For the value of the coefficient "0", the state of the IC does not affect the survivability of the CTS, and for the value "1", the survivability of the CTS is determined by the state of the IC.

Let us consider a directed graph (*V, E*) modeling an CTS consisting of a set of nodes and the arcs connecting them.

$$V = (v_i),\ i = 1,...,n$$
$$E = (e_{ij}),\ i = 1,...,n,\ j = 1,...,n \qquad (4.48)$$

where V – the set of system nodes;

E – the set of directed arcs of the system;

v_i – the node with index i;

e_{ij} – the graph edge directed from node v_i to node v_j;

n – the number of nodes in the directed graph

The nodes (vertices) of the directed graph model the states of the aggregates, and the weight of each node v_i corresponds to its functional state. The directed arcs (edges) of the directed graph model the IC, and the weight of each arc v_{ij} corresponds to its functional state, while the direction of the arc is indicated by the order of the indices i and j. Thus, the arc eij is directed from the i-th node v_i to the j-th node v_j.

In the simplest case, when only discrete states of the nodes and arcs are considered (according to a binary scale of values "operational" - "non-operational"), at the discrete time moment t = 0, all weight values of the nodes and arcs are equal to one.

$$v_i = 1,\ i = 1,...,n$$
$$e_{ij} = 1,\ i = 1,...,n,\ j = 1,...,n \qquad (4.49)$$

For the assessment of structural threat and the calculation of the structural threat coefficient, the Modeling Impact Pulse (MIP) method is used, which is applied to solve the problem of evaluating the structural

threats of the IC. The MIP models the impact on an individual edge and is defined by the vector

$$I(t) = (imp_{ij}(t)), i = 1,\ldots,n,\ j = 1,\ldots,n, t = t_1..t_d, \tag{4.50}$$

where $imp_{ij}(t)$ is the set of DMI module values corresponding to the directed arcs of the digraph;

t is the discrete time moment;

t_1 is the initial discrete time moment;

t_d is the final discrete time moment

At each discrete time moment t, the vector of DMI modules $I(t)$ is defined in such a way that for each arc eij at the discrete time moment t_z, there exists $imp_{ij}(t_z)$ equal to zero if at the moment t_z the impulse does not propagate through the arc, and equal to 1 if the impulse affects the arc.

At the initial stage of the study, the affected arc eab is selected (the impact on the IC between the aggregates with indices *a* and *b* is modeled by some external factor). Then, a vector $I(t)$ is formed, in which at the discrete time moment $t = 1$ the DMI modules have the following values:

$$imp_{ij}(t = 1) = 1,\ fori = a,\ j = b, \tag{4.51}$$
$$imp_{ij}(t = 1) = 0,\ fori \neq a,\ j \neq b$$

The DMI sequentially propagates through the orgraph, disabling adjacent nodes and arcs at each subsequent step of discrete time. In this process, the arc and the node change their value from 1 ("operational") to 0 ("non-operational"), and the values of the modules of the vector $I(t)$ reflect the passage of the DMI through the nodes at each discrete moment in time, along the directed edges of the graph. The impulse impact of the DMI on an arc of the orgraph is determined by the following recurrent relation

$$\frac{e_{ij}(t_n)}{e_{ij}(t_{n-1})} = 1 - imp_{ij}(t_n), \quad i = 1,2,...,n \quad j = 1,2,...,n \quad t = 0,1,2,3... \tag{4.52}$$

where $e_{ij}(t)$ is the value of the weight (transfer coefficient) of the arc e_{ij} at the discrete moment of time t.

When passing through the arc eij, the module values of the DMI will be related by the following relation for two discrete moments of time t and $t+1$

$$imp_{ij}(t+1) = imp_{ij}(t)e_{ij}(t) \tag{4.53}$$

Thus, as the impulse propagates through the system, it is attenuated depending on the weights of the arcs of the CTS orgraph. In the considered case, the weights of the arcs take binary values (0 or 1) depending on the operability of the arc. In order to model intermediate states of partial operability of the IC, the DMI algorithm provides for the use of a wide range of weights.

After passing through an arc, the corresponding value of the DMI module is set to zero - as having been expended on this arc. The criterion for the termination of the DMI action is a zero value of the sum of the modules *I(t)* at the moment of time t_d

$$\sum_{i=1}^{n}\sum_{j=1}^{n} imp_{ij}(t_d) = 0 \tag{4.54}$$

The structural threat coefficient for links (STC) for a given IC is calculated as the ratio of the damaged links of the system to all links of the system, when the considered IC is damaged and the impulse propagates freely through the system.

Let, after the damage of the IC e_{ij}, the propagating impulse damages is links, and the total number of links in the graph is N_s, then we obtain the following expression for the threat coefficient k_{ss}:

$$k_{ss}(e_{ij}) = \frac{i_s}{N_s} \tag{4.55}$$

where i_s is the number of links damaged by the DMI;

N_s is the total number of links in the graph;

$k_{ss}(e_{ij})$ is the structural threat coefficient for links for the node e_{ij}

The structural threat coefficient for aggregates (STA) for a given IC is calculated as the ratio of the damaged aggregates of the system to all aggregates of the system, when the considered IC is damaged and the impulse propagates freely through the system.

Let, after the damage of the IC eij, the propagating impulse damages ia nodes modeling the aggregates of the CTS, and the total number of nodes in the graph is N_a. As a result, we obtain the following expression for the threat coefficient k_{sa}

$$k_{sa}(e_{ij}) = \frac{i_a}{N_a}, \tag{4.56}$$

where i_a is the number of aggregates damaged by the DMI;

N_a is the total number of aggregates in the graph;

$k_{sa}(e_{ij})$ is the structural threat coefficient for aggregates for the node e_{ij}

Let us consider the application of the developed method using the example of an CTS consisting of 21 aggregates connected by complex inter-aggregate links of a unified nature. In Fig. 4.11, an orgraph of the generalized CTS is shown, and in Fig. 4.12 and 4.13 - the final stages of link damage between nodes 1 and 5 and 2 and 6, respectively. The failed links of the graph are indicated by dashed lines. In the case of the damage of the link between aggregates 1 and 5, the number of damaged links is $i_s = 5$, and the number of damaged aggregates is $i_a = 5$. In the case of the damage of the link between aggregates 2 and 6, the number of damaged links and aggregates is $i_s = 5$ and $i_a = 5$, respectively.

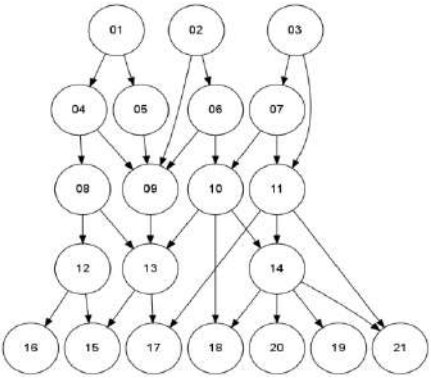

Figure 4.11. Orgraph of the generalized CTS

The total number of links and aggregates in the system is Ns = 17 and Na = 21. As a result, the values of STC and STA for the first and second cases

$$k_{ss}\left(e_{1,5}\right) = \frac{i_s}{N_s} = \frac{5}{17} \approx 0.2941$$

$$k_{sa}\left(e_{1,5}\right) = \frac{i_a}{N_a} = \frac{5}{21} \approx 0.2381$$

$$k_{ss}\left(e_{2,6}\right) = \frac{i_s}{N_s} = \frac{12}{17} \approx 0.7059$$

$$k_{sa}\left(e_{2,6}\right) = \frac{i_a}{N_a} = \frac{11}{21} \approx 0.5238$$

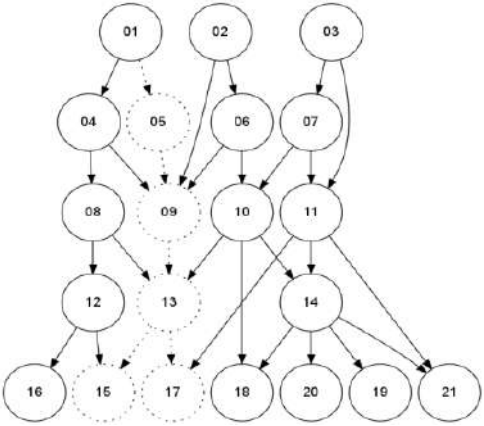

Figure 4.12. Final stage of damage to the IC between aggregates 1 and 5

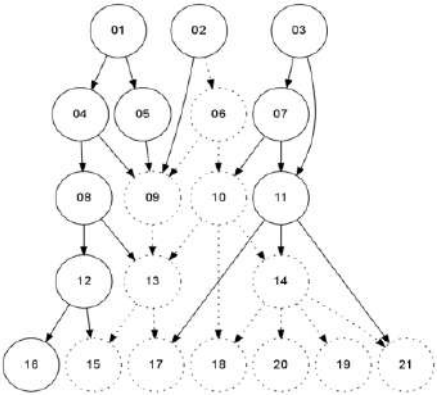

Figure 4.13. Final stage of damage to the IC between aggregates 2 and 6

From the given estimates, it follows that, despite the same level in the hierarchy of the CTS, the link between aggregates 2 and 6 is almost twice as high in priority as the link between aggregates 1 and 5.

To implement the DMI methodology for assessing the structural threats of IC, an algorithm for the propagation of DMI through the links of the CTS was developed (Fig. 4.14).

The representation of the modeled graph data is carried out using a JSON file, in which the initial parameters of the modeled system are specified. In

particular, the definition of the types of nodes and links for the CTS looks as follows

```
{
...
"v":
    {
    "01":{"04":1, "05":1},
    "02":{"06":1, "09":1},
    "03":{"07":1, "11":1},
    "04":{"08":1, "09":1},
    "05":{"09":1},
    "06":{"09":1, "10":1},
    "07":{"10":1, "11":1},
    "08":{"12":1, "13":1},
    "09":{"13":1},
    "10":{"13":1, "14":1, "18":1},
    "11":{"14":1, "21":1, "17":1},
    "12":{"15":1, "16":1},
    "13":{"15":1, "17":1},
    "14":{"18":1, "19":1, "20":1, "21":1},
    "15":{},
    "16":{},
    "17":{},
    "18":{},
    "19":{},
    "20":{},
    "21":{}
      },
...
}
```

In addition to the CTS nodes and the links between them, the file specifies the initial data, the names and weights of the nodes, the name of the modeled system, and other service parameters. This allows the CTS model to be flexibly modified without changing the main source code of the program.

The main program is organized according to a modular principle and includes the main part (module *main*()), auxiliary functions, and the DMI modeling module. The set of auxiliary functions includes the following functions: MakeEdge – generates a list of the graph's arcs and their weights; GetJson – analyzes the JSON file and constructs the CTS digraph based on it; AddToCsvFile – is used for collecting information into CSV files; *Impuls* – simulates the propagation of the DMI through the digraph for the given

MS.

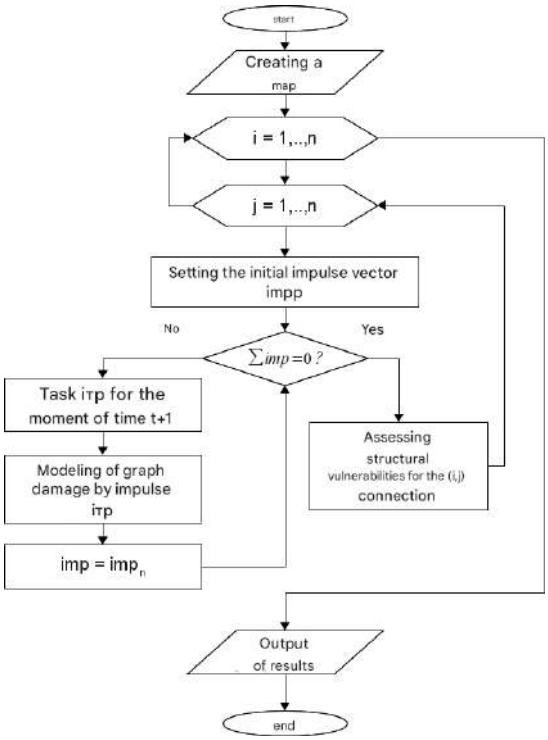

Figure 4.14. Algorithm for assessing the structural threats of IC in the CTS

The “highlight” of the algorithm design is the numbering of the digraph arcs, which is implemented using Python language dictionaries, where the entry key is a data type “tuple” consisting of two elements - the names of the initial and final node of the arc. In particular, for the digraph shown in Fig. 4.10.

Such an array looks as follows

{('01', '04'):1, ('01', '05'):1, ('02', '06'):1, ('02', '09'):1, ('03', '07'):1, ('03', '11'):1, ('04', '08'):1, ('04', '09'):1, ('05', '09'):1, ('06', '09'):1, ('06', '10'):1, ('07', '10'):1, ('07', '11'):1, ('08', '12'):1, ('08', '13'):1, ('09', '13'):1, ('10', '13'):1, ('10', '14'):1, ('10', '18'):1, ('11', '14'):1, ('11', '17'):1, ('11', '21'):1, ('12', '15'):1, ('12', '16'):1, ('13', '15'):1, ('13', '17'):1, ('14', '18'):1, ('14', '19'):1, ('14', '20'):1, ('14', '21'):1}

Such a representation facilitates the formation of the data array for the digraph arcs and working with them, since it enables the automated organization of arc lists based on the initial digraph matrix, as well as their processing during the simulation of DMI propagation. In particular, the aforementioned MakeEdge subroutine becomes compact in terms of both performance and structure:

```
def MakeEdge(v,Vs,e):
  for i in Vs:
    if len(v[i]) > 0:
      ishod = v[i].keys()
      ishod.sort()
      for j in ishod:
        e[(i,j)] = v[i][j]
```

The input of this program consists of the dictionaries *v* and *e*, as well as the ordered list of vertices V_s. Initially, the dictionary *e* is empty, while the dictionary *v* contains the data obtained from the JSON file. If there is a connection between node *i* and node *j* during the digraph analysis, a "key-value" pair in the format (*i*,*j*):1 is added to the dictionary *e*. That is, it enables the formation of connection lists for any possible configuration specified in the initial JSON file.

The presented code fragment reflects a practical method of constructing a graph and generating a list of arcs using Python tools; however, for analyzing scalability and further adapting the algorithm to other computing platforms, it is advisable to extract its logical foundation in the form of a generalized DMI (inter-component impulse flow) propagation algorithm. Such a formalized approach allows describing the model independently of the specific programming language and clearly evaluating computational costs.

Impulse propagation algorithm in the cognitive simulation model

Input:

G(V, E) – oriented graph with edge weights $e_{ij} \in (0, 1]$

Imp(v) – vector of initial impulses

τ_max *– maximum iteration count*

Output:

F(v) – final impact (damage) levels of nodes

Algorithm PulsePropagation:

1: Initialize $F(v) \leftarrow 0$ *for all* $v \in V$

2: Set current_imp $\leftarrow$ *Imp(v)*

3: $t \leftarrow 0$

4: while $t < \tau\_max$ *and* $||current\_imp|| > \varepsilon$ *do*

```
5:     next_imp ← zero vector
6:     for each edge (i → j) ∈ E do
7:         next_imp[j] += current_imp[i] * e_ij
8:     end for
9:     F(v) ← F(v) + next_imp(v)
10:    current_imp ← next_imp * α      # α – attenuation factor
11:    t ← t + 1
12: end while
13: return F(v)
```

In the terminology of the cognitive model, the graph nodes correspond to the subsystems or modules of the CTS, while the arcs represent influence relations and the transfer of disturbances. The impulse propagating along the arcs models the cascading development of failures or the degradation of functional connections.

Computational complexity assessment

Each iteration requires traversing all graph arcs E, therefore the total complexity is:

$$T(n)=O(|E|\cdot\tau),$$

where τ is the number of iterations until stabilization (typically $\tau \ll |V|$).

When using sparse data structures, the computational complexity approaches $O(|V|+|E|)$, and the required memory is linear with respect to the number of system elements. This property makes the algorithm scalable for modeling CTS with thousands of nodes.

Thus, the principles, criteria, and methodology for assessing the influence of the state of inter-module connections on the survivability of the CTS have been developed. An algorithm for assessing structural survivability based on the method of evaluating the influence of the state of inter-module connections has been developed and implemented, and the software-specific features of the method's implementation have been described. The obtained results will serve as the basis for the further development of a method for assessing the influence of ICs on the survivability of hierarchical CTS, in which the emergence of closed-loop interaction patterns is possible, as well as for the assessment of CTS in which the connections are represented by non-binary parameters.

To verify the developed DMI propagation modeling algorithm, numerical experiments were conducted using directed graphs of different sizes and connection densities. The purpose of the experiments was to verify the correctness of the results, the robustness of the algorithm under changes in

the graph structure, and the evaluation of scalability with respect to the number of nodes and arcs.

Table 4.1

Parameters of experimental scenarios

Parameter	Scenario A	Scenario B
Number of nodes, N	21	100
Number of arcs, E	*	*
Average connection density	0.18	0.24
Initial impulse node	No. 3	No. 10
Initial impulse level I_0	1.0	1.0
Attenuation coefficient α	0.8	0.85
Maximum number of iterations τ_{max}	10	20
Stabilization criterion ε	10^{-3}	10^{-3}
Structure storage format	JSON	JSON
Execution environment	Python 3.10 / NetworkX	Python 3.10 / NetworkX

For both scenarios, the initial graphs were constructed according to the principle of hierarchically interconnected subsystems with varying degrees of connectivity. The simulation was performed with a fixed value of the attenuation coefficient α, which corresponds to the gradual degradation of the impulse during its propagation along the arcs of the cognitive model.

Simulation results

Based on the computational experiment, the values of the integral indicators were obtained - the systemic impact coefficient *K_{SA}*, the structural stability coefficient *K_{SS}*, the impulse propagation time *t_d*, and the computational time *t_{calc}*.

Table 4.2

Summary of numerical experiment results

Scenario	K_{SA}	K_{SS}	Number of affected nodes	Impulse propagation time t_d (s)	Execution time t_{calc} (s)
A (N=21)	0.67	0.78	15	0.83	0.012
B (N=100, E = 240)	0.71	0.82	68	—	—

Analysis of the obtained data shows that as the number of nodes and the density of connections increase, the values of the indicators *K_{SA}* and *K_{SS}* grow, which reflects a more pronounced cascading effect of failure propagation in highly connected

structures. At the same time, an increase in the computation time t_{calc} is observed; however, it is linear in nature, which confirms the scalability of the proposed algorithm $O(|V|+|E|)$.

To analyze the structure of failure risk, the K_{SA} metric was calculated, characterizing the degree of influence of each node on the cascading propagation of disturbances. Figure 4.14 shows the distribution of K_{SA} values for the ten most critical nodes of scenario B (a model with $N = 100$ and connection density 0.25).

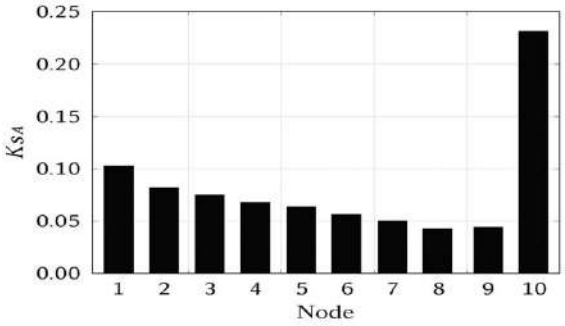

Figure 4.15. Distribution of K_{SA} values for the 10 most critical nodes of scenario B

From the analysis of the distribution of the structural influence coefficient K_{SA}, it is evident that nodes Node 1–Node 4 demonstrate values $K_{SA} > 0.8$, which indicates their dominant contribution to the propagation of the DMI throughout the network. These nodes form the core of the cascading influence, through which the main routes of perturbation transmission pass. The second group of nodes (Node 5–Node 8) has intermediate values $K_{SA} = 0.45–0.65$, which characterizes them as buffer components that limit the speed of DMI propagation and stabilize local subsystems. The remaining nodes (Node 9–Node 10) with $K_{SA} < 0.3$ perform peripheral functions, and their failure has minimal impact on the system's resilience.

Thus, the graphical distribution of the K_{SA} coefficient makes it possible to formalize the procedure for vulnerability ranking and identify priority directions for technical monitoring and redundancy within the cognitive survivability model of the CTS.

Figure 4.16 shows the dependence of the computation time t_{calc} on the number of edges $|E|$. The obtained data confirm that the execution time of the algorithm increases almost linearly with the

growth of the number of connections, which corresponds to the theoretical complexity estimate O(|*V*|+|*E*|).

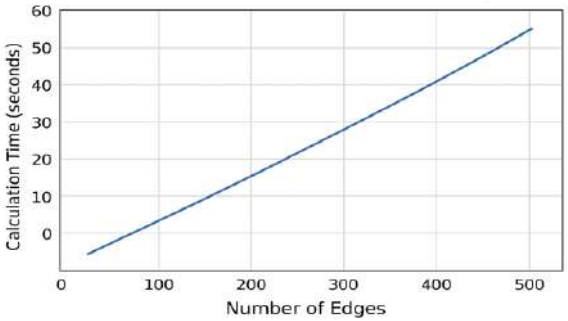

Figure 4.16. Dependence of computation time *t_{calc}* on the number of edges |*E*|

The linear nature of the dependence demonstrates good scalability of the proposed DMI propagation simulation algorithm. This makes it possible to use it for the analysis of large-scale systems (with more than 10^3 vertices) without significant performance loss, which confirms the efficiency of the chosen data structure and the graph traversal method.

To obtain a summarized quantitative representation of DMI propagation in scenario B, a heat map of impact intensity per node at each time step was constructed (Fig. 4.17). The map visualizes how the impulse propagates through the topology and gradually attenuates due to edge weakening and local damping. The map allows easy identification of nodes with prolonged exposure (persistent “hot spots”) and periods of maximum network load.

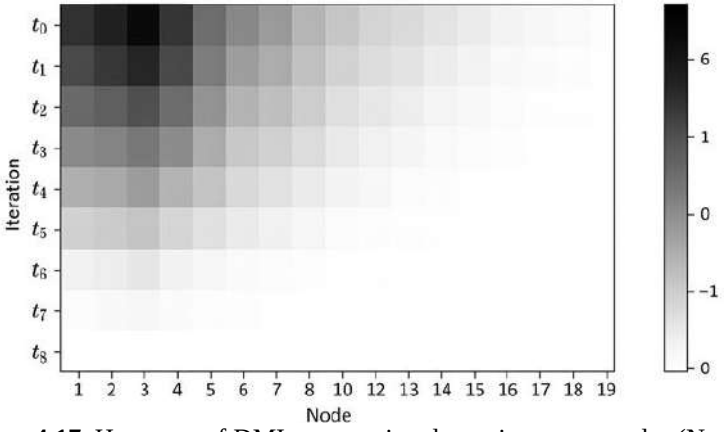

Figure 4.17. Heat map of DMI propagation dynamics across nodes (N = 21) over T = 10 iterations

X-axis - node indices (1..21), Y-axis - iterations t0..t9 (t0 is the top row). Intensity (gradations from white to black) is proportional to the value *I_j(t)* (normalized to the maximum). Parameters: β = 0.8, γ = 0.1, initial node - Node 1.

The mathematical model used when updating the heat map in Fig. 4.17 is as follows:

$$I_j(t+1) = \beta \sum_{i \in pred(j)} \omega_{i,j} \cdot I_i(t) - \gamma \cdot I_j(t), \tag{4.57}$$

where $I_j(t)$ - the impact intensity at node *j* in iteration *t*;

pred(j) - the set of predecessor nodes having edges directed into *j*;

$\omega_{i,j}$ - the weight $i \to j$ of the edge (in the simplest case 0/1, or a numerical weight);

β - the transmission coefficient along the edge (amplification/attenuation during the propagation of the damaging impulse, e.g. β = 0.8);

γ - local damping/leakage in the node (for example, γ = 0.05–0.15).

The initial condition - select one or several starting vertices with $I_{start}(t_0) = 1.0$ (or another value), and all other vertices are initialized to 0.

Analysis of the heat map shows: (i) an initial strong local impact around the starting vertex; (ii) during the first 2–4 iterations, an intensive expansion of the affected area is observed; (iii) subsequently - gradual attenuation and transition to a steady state by iterations t_8–t_9. Nodes that maintain high intensity after t_5 correspond to vertices with high K_{SA} values (see Fig. 4.15) and form the “core of cascade influence.”

Table 4.3 presents the quantitative characteristics of DMI propagation across the 10 most critical nodes of the considered graph (scenario B, *N* = 21, *T* = 10 iterations). For each node, the following are specified: the maximum impact intensity over the entire time I_j^{max} (dimensionless, normalized to 1), the time (iteration) of achieving this maximum t_j^{max}, the total impact energy $E_j = \sum_{t=0}^{T-1} I_j(t)$, and the proposed normalized metric of local contribution to cascade propagation K_{SA} (based on integral energy and peak response). Units are conditional (intensities and energies are normalized for convenience of comparison).

Table 4.3

Key characteristics of DMI propagation across the 10 most critical nodes (N = 21, T = 10)

№	Узел (node)	I_j^{max}	t_j^{max} (iteration)	$E_j = \sum_{t=0}^{T-1} I_j(t)$	K_{SA} (norm.)
1	Node_10	1.00	1	4.26	1.00
2	Node_11	0.94	1	3.85	0.90
3	Node_14	0.88	2	3.20	0.75
4	Node_03	0.82	0	2.98	0.70
5	Node_07	0.77	1	2.55	0.60
6	Node_04	0.71	2	2.12	0.50
7	Node_09	0.65	1	1.84	0.43
8	Node_02	0.58	0	1.41	0.33
9	Node_06	0.52	2	1.12	0.26
10	Node_01	0.45	0	0.86	0.20

In Table 4.3, I_j^{max}- the maximum (peak) intensity of influence on node j over all T iterations (normalized to 1 for the strongest peak in the sample); tjmax t_j^{max} - the iteration at which I_j attains its peak (iterations are numbered 0.. T−10); E_j - the integral energy of the node (the sum of intensities across all iterations); it characterizes the total impact on the node during the simulation; K_{SA} - the normalized estimate of the node's contribution to cascade propagation (proposed for comparability with analytical survivability metrics); it is computed as a weighted combination of E_j and I_j^{max}, normalized by the maximum value (Node_10 is taken as 1.00).

The analysis of Table 4.3 confirms the visual conclusions obtained from the heat map (Fig. 4.17). Nodes with high Ijmax I_j^{max} and large integral energy E_j form the "core of cascade influence" - it is through them that the main part of the DMI flow passes, and they determine the largest share of the systemic risk during the development of an emergency scenario. In the presented data, Node_10 and Node_11 stand out as critical points: they demonstrate rapid intensity growth (peak at early iterations t=0÷1), high total energies, and maximal K_{SA} values. This means that when developing survivability enhancement measures it is advisable to focus resources on monitoring, redundancy and strengthening control specifically for these nodes (emergency measures: redundant sensors, predictive maintenance, physical duplication, etc.).

Ijmax I_j^{max} and t_j^{max} allow assessing which nodes provide early warning signals (early peaks = potential "accident radar"). E_j is useful for prioritizing prolonged impacts: a node with a large E_j accumulates most of the event energy and potentially requires measures to reduce accumulated

damage. K_{SA} is an aggregated metric for ranking nodes when planning survivability enhancement measures. The values shown in the table are synthetic. In real tests these quantities are computed directly from the time series $I_j(t)$ (the simulation output).

1. The procedure for computing K_{SA} should be formally specified in the text (for example,

$$K_{SA} = \alpha \cdot \frac{E_j}{\max(E)} + \beta \cdot \frac{I_j^{\max}}{\max(I^{\max})},\ \alpha + \beta = 1 \qquad (4.58)$$

In an appendix or methods section it is recommended to provide concrete values of the weights α,β\alpha,\betaα,β and justification for their choice.)

2. For completeness of the study, it is useful to add a column “impact on adjacent nodes” (for example, the number of directly affected neighbors) and/or a metric of propagation time to a given threshold (t_d) - this will give a more complete picture of vulnerability.

In the assessment of the survivability of technical systems for the purpose of providing information support for decision-making at the operation stage and for proactive actions in the event of an emergency involving an CTS, it is necessary to develop a decision support system (DSS). The developed DSS is based on a CSM for assessing the survivability of an CTS, combining interacting systems, subsystems, and individual technical nodes, which are considered as objects with individual characteristics. The interaction of nodes is determined by their incoming and outgoing parameters. The DSS assesses the overall state of technical systems using a unified system of node parameters, as well as the values of node characteristics and their interaction, based on the value and dynamics of incoming and outgoing parameters.

Each node is connected to the system either as a virtual component from a node library (a model of an operating node of a marine technical system) or as a real component receiving diagnostic parameters of technical systems via an information interface and converting them into DSS system parameters. Both hardware and software interfaces are used. The application of a unified parametric model and system interfaces for individual nodes makes it possible to include in the model nodes from a wide variety of systems and the broadest possible range of configurations. For a generalized CTS, the total number of nodes is assumed to be mmm. The following individual characteristics are selected for each node: service life (S_t); current state (S_n) and the state vector of the node; state probability vectors and node maintainability vector; resource vector; node influence

matrix; confidence coefficient. The characteristics of the node and its interaction with adjacent nodes of the system, in accordance with the CTS CSM, are shown in Fig. 4.18. The service life is defined at the moment of the system's creation, automatically increases as the node is operated, and changes at the moment the node is replaced by a new one. Each node is characterized by nnn states S_n, each of which reflects the evolution of the node's operability.

In this case, S_0 is taken as the fully operational state of the node, and S_i, for i=1,..,n, reflect the transition of the node into progressively less operational states. In the simplest case, the node has two states: n=0 - the node is operational, and n=1 - the node is non-operational. The "node state" characteristic S_t reflects its condition at the current moment in time. Intermediate values for the number of states n>2 may reflect various types of node failure (breakdown, overheating, vibration above design levels, etc.), as well as several stages of node wear (in particular - different distribution states).

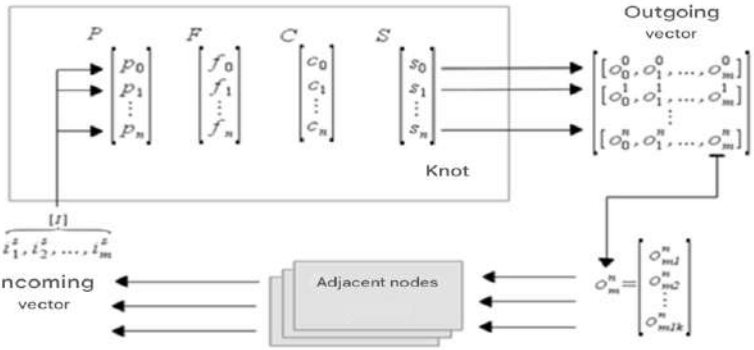

Figure 4.18. Characteristics of a node and its interaction with adjacent nodes of a technical system

Influence parameters are treated as dimensionless scale-free quantities in the range from 0 to 1, where 0 reflects a complete absence of connection (failure of the node does not affect the adjacent node), and 1 reflects a direct connection (failure of the node unconditionally leads to failure of the adjacent node).

The node influence matrix is a combination of the incoming and outgoing influence vectors, reflecting the interaction of the node with other nodes of the system. The incoming influence vector reflects the influence parameters on this node from other nodes of the system and depends on the

state of the system's nodes connected to this node. The outgoing influence vector reflects the influence parameters of this node on the system's nodes connected to it and changes as the system transitions through each of the nnn states. Consequently, the outgoing vector *O* consists of nnn influence matrices. Each influence matrix is an *m×k* combination, where mmm is the number of nodes in the system adjacent to the given node, and *k* is the number of possible states of adjacent nodes. The matrix entry value is a modifier of the probability of state transition of the adjacent node. Accordingly, the full outgoing influence matrix of a node is an *m×k* composed of nnn influence vectors with mmm parameters in each of them. The node state probability vector *P* reflects the probabilities of transition of the node into one of the nnn states. The probability of transition of the node into one of the nnn states is a function of the current node state *S*, its service life *T=t*, as well as the incoming influence vector modeling the incoming influence parameters from other nodes of the system. The node maintainability vector *F* indicates the state of the node in each of the nnn states and reflects its ability to perform the assigned task over the interval *Δt*.

$$F_i = \frac{P_1^t}{P_o^t} = \frac{P(S_i I \Delta t)}{P(S_o I \Delta t)} \tag{4.59}$$

The presented characteristic is qualitative, corresponding to the ratio of the probability of performing the target operation in state S_0 to the probability of performing the operation in . In the case of two node states, such a vector will have only two values – $F_0 = 1$ (the node is operational) and $F_1 = 0$ (the node is non-operational)

$$F_i == \frac{P(S_o I \Delta t)}{P(S_o I \Delta t)} = \frac{1}{1} = 1, \tag{4.60}$$

$$F_0 = \frac{P(S_1 \mid \Delta t)}{P(S_0 \mid \Delta t)} = \frac{0}{1} = 0$$

The recovery resource vector reflects the cost of restoring the node to state . It may have both a temporal component (man-hours, active restoration time under optimal personnel allocation) and an economic component.

At each moment in time, a system node is characterized by several generalized parameters: state, service life, expected operability, expected damage, and criticality. The expected operability is the mathematical expectation of the node's operability and is calculated by the formula

$$F_{\Sigma} = \sum_{i=0}^{n} f_i \cdot p_i \tag{4.61}$$

The expected recovery resource of a node is the mathematical expectation of resource expenditures for restoring the node and is calculated by

$$C_{\Sigma} = \sum_{i=0}^{n} c_i \cdot p_i \tag{4.62}$$

The functional-structural relationships of the model are characterized by: the outgoing connections vector *A*, which is an associative array; the incoming connections vector *I*, which is an associative array; and the position of the object in the system hierarchy L, which reflects the relations, nature, and interaction of the object with other hierarchical objects, its influence on other objects, and on the entire CTS complex as a whole. The weights of the connections are selected based on statistical forecasting methods or expert evaluation.

The results of studies using the expert assessment method are used as criteria for formalizing the values of operability and the weights of incoming and outgoing connections.

In addition to objects, the model also includes a set of interconnection matrices (IM), each of which reflects the relationship between system objects for a specific type of connection. In the most general case, when considering only the statistical interaction of CTS objects, only the main interconnection matrix is used from the entire set of IMs - it reflects the influence and interrelation in terms of the operability of elements. The operation of the CTS model is based on the concept of events affecting the objects of the model and in some way changing them and their characteristics. The model does not require that all parameters participate simultaneously in survivability assessment and, thanks to its object-oriented architecture, can include automatic collection of system state information.

Let us present the individual characteristics of CTS survivability model objects as follows. The state vector of an object S_0, …, S_n characterizes the object with n states, each reflecting the evolution of the node's integrity. In accordance with [58],

S_0 is taken as the absolutely functional state of the node, S_n as the failure state of the node, and the set of states S_i for i=1,..,n reflects the transition of the node to progressively less functional states. In the simplest case, the node has two states: n=0 – the node is functional, and n=1 – the node is non-functional.

The probability estimate of the object's state transition P_i indicates the probability of the object transitioning to state S_i within the time interval t.

Thus, for the current state of the system S_c, it can be written as follows:

$$\left\{\begin{array}{c} P_0 = P(S_c \to S_0) \\ P_1 = P(S_c \to S_1) \\ \cdots \\ P_n = P(S_c \to S_n) \end{array}\right\} \tag{4.63}$$

The state vector $P_0,\ldots,P_n$ in (1) can be used for describing and predicting the behavior of the object over the time interval t. The operating time of the object T and the time of the last operability test of the object T_l, as in [41], are used in the calculation of the object's amortization, as well as in the diagnostics of the system for monitoring its components. The operability assessment of the object F, in the general case, expresses the probability of the object performing the assigned task within the specified time t and is determined by

$$F_i = \frac{P_1^t}{P_0^t} = \frac{P(S_i \mid \Delta t)}{P(S_0 \mid \Delta t)} \tag{4.64}$$

Then the operability assessment of the object (4.63) F for n possible object states

$$\left\{\begin{array}{c} F_1 = \dfrac{P_1^t}{P_0^t} = \dfrac{P(S_1 \mid \Delta t)}{P(S_0 \mid \Delta t)} \\ F_2 = \dfrac{P_2^t}{P_0^t} = \dfrac{P(S_2 \mid \Delta t)}{P(S_0 \mid \Delta t)} \\ \cdots \\ F_n = \dfrac{P_n^t}{P_0^t} = \dfrac{P(S_n \mid \Delta t)}{P(S_0 \mid \Delta t)} \end{array}\right\} \tag{4.65}$$

The operability assessment (4.65) is a qualitative characteristic, which can also be obtained based on expert evaluation, reflecting the ratio of the probability of performing the target function in state S_0 to the probability of performing the function in state S_i.

The expected operability over the time interval t is evaluated by the mathematical expectation of the object's operability

$$F_{\Sigma} = \sum_{i=0}^{n} f_i p_i \,, \tag{4.66}$$

where f_i is the operability of the object in state i;
p_i is the probability of transition to state $\underline{i}$

In the case of two object states, the vector (4) has the values – $F_0 = 1$ (the object is operational) and $F_1 = 0$ (the object is non-operational).

$$\begin{aligned} F_0 &= \frac{P(S_0 \mid \Delta t)}{P(S_0 \mid \Delta t)} = \frac{1}{1} = 1 \\ F_1 &= \frac{P(S_1 \mid \Delta t)}{P(S_0 \mid \Delta t)} = \frac{0}{1} = 0 \end{aligned} \,, \tag{4.67}$$

The resource estimate C_i represents the estimate of the cost of restoring the object from its current state to the operational state

$$C_i = W(C_i \to C_0) \tag{4.68}$$

For a fully operational object, (4.68) takes the form

$$C_0 = W(C_0 \to C_0) = 0$$

The units of measurement can be either temporal (man-hours) or financial costs. The overall assessment of work on the object is the mathematical expectation of resource expenditures for restoring the object

$$C_{\Sigma} = \sum_{i=0}^{n} c_i p_i \tag{4.69}$$

where c_i is the operability of the node in state i.

The operation of the developed CTS survivability assessment system is possible in several modes. In the integral assessment mode, the system indicates potential failures at different moments in time, assesses their criticality, and provides recommendations for restoring/replacing nodes depending on their impact on the system and the economic effect. In the differential assessment mode, the system evaluates each hierarchy of technical systems according to the "system – subsystem – node" category and provides recommendations similar to the integral assessment mode. In the search mode, the system simulates possible failures and node breakdowns with various target search functions. The target functions include: identification of critical and vulnerable points of the system; evaluation of various system prevention options (depending on criticality and economic effect, which may be expressed in both price and resource

expenditure form); worst-case scenarios of possible failures and the possibility of “emergency resonances” (cases where the failure of minor elements may trigger a cascading chain of breakdowns and system failures).

The results of modeling the propagation of DMI demonstrated that the survivability of complex technical systems is determined not only by the local resilience of individual elements but also by their mutual structural interconnectedness. The analysis of cascading effects made it possible to identify the nodes with the greatest integral contribution to the reduction of systemic stability, which forms the basis for targeted reinforcement of critical components. The developed cognitive-simulation model can be considered as a foundational level for building more advanced survivability assessment models, incorporating cognitive, fuzzy, and neural network approaches. Further on (subsection 4.3), extended methodological and instrumental solutions are considered, aimed at the formalization, quantitative assessment, and optimization of the survivability of CTS under conditions of uncertainty and dynamically changing external and internal environmental factors.

4.3 Intelligent–digital models for analyzing survivability and failure risk of complex technical systems

4.3.1 Methodological foundations of cognitive modeling of technical system survivability

In order to ensure the flexibility of the virtual cognitive model of a vessel for assessing the survivability of its technical systems, an object-oriented CSM has been developed, in which each component is represented as a relatively autonomous object. Each object of the model is a data structure containing the individual characteristics of the CTS elements, as well as parameters reflecting their functional-structural relationships with other elements of the system. In practical implementations, the cognitive-simulation model receives input data (telemetry, event logs, failure statistics) from monitoring sources, including the digital twin. In subsection 4.3, the CSM is considered as a methodological and simulation tool; integration with the DT is considered as a way to enrich input data and automatically update P_0 and features, while the model itself remains the subject of research.

The purpose of this approach is to formalize a methodology for assessing the survivability of complex technical systems based on modeling interactions between elements under the impact of damaging factors and internal failures. This approach ensures adaptability and scalability of the model, allowing it to be used both in local and integrated survivability analysis scenarios.

The object-oriented architecture of the CSM allows each element to be

assigned its own behavior, rules of reaction to incoming impacts, as well as recovery dynamics after failure. This forms the basis for simulation modeling of the processes of degradation and recovery of CTS, providing the capability to assess systemic survivability metrics and identify critical nodes.

The structure of the proposed survivability assessment methodology is presented in Fig. 4.19. It reflects a sequence of stages - from the description of a complex technical system to the calculation of survivability metrics and the formation of control decisions. The diagram shows that the core of the methodology is the CSM which ensures the integration of structural, functional, and dynamic aspects of CTS behavior.

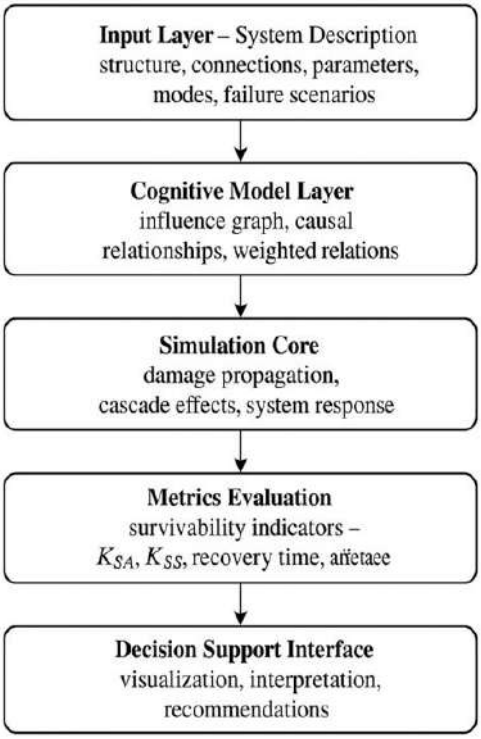

Figure 4.19. Methodological scheme for assessing the survivability of a complex technical system based on cognitive-simulation modeling

Simulation of the propagation of failures and external impacts enables a quantitative assessment of the resilience and effectiveness of recovery processes. The obtained survivability metrics are used in the decision

support subsystem, where the modeling results are interpreted for practical tasks of operation and modernization.

Interaction of objects in the model is carried out via data exchange at the moment of an event that reflects occurring changes in one or more objects. When modeling a technical complex, its characteristics reflect generalized behavior and impacts on adjacent, interconnected technical systems. An object of the model may exist in two ("operational", "failed") or more states (intermediate values indicate partial operability of the object and/or its assessment on a dimensionless scale).

The individual characteristics of an object include:

- the object descriptor *D*, which serves as its identifier in the system;
- the state *S*, represented as a state vector of the object $S_0, \ldots, S_n$ (normal, failed, intermediate);
- the estimated probability of state transition of the object within the next time interval *t* - $P_0, \ldots, P_n$;
- the operating time *T* of the object as of the current moment, expressed either in calendar hours or in direct operating hours;
- the time of the last functionality test of the node - T_l;
- the assessment of node operability $F_0,\ldots,F_n$ for each of its respective states $S_0,\ldots,S_n$, represented as a vector;
- the resource assessment vector *C* for restoring the system from its current state to the "zero" state (fully functional state).

To improve reproducibility of the model and ensure unambiguous software implementation, Table 4.4 presents the data types, measurement units, and allowable value ranges for the parameters of a cognitive model object.

Default values and ranges are provided for reproducibility; specific values are determined by the subject area (project, operation, DT data).

Sources for deriving initial failure probabilities P_0. For practical applicability of the cognitive-simulation model, it is recommended to use one of the following three approaches for generating input failure probabilities P_0:

1. Statistical - estimation based on historical data (logs, repairs, failures). This is the preferred method when large volumes of telemetry data are available (e.g., from a digital twin or corporate registries);

2. Expert-based - used when sufficient statistics are not available; calibration methods should be applied (Delphi, pairwise comparison) accompanied by sensitivity analysis;

3. Hybrid/integration-based - if a digital twin is available, preliminary probabilities (RUL models, simulations) may be obtained from it and then adjusted with statistics and/or expert assessment. Recommendation: if a digital twin is available, integrate its predictions via an interface (API) and

periodically re-estimate P_0 during model validation and retraining.

Table 4.4

Parameters of a cognitive model object of the CTS

Parameter	Semantic purpose	Data type	Unit of measurement	Range / allowable values
D	Node identifier	string	—	unique code
S	Current state	enum	—	{normal, degraded, failed}
P	Failure probability (P_0)	float	—	[0,1]
T	Operating time (service hours)	float	h	≥0
T_l	Time of last testing	float	h	≥0
F	Normalized operability	float	—	[0,1]
C	Residual resource (residual capacity)	float	arb.units	≥0
Link_weight	Link coefficient (i→j)	float	—	≥0 (recommended ≤2)
Threshold_fail	Failure threshold	float	—	[0,1] (e.g., 0.2–0.4)

The threshold-based logic described above allows the continuous operability value F (or w) to be mapped to a discrete state S. This mapping is crucial for clear visualization and decision-making. We introduce two key thresholds for system definition: F_{crit} (critical failure threshold) and F_{fail} (degraded operation threshold), which can be specialized for each object.

Table 4.5

Interpretation of node state based on normalized operability F

State (S)	Operability range (F)	Interpretation	Visualization note
Normal (S_0)	$(F_{fail}, 1]$	Full or slightly reduced performance; node is fully operational.	Solid outline
Degraded (S_1)	$(F_{crit}, F_{fail}]$	Operability is reduced, system may exhibit partial failure or require immediate attention (e.g., node W_O in example).	Dashed outline
Failed (S_2)	$[0, F_{crit}]$	Critical failure state; node cannot perform its primary function.	Solid bold or red outline

For the simulation examples presented in the monograph, we adopt F_{crit} = 0.2 and F_{fail} = 0.5. This explains why nodes with operability 0.52 are shown with a dashed outline (Degraded State), while those with 0.40 (below F_{fail} = 0.5) are also in the degraded state. The explicit distinction between the Degraded and Failed states is used in the decision support layer.

When historical data are available, it is recommended to use statistical estimation of probabilities and additionally perform sensitivity analysis of the model with respect to P_0. The state of each object in the model is updated upon receiving an external impact (impulse). The pseudocode of the impulse processing function is provided below.

For a complete definition of the CSM, it is essential to detail the parameters associated with the directed edges (links) that represent the functional and structural relationships between objects. These parameters govern the cascading effect described.

Table 4.6

Parameters of CSM edges (links)

Parameter	Semantic purpose	Data type	Range/allowed values
Link_weight[i][j] -L_w	Coefficient defining the influence transmission strength from node *i* to node *j*.	float	≥ 0 (e.g., ≤ 2)
α (attenuation coeff.)	Coefficient of impulse decay. System-wide measure of energy dissipation during impulse propagation through links.	float	(0, 1]
Link_type - L_t	The nature of the physical or functional connection (e.g., power line, control signal, data exchange).	enum	{Functional, structural, informational, etc.}

For effective use of the CSM, the dynamic state of each object is influenced not only by its intrinsic parameters (Table 4.4) but also by the characteristics of its connections (Table 4.5), particularly the attenuation coefficient α, which is crucial for controlling the rate and depth of cascading failures. The initial state of the objects in the simulation is defined by setting the operability F (or w_i) to 1.0 and specifying the initial failure probability P_0.

Algorithm OnImpulse(node i, impulse I)

delta ← compute_impact(i, I) # damage calculation based on the node's

characteristics

F_i ← max(0, F_i - delta) # updating the operability

if F_i < threshold_fail then

S_i ← 'failed' # transition to failure state

update_outgoing_effects(i, delta) # propagation of the impulse to adjacent nodes

end Algorithm

Such a block reflects the logic of simulating node degradation and the further propagation of failure through the directed graph of connections.

The degradation simulation implemented by the *OnImpulse* algorithm models failure propagation. For a comprehensive survivability analysis, the model must also incorporate the recovery dynamics, enabling the assessment of recovery time metrics (T_{rec}) and the effectiveness of control decisions.

Recovery dynamics

Recovery is modeled as a process that increases the node's operability $w_i(t)$ over time, consuming the available resource capacity C. This mechanism is activated when a control decision to repair a specific node i is made, provided external resources are available.

The actual recovery process is implemented through a separate management function that activates recovery logic on a specified node when resources are available and the node's operability is below the maximum threshold (w_{max}=1). This algorithm explicitly uses the resource assessment vector C and the defined failure thresholds (F_{crit}).

Algorithm OnRecovery(node i, available_resource R)

Check if node is failed and recovery is required

*if w[i] > F_crit and R > 0 then # Determine the required effort based on the node's current state required_effort = C[i] * (1 - w[i]) # Example: Total effort proportional to remaining deficit*

 # Calculate the achievable recovery rate based on available resources

*recovery_rate = min(R, required_effort) / C[i] * R_rate[i]*

 # Update operability (using the formal equation)

*w[i] = min(1, w[i] + recovery_rate * delta_t)*

 # Update available resources (resource consumption)

*R = R - min(R, required_effort) * delta_t*

 # Check for state transition after recovery

if w[i] > F_fail then

S[i] = 'normal' # Transition from degraded to normal state

record_recovery_metrics(i, recovery_rate, R)

end if

end Algorithm

This separate, resource-dependent recovery algorithm completes the lifecycle modeling of system survivability, allowing the decision support

system to optimize resource allocation *R* based on calculated recovery rates and required effort *C*.

The function compute_*impact()* can be implemented as linear, exponential, or stochastic, depending on the nature of the impact and the specifics of the modeled subsystem. Using the example of impulse modeling, the operation of the DSS is considered for possible events and scenarios. The impulse impact modeling was carried out using a program written in Python, employing *graphviz* visualization tools.

One of the ways to assess the survivability of a system is to simulate an impact on it that leads to changes in the weights of the graph vertices according to the impulse rule. The impulse impact is defined by an impulse vector Imp of the form: *impj(t), j = 1, 2, ..., n* for discrete time $t = 0, 1, 2, 3...$, which is specified by a relation of the form

$$imp_j(t) = w_j(t) / w_j(t-1) \tag{4.70}$$

Formula (4.70) defines a discrete scheme for updating the weight of a vertex $w_j(t)$ upon receiving an impulse $imp_j(t)$. For clarity, we provide the explanation of the symbols used and the parameters:

- $w_j(t)$ - the weight (integral characteristic of the state, "energy"/operability) of vertex *j* at step *t* (a dimensionless value, usually normalized to [0,1] or given in relative units);

- $imp_j(t)$ - the contribution of the impulse applied to vertex *j* at step *t* (a measure of external or internal impact);

- $\alpha \in (0,1]$ - attenuation coefficient during the transmission/propagation of the impulse (energy dissipation at each step of propagation along the links) - when $\alpha < 1$, energy is lost in the process of propagation;

- N_i - the set of neighboring vertices (adjacent nodes) to which the impulse from node i can be transmitted;

- *link_weight[i][j]* - the weight of the edge (transmission coefficient from *i* to *j*), accounting for geometric/physical/logical properties of the connection - usually ≥ 0 and upper bounded (e.g., ≤ 1.5);

- *threshold_fail* - the threshold value of weight/operability below which the node is considered to have failed.

In practice, $w_j(t)$ can be interpreted as the normalized operability (1 - fully functional, 0 - out of service). The parameter α and the edge weights determine the speed/intensity of the cascading effect; their selection directly influences the depth and rate of the "cascade" of failures. It is recommended to provide specific values of α and the ranges of edge weights in the section with numerical experiments for reproducibility.

The instantaneous cascade damage $\Delta w_{cascade,i}(t)$ incurred by node *i* at

time step t from its neighbors $N_{in}(i)$is calculated by summing the weighted and attenuated incoming impulses. This formalizes the calculation component within the compute_impact function.

$$\Delta w_{cascade,i}(t) = \sum_{j \in N_{in}(i)} I_{out,j}(t), \tag{4.71}$$

where $I_{out,j}(t)$ is the impulse propagated from neighbor j (secondary damage)

The normalized operability (weight) $w_i(t)$ is then updated by subtracting both the cascade damage and any external impulse $I_{ext,i}(t)$, which corresponds to the initial damaging factor (e.g., from vector (4.73)).

$$w_i(t) = \max(0, w_i(t-1) - \Delta w_{cascade,i}(t) - I_{ext,i}(t)), \tag{4.72}$$

The outgoing impulse $I_{out,k}(t+1)$ transmitted from damaged node i to its downstream neighbor $k \in N_{out}(i)$ is proportional to the total change in operability of node i during the current step, scaled by the link weight and the attenuation coefficient α :

$$I_{outtt,k}(t+1) = (w_i(t-1) - w_i(t)) \cdot Link\_weight[i][k] \cdot \alpha \tag{4.73}$$

The operability update during recovery is formalized as:

$$w_i(t+1) = \min(1, w_i(t) + R_{rate,i} \cdot \Delta t), \tag{4.74}$$

where $Rrate,i \in [0,1]$ is the recovery rate of node i, representing the speed at which the node's operability is restored per time step. This rate depends on the node's complexity and the resources allocated;

Δt is the duration of the current time step (usually set to 1 in discrete simulations);

min(1,…) operator ensures that operability cannot exceed the maximum value of 1

Such a representation makes it possible to account for the cascading effect, in which the failure of one element triggers a chain reaction of changes in the connected nodes.

For clarity, below is a simplified pseudocode of the algorithm for updating the state of a vertex under an impulse impact:

Algorithm OnImpulse(node i, impulse I):

```
# Input: node i — index of the node, I — magnitude of the incoming impulse
# Output: updated node state and impulses prepared for neighboring nodes
delta ← compute_impact(i, I)                    # computes the local damage
w[i] ← max(0, w[i] - delta)          # update the node's weight (operability)
if w[i] < threshold_fail:
  state[i] ← FAILED
else:
  state[i] ← DEGRADED / NORMAL (according to thresholds)
# Form the outgoing impulse for neighboring nodes
for each neighbor j in Neighbors(i):
  out_impulse ← alpha * delta * link_weight[i][j]
  enqueue_event(time = t+1, node = j, impulse = out_impulse)
record_metrics(i, delta, t)      # logging for analysis/visualization
end Algorithm
```

The presented fragment illustrates the basic mechanism of impulse propagation in the cognitive model. It enables discrete transfer of energy or information between vertices, taking into account connection coefficients and losses, which makes it possible to simulate cascading degradation processes in complex technical systems. In an implementation using adjacency lists, processing a single impulse at a node requires time $O(deg(i))$, where $deg(i)$ is the degree of the node. A full propagation cycle over all events for a graph $G = (V, E)$ yields a time complexity of $O(|V| + |E|)$, provided that each node and edge is processed a constant number of times (or under reasonable stopping conditions). Memory consumption is proportional to $O(|V|+|E|)$ (storage of the adjacency list, event queues, and metrics).

A single impulse impact on the system is applied using the vector *Imp* of the form

$$Imp(0) = (imp_1 = 1, imp_2 = 1, \ldots, imp_i = I, \ldots, imp_n = 1) \quad (4.75)$$

Such an impact models the damage of object number *i* by an impulse of strength *I*. Let us consider a simplified model of the CTS, for example, the power supply system of a refrigerated vessel (Fig. 4.20), affected by a single impulse impact on one of its elements. The impact vector *Imp()* serves as a damaging factor.

The system includes three main power buses, main switches (*S1–S9*), fuses (*B1–B3*), the main power distribution board (*W*), refrigeration units (*Ref.1, Ref.2*), main and auxiliary electric power consumers (*P1–P3*), and is powered by two diesel generators (*G1, G2*) and a main generator (*S.Gen*).

Depending on the ship's energy demands and maneuvering mode, the power supply system may draw energy from: the main generator; one of the diesel generators; two generators simultaneously; the main generator and diesel generators separately; or from port power systems through switch *S7*. The schematic diagram (Fig. 4.20) is represented in generalized form as an oriented graph in Fig. 4.21.

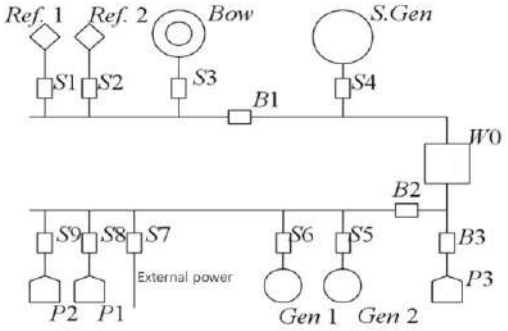

Figure 4.20. Simplified diagram of the power supply system of a refrigerated vessel

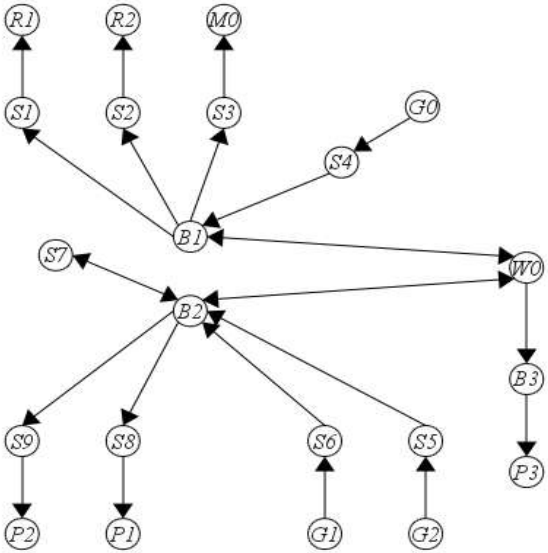

Figure 4.21. Oriented graph of the power supply system (*R1–R2* – refrigeration units; *M0* – maneuvering drive; *W0* – main power distribution board)

Thus, the first operating mode of the vessel's power supply system from the main generator *G0* corresponds to the case of an impulse action applied to object *B1*. Let all system elements be initially fully functional (in state S0 with operability *F0* = 1). To simplify the model, let us assume the interaction coefficient between objects is the same everywhere and equals 1.6.

To quantify the system's performance under stress, the methodology defines two primary survivability metrics that are calculated during the simulation (as referenced in Figure 4.19).

Integrated operability survivability index (K_{SA})

The integrated survivability index K_{SA} measures the system's ability to maintain a functional state over time under the influence of damaging factors. It is calculated as the time-averaged normalized operability across a set of critical nodes $V_K \subseteq V$:

$$K_{SA} = \frac{1}{T_{\max} \cdot |V_K|} \sum_{t=1}^{T_{\max}} \sum_{i \in V_K} w_i(t), \tag{4.76}$$

where T_{max} is the simulation horizon;

V_K is the number of critical nodes;

$w_i(t) \in [0, 1]$ is the normalized operability of node i at time step t.

K_{sa} ranges from 0 (complete functional collapse) to 1 (full operability maintained)

Structural stability index (Kss)

The structural stability index *Kss* measures the preservation of the system's ability to fulfill its primary mission or maintain critical functional connections. It is defined as the ratio of currently performed critical functions $M_{current}(t)$ to the initial total number of critical functions $M_{initial}$:

$$K_{SS}(t) = \frac{M_{current}(t)}{M_{initial}} \tag{4.77}$$

For the refrigerated vessel example, $M_{initial}$ could be the number of consumers (R1, R2, MO, P1, P2, P3) and $M_{current}(t)$ would be the number of those consumers that remain functionally connected to an operational power source at time t (i.e., nodes with $w_i(t)$ >threshold_fail).

The states of system objects after step-by-step simulation of impulse propagation are presented in Table 4.7. The propagation of the impulse is illustrated graphically by a sequence of diagrams (Figs. 4.22–4.25). Partially damaged nodes are marked with dashed lines.

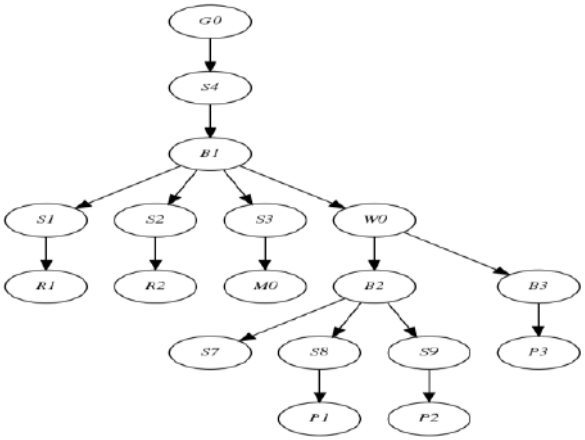

Figure 4.22. Oriented graph of the power supply system in the main generator operation mode

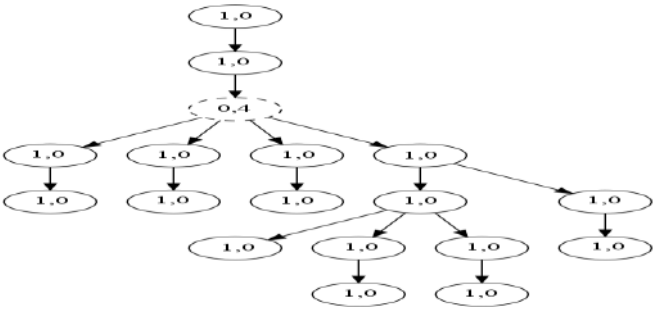

Figure 4.23. Oriented graph at time moment $T = 1$

Table 4.7

Impulse damage of node B1

T	*B*1	*B*2	*B*3	*G*0	*M*0	*P*1	*P*2	*P*3	*R*1	*R*2	*S*1	*S*2	*S*3	*S*4	*S*7	*S*8	*S*9	*W*0
0	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
1	0,40	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
2	0,40	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	0,52	0,52	0,52	1,00	1,00	1,00	1,00	0,52
3	0,40	0,68	0,68	1,00	0,68	1,00	1,00	1,00	0,68	0,68	0,52	0,52	0,52	1,00	1,00	1,00	1,00	0,52
4	0,40	0,68	0,68	1,00	0,68	1,00	1,00	0,88	0,68	0,68	0,52	0,52	0,52	1,00	0,88	0,88	0,88	0,52
5	0,40	0,68	0,68	1,00	0,68	1,00	1,00	0,88	0,68	0,68	0,52	0,52	0,52	1,00	0,88	0,88	0,88	0,52

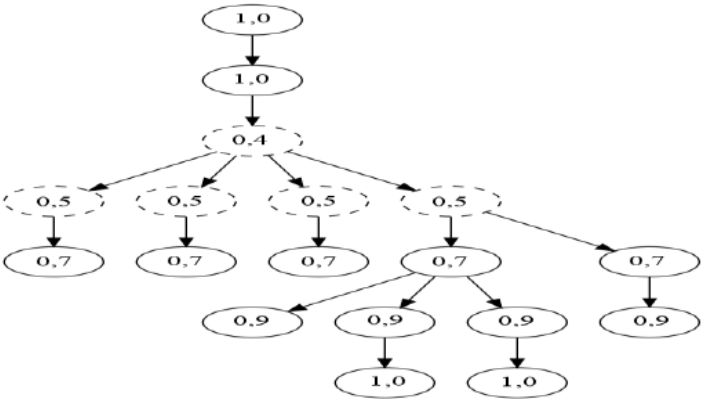

Figure 4.24. State of the graph at time moment $T = 5$

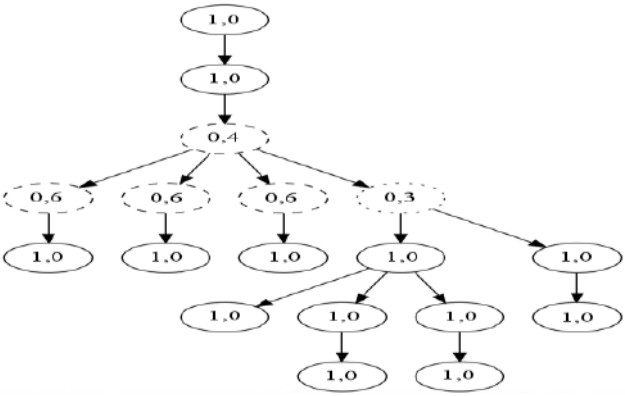

Figure 4.25. State of the graph when node *W0* is damaged at time moment $T = 3$

From Table 4.7 and the diagrams in Figs. 4.23 – 4.25, one can trace the development of the damage process within the power supply system. The impulse disturbance disabled object *B1* and propagated to objects *S1*, *S2*, *S3*, and *W0*, causing significant damage to them (their operability dropped to 0.5), and also indirectly affected the operability of objects *R1*, *R2*, *M0*, *B2*, and *B3*. At the next step, the affected objects were *S7*, *S8*, *S9*, and *P3*.

Next, let us consider the damage of a partially functional system - a system in which node *W0* is only partially operational (*F_W0* = 0.5). The partial malfunction of node *W0* (Fig. 4.25) led to its failure already at the initial stage of disturbance propagation through the power supply system. When studying the behavior of the system in a configuration where the

network is powered by the auxiliary generators *G1–G2* under an impulse impact on node *B1*, the results obtained are reflected in the diagrams in Figs. 4.26 and 4.27.

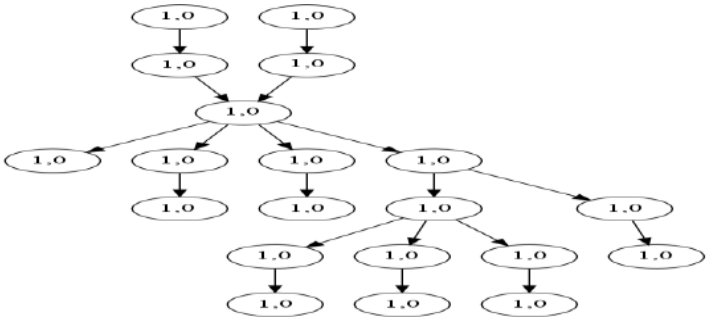

Figure 4.26. Initial configuration of the system in variant 2

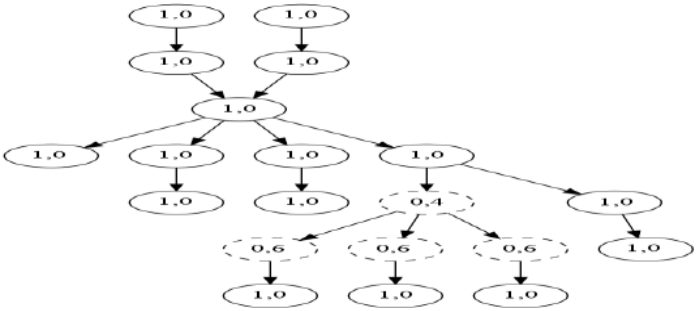

Figure 4.27. Final configuration of the system when node *B1* is damaged

From the diagram (Fig. 4.27) it follows that even with a partial malfunction of object *W0*, the damaging influence of object *B1* does not propagate to it. Thus, although objects *W0* and *B1–B3* continue to play a key role in the power supply system, ensuring the survivability of objects *B1* and *B3* is less critical than that of objects *W0* and *B2*. Similarly, the configuration of the system and the priorities of its objects will change depending on the operating mode of the power supply system, and each new configuration will lead to a change in the priorities of interactions among the system's objects and in the overall response of the system to the damaging factor. Accordingly, the assessment of the CTS survivability should also change.

From the obtained results, it follows that the survivability assessment

model of the CTS has an object-oriented nature. In the cognitive model of the CTS, redundant data field reservation is necessary, since in different system configurations, connections of various nature and functional-structural principles may arise. The completeness of accounting for these connections depends on the capabilities of cognitive system analysis; however, even at the level of relatively simple models, it is possible to obtain practically useful results.

The conducted analysis has shown that varying the parameters of impulse transmission (connection weights, attenuation coefficients, and the magnitude of the initial impact) leads to expected changes in the dynamics of failure propagation. With an increase in connection density, the speed and depth of the cascading effect grow, while with strong attenuation the process stabilizes more rapidly.

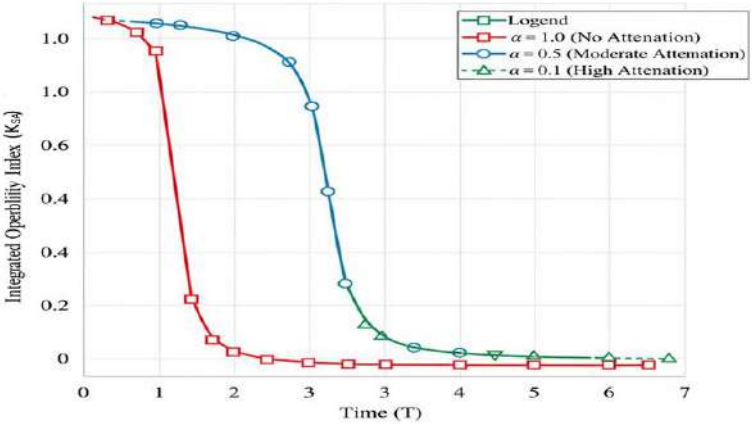

Figure 4.28. Influence of attenuation coefficient on the integrated survivability index (K_{SA}) dynamics

The results presented in Figure 4.28 visually confirm the critical role of the attenuation coefficient α in system resilience. The simulation demonstrates that:

1. When α = 1.0 (no dissipation), the cascade effect is maximal, leading to the sharpest and deepest drop in $K_{SA;}$

2. As α decreases (e.g., α =0.5 and α =0.1), the energy dissipation limits the damage spread, resulting in a shallower drop in K_{SA} and faster stabilization of the system's overall operability.

This graph serves as a powerful validation tool for the model, allowing operators to visually assess the vulnerability of the CTS configuration based

on its internal connection properties.

Thus, the CSM adequately reflects the characteristic regularities of the functioning of complex technical systems and can be used to assess their survivability and to identify vulnerable nodes. The simulation results confirm the stability and reproducibility of the developed approach, which creates a basis for the further integration of the CSM into decision support systems and digital monitoring tools.

4.3.2 Cognitive-simulation modeling of the survivability dynamics of complex technical systems

As part of the development of the cognitive-simulation approach, two types of modeling impacts were implemented-damaging impulse modeling (DMI) and diagnostic modeling impulses (DxMI). DMI modeling makes it possible to assess the structural vulnerability of a CTS-the propagation of failures between subsystems while keeping the original impulse parameters unchanged. DxMI modeling is focused on analyzing functional vulnerability, where the mission objectives and external operational conditions (e.g., environmental or resource constraints) are dynamically considered, and the impulse dynamics evolve over time. Both mechanisms are implemented in a unified cognitive-simulation software complex in Python with graphical visualization of connections (*Graphviz*) and support for JSON/CSV formats. To implement the method for assessing CTS survivability, algorithms for DMI (Fig. 4.29) and DxMI (Fig. 4.30) modeling were developed.

The structural modeling process (DMI), formalized in the block diagram (Figure 4.29), executes the core survivability logic defined in subsection 4.3.1. Specifically, the step "Modeling graph damage" corresponds directly to the Algorithm OnImpulse and is implemented using the formal equations:

- cascade damage calculation: $\Delta w_{cascade,i}(t)$ (Equation (4.71));
- node operability update: $w_i(t)$ (Equation (4.72));
- impulse propagation: $I_{out,k}(t+1)$ (Equation (4.73))

Thus, the DMI methodology is the practical application of the cognitive-simulation loop for structural vulnerability assessment, where the propagation of failure across the graph structure is systematically calculated.

The Diagnostic Modeling Impulse (DxMI) extends the core CSM logic (DMI) by incorporating the dynamic influence of the external operational environment on the system's performance. This dynamic assessment is formalized by modifying the node operability update mechanism to include a factor representing external operational constraints.

The modified node operability update for DxMI is defined as:

$$w_i(t) = \max(0, w_i(t-1) - \Delta w_{cascade,i}(t) - I_{extt,,,i}(t) - \Psi_i(t)), \quad (4.78)$$

where $\Delta w_{cascade,i}(t)$ is the cascade damage (Equation (4.71));

$I_{ext,\,i}(t)$ is the external impulse (the specific element imp_i of the vector (4.75) from Section 4.3.1)

$\Psi_i(t) \in [0,1]$ is the external operational constraint factor, quantifying the normalized impact of the mission profile, environmental stress (e.g., high seas, low temperature), or resource limitations on the operational capacity of node i at time t. $\Psi_i(t)$ is directly influenced by the dynamic parameters defined in Table 4.8.

This formal modification distinguishes the DxMI model, which evaluates functional survivability under dynamic scenarios, from the DMI model, which primarily assesses structural fragility.

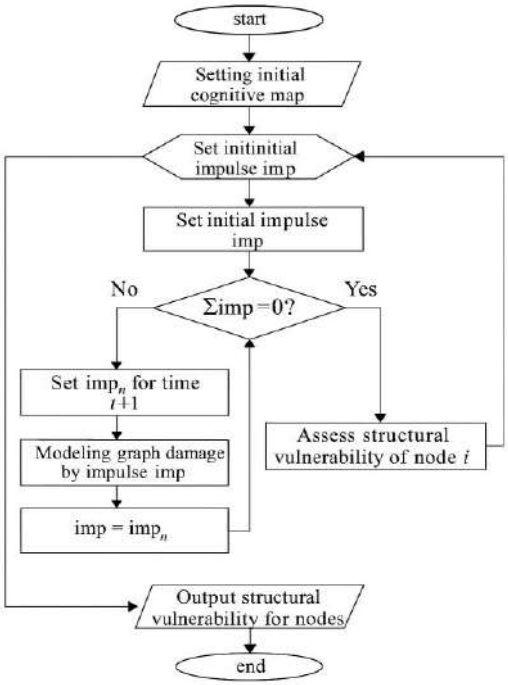

Figure 4.29. Block diagram of the DMI modeling algorithm

To implement the developed CTS survivability assessment method, a simulation software complex was created. Python, as a cross-platform

language, was used as the core tool. This made it possible to obtain both numerical and graphical results for CTS models of various classes and complexity.

The CSM method is divided into several stages. First, a cognitive map is constructed, after which structural and functional threats are evaluated via destructive and diagnostic simulation impulses. Based on the obtained threat assessments, the risk of CTS failures is calculated. In the developed method, the core of the cognitive model is a cognitive map represented as a directed graph. Depending on the modeling level, the cognitive map reflects the interaction of system subsystems with varying precision and reliability - in particular, graph arcs may represent mutual influence of system entities or resource-based connections ("energy" – "information" – "substance").

The CSM method is divided into several stages. First, a cognitive map is constructed, after which structural and functional threats are evaluated via destructive and diagnostic simulation impulses. Based on the obtained threat assessments, the risk of CTS failures is calculated. In the developed method, the core of the cognitive model is a cognitive map represented as a directed graph.

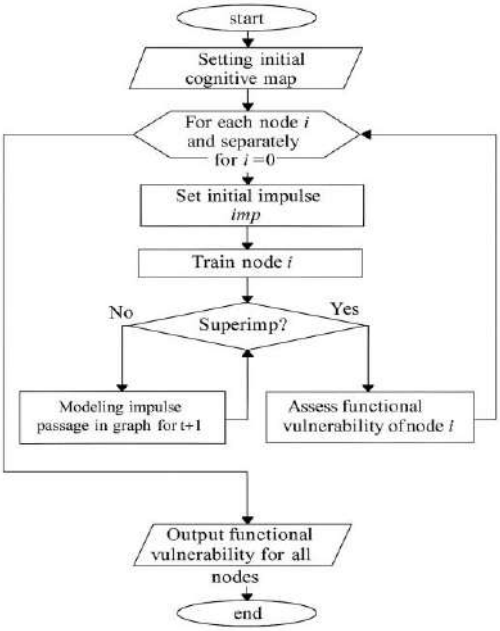

Figure 4.30. Block diagram of the DxMI modeling algorithm

Depending on the modeling level, the cognitive map reflects the interaction of system subsystems with varying precision and reliability - in particular, graph arcs may represent mutual influence of system entities or resource-based connections (“energy” – “information” – “substance”). When the exact values of arc weights and model variables are defined, the structural model transforms into a functional one. When constructing the cognitive map, local survivability, operability, and reliability indicators (obtained from CTS operation regulations) are formalized and converted into a unified system-wide, dimensionless evaluation. The cognitive map does not detail the exact nature and dynamics of interdependencies between aggregates, changes of the CTS as a whole depending on context, or its global dynamics. Considering all the above required a change of modeling level - i.e., the transition from a *cognitive map* to a *cognitive model* of the CTS. Structural vulnerabilities of the CTS are assessed using CSM. The method is implemented in an algorithm that generates datasets describing the characteristics of a real system. The evaluations obtained by manipulating cognitive model parameters according to certain abstract rules correspond to real CTS processes.

The developed CIM is based on the methodological principles described in Section 4.3.1: each model element has a set of parameters (*F*, *P*, *C*, etc.), and the impulse transfer between nodes is defined by a system of weights *w* and a damping coefficient α. This ensures compatibility between survivability and risk assessment models, as well as scalability when integrating with digital twins.

To practically implement the DxMI approach, the model parameters must be dynamically adjusted to reflect specific operational scenarios. Unlike DMI, which tests structural rigidity, DxMI varies the interaction logic to simulate different environmental and operational stresses. Table 4.8 illustrates how the attenuation coefficient α and link weights are modified for different vessel operating modes in the simulation.

By switching between these modes in the DxMI simulation, the system identifies functional vulnerabilities that are not apparent under static structural analysis.

The data in Table 4.8 confirms that the diagnostic modeling impulse (DxMI) requires a dynamic definition of the cognitive map parameters based on the operational context. Key observations are:

1. Attenuation (α): Moves closer to 1.0 (no damping) in critical modes (M_2, M_3). This reflects the physical reality that under high stress or emergency, systems become more sensitive, and failures propagate faster and wider, mimicking a heightened cascade effect.

2. Link Weight Modifier: Increases the inter-system dependence (1.2 to 1.5) during stress, indicating that the failure of one system (e.g., Power

Supply in M_2) has a disproportionately larger impact on others compared to normal operation.

3. Purpose: This dynamic tuning is the core of DxMI, allowing the model to switch from structural analysis to functional risk assessment under specific mission profiles.

Table 4.8

Configuration of model parameters for different operational modes (DxMI scenarios)

Operational mode	Scenario description	Attenuation α	Link weight modifier	Critical subsystems
Normal transit (M_1)	Standard cruising in calm waters. Low stress on maneuvering and auxiliary systems.	0.85 (High damping, localized impact)	1.0 (Standard)	Main engine, cooling
Maneuvering (M_2)	Port entry/exit or complex navigation. High interdependence of all systems.	0.95 (Low damping, wide propagation)	1.2 (Increased sensitivity)	Steering, thrusters, power distribution
Emergency (M_3)	Operation under partial failure or extreme weather. Maximum stress propagation.	1.0 (No damping, maximum cascade)	1.5 (Critical dependency)	Fire Safety, bilge pumps, emergency power

The initial model was formulated in JSON format and passed as input to a Python program. The program output a set of CSV tables and DOT-format files, which were batch-processed via the *make* utility into Graphviz. As a result, a set of diagrams in PNG format was produced. The analysis of the results was carried out using LibreOffice Calc. The implementation was completed in Python (version 3.11+) using the libraries *networkx*, *numpy/pandas*, and *graphviz*. The architecture is modular - the core simulation package can work autonomously or be integrated into a digital twin system via a REST/JSON API.

For experiment reproducibility, metadata storage is provided (node configuration, impulse parameters, model versions). For destructive and diagnostic impulses, two separate scripts were created - each containing a shared function set. Both programs include a main module, helper functions, and an impulse simulation module (DMI or DxMI depending on

purpose). The programs are designed to be launched either independently ("standalone" mode) or as part of more complex software systems (as Python libraries). The developed modeling software complex provides both numerical and graphical CSM results for CTSs of various classes and complexity.

Numerical experiments showed that when varying the damping coefficient α within the range 0.6–0.95, the vulnerability structure remains stable: a consistent "core of influence" is formed by a stable set of nodes. The average computational complexity of the model was $O(|V|+|E|)$, confirming its applicability to real CTSs (up to 10^4 nodes) using streaming data processing. To increase result reliability, a *reproducibility mode* is provided, fixing random weight generation and initial conditions.

The developed cognitive-simulation method provides numerical and visual assessment of structural and functional threats, identifying nodes that form the core of cascading impacts. The modeling results can be integrated into decision support systems and digital twins for adaptive CTS survivability management in real time.

4.3.3 Information–cognitive model of structural–functional interdependence of complex technical systems

Complex technical systems whose functioning is characterized by interdependence and interaction possess an increased risk of operational failures. For the reliable operation of such CTS, it is critically important to make timely control decisions aimed at forecasting and minimizing the negative consequences of emergency situations. For these purposes, a systemic approach can be applied in assessing the reliability of CTS operation, taking into account the interdependence and interaction of such systems. The foundation of the systemic approach may be the development and use of cognitive models, which make it possible to identify the contradictions of joint functioning of CTS, perform their qualitative analysis, and develop a structural diagram of cause-and-effect relationships of the key components of the studied technical system [69, 120, 121]. This makes it possible to develop an effective CTS management strategy. The first stage in the development of a cognitive model of technological interdependence and interaction of CTS should be the formulation of the goals (purpose) of the technical systems' operation.

For successful research, for example, the equipment of a vessel is represented as a set of systems consisting of interdependent and interconnected most critical CTS: the SPP; the ship power supply system (SPS); the ship auxiliary systems (SAS). In the course of the research, it is necessary to take into account the influence of the external environment (EE), understood as a combination of difficult-to-predict natural and non-

natural factors in emergency extreme situations. The main structural objects of the SPP are: the main engine (ME), the ship's propulsion device (PD), the shaft line (SL), as well as auxiliary mechanisms (AM). The SPS consists of sources of electric energy (SEE) in the form of direct and alternating current generators, batteries, cable networks (CN), switchboards (SB), and power converters (PC). The SAS represent a complex of various systems, the main of which are: bilge system (BS); ballast system (BAS); fire extinguishing – fire extinguishing complex (FEC); comfort air conditioning, technical air conditioning, and steam heating – climate control complex (CCC).

Each of the subsystems can be represented in the digital twin as a data object with the description of its structure, energy and information flows, as well as state metrics (temperature, vibration, load, voltage, etc.). This creates the basis for the automated formation of a cognitive map, in which the weights of the arcs are determined based on real operational correlations between parameters.

The assessment of structural and functional vulnerability relies on the metrics formalized in Section 4.3.1. To quantify these vulnerabilities explicitly:

A. Structural vulnerability index (*SV*): The *SV* index quantifies the ease with which a critical component (or the system as a whole) degrades due to impulse effects, based purely on its structural position and connection weights (Link_weight and α). *SV* is inversely proportional to the Integrated Operability Survivability Index (K_{SA}):

$$SV = 1 - K_{SA}, \tag{4.79}$$

where $SV \in [0,1]$. *SV*=1 implies complete collapse (zero survivability), and *SV*=0 implies full resilience

B. Functional vulnerability index (*FV*): The *FV* index measures the risk that the system's mission will fail under specific operational conditions, reflecting the impact of damage propagation on mission objectives. *FV* is inversely proportional to the structural stability index ($K_{SS}(t)$):

$$FV(t) = 1 - K_{SS}(t), \tag{4.80}$$

where $FV(t) \in [0,1]$. *FV(t)*=1 implies complete mission failure at time *t*, and *FV(t)*=0 implies full functional stability

For the selected composition of CTS, the structural diagram of interdependence and interaction of systems has the form shown in Fig. 4.30. In the cognitive model of CTS, the interaction and interconnections of the components located in the SPP, SPS, and SAS are taken into account (Fig. 4.31). In the development of the cognitive model, the DOT language was used, which serves for the description and construction of model graphs.

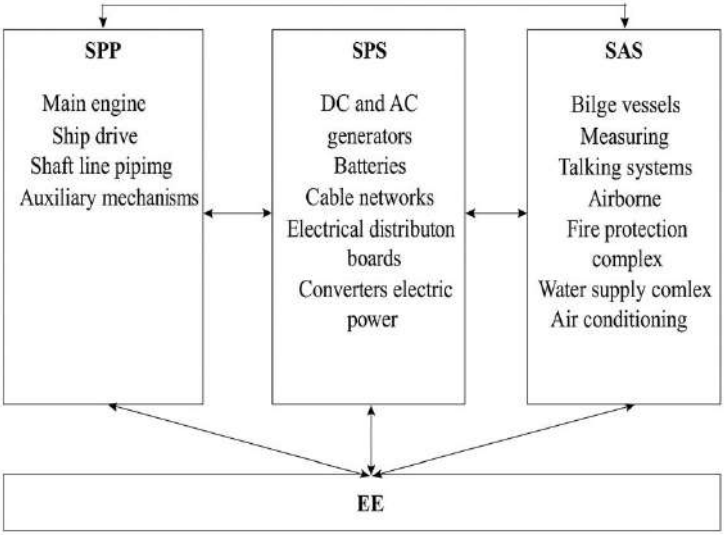

Figure 4.30. Structural scheme of interdependence and interconnection of CTS

The code was interpreted using the *Graphviz* software package. The modeling was carried out in accordance with the algorithm shown in Fig. 4.32. According to the cognitive map of interdependent and interconnected maritime technical systems (Fig. 4.33), the initial impact on the vessel is exerted by the external environment. During the experiments, the damaging impact impulse associated with environmental factors propagates toward the key elements of each subsystem. Traversing the corresponding links through all CTS aggregates, the damaging impulse reaches the Receiver Block (RB), which is intended for analyzing the operability state of each subsystem of the complex. In the modeling process, an additional approach was applied, consisting of the analytical or expert selection of the corresponding graph vertices to which the modeling impact is applied, as well as the moments in time when these impacts are introduced. The values

of the structural vulnerability coefficient by CTS aggregates, established during the experiments and calculations, are presented in the histogram in Fig. 4.34. The x-axis shows the names of all objects in the resulting informational cognitive model, while the y-axis represents the range of values taken by the structural vulnerability coefficient (from 0 to 1 with a step of 0.1).

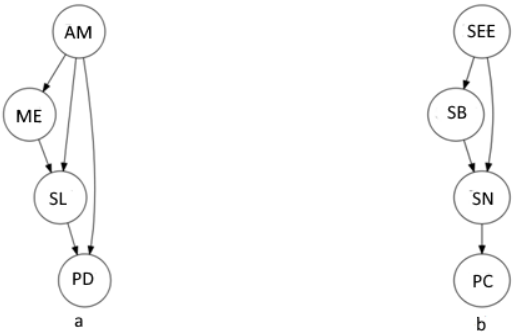

Figure 4.31. Interaction and interrelations of CTS model aggregates: a – interaction of objects of the SPP subsystem; b – interaction of objects of the SPS subsystem

The diagram (Fig. 4.32) presents the general sequence of vulnerability assessment.

In particular, the block "Simulation of disturbances and impulse transmission" can be formalized as follows:

```
For each node i in Graph:
    ReceiveImpulse(I, t)
    ΔF = ImpactModel(i, I, α)          # change of state
    F[i] = max(0, F[i] - ΔF)
    if F[i] < threshold_fail:
        S[i] = "failed"
    PropagateImpulse(i, I * w(i,j), t+1)
```

Here α is the attenuation coefficient, $w(i,j)$ is the weight of the connection between nodes, threshold_fail is the failure threshold of an element. Such an approach simplifies the implementation of cognitive connection simulation in software tools (Python + Graphviz + networkx).

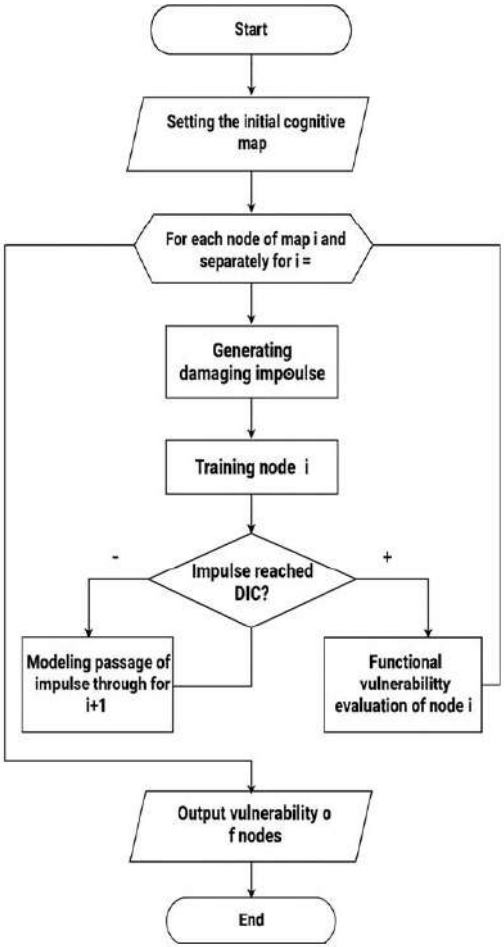

Figure 4.32. Algorithm for vulnerability assessment of nodes

The analysis of the interdependence graph showed that the vertices characterizing the main engine, power supply sources, and fire extinguishing complex have the highest structural vulnerability coefficient values (0.8–0.9). This confirms their key role in the formation of cascading effects and the justification for prioritizing their control. When integrating the model with digital twins of subsystems, it becomes possible to perform online assessment of structural vulnerability in real time, where the arc weights and node states are updated based on streaming data.

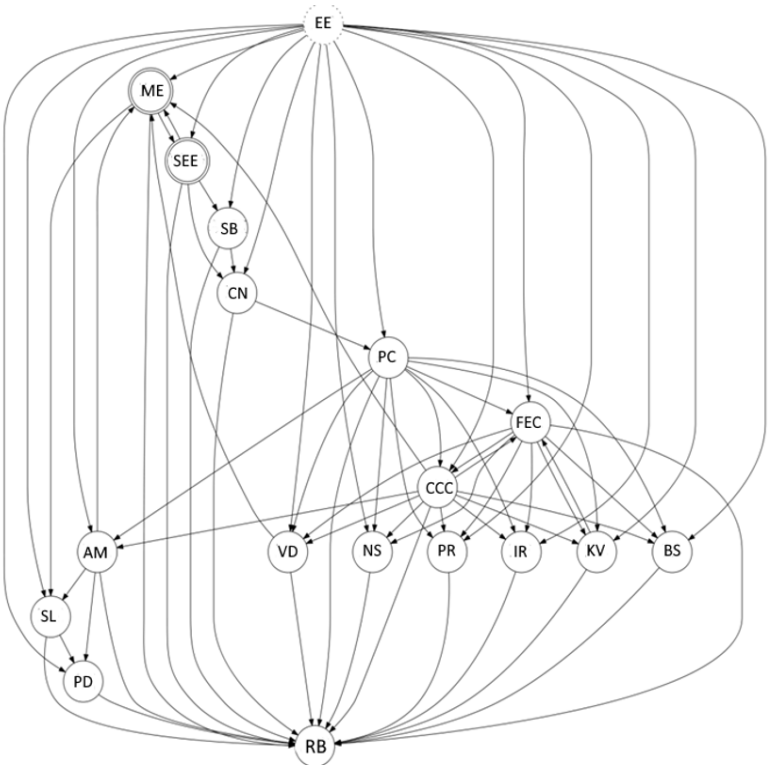

Figure 4.33. Cognitive map of interdependence and interaction of CTS

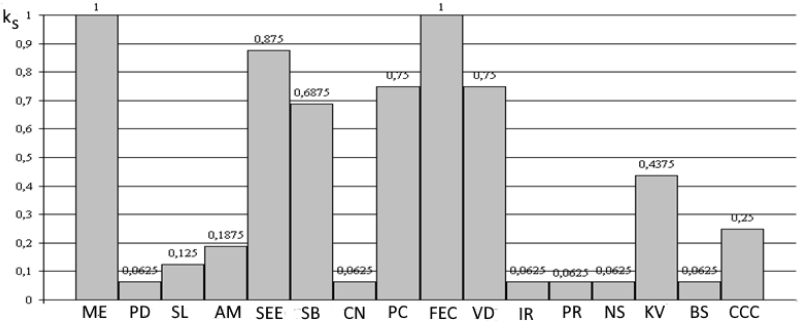

Figure 4.34. Values of structural vulnerability coefficients by CTS aggregates

To substantiate the conclusions regarding critical components, Table 4.9 presents the calculated *SV* and *FV* indices for the key subsystems of the refrigerated vessel example under two representative operational modes (M_1 and M_2).

Table 4.9

Calculated vulnerability indices for key CTS components

Subsystem	Abbreviation	Structural vulnerability	Functional vulnerability	Criticality rank
Main engine	ME	0.85	0.90 (M_1) / 0.70 (M_2)	High (vulnerable)
Fire extinguishing system	FEC	0.72	0.88 (M_1) / 0.80 (M_2)	High (vulnerable)
Power supply sources	SEE	0.65	0.80 (M_1) / 0.75 (M_2)	High (vulnerable)
Propulsion system	PS	0.50	0.35 (M_1) / 0.40 (M_2)	Moderate
Navigation system	NS	0.20	0.10 (M_1) / 0.05 (M_2)	Low

SV is calculated based on the maximum observed *SV* over T_{max} across all input impulse scenarios. *FV* is mission-specific and shows distinct differences between mode M_1 (e.g., Maneuvering) and mode M_2 (e.g., Cruising).

The data in Table 4.9 quantitatively confirms that the systems categorized as main engine (ME), fire extinguishing system (FEC), and power supply sources (SEE) exhibit the highest structural and functional vulnerability, primarily due to their high degree of involvement in mission-critical functions and central structural position within the cognitive map.

To provide a visual interpretation of why these components are critical nodes, Figure 4.35 illustrates the relative centrality measures of the key subsystems within the cognitive map.

According to the results of the conducted studies, the most vulnerable components in the CTS are the ME, the FEC, and the SEE, which is associated with their high degree of dependence on other CTS aggregates. The developed informational cognitive model of the CTS serves as the foundation of the designed DSS for assessing and ensuring the reliability of CTS operation. The use of informational cognitive models of CTS makes it possible to automate and optimize the process of reliability assessment of complex technical systems.

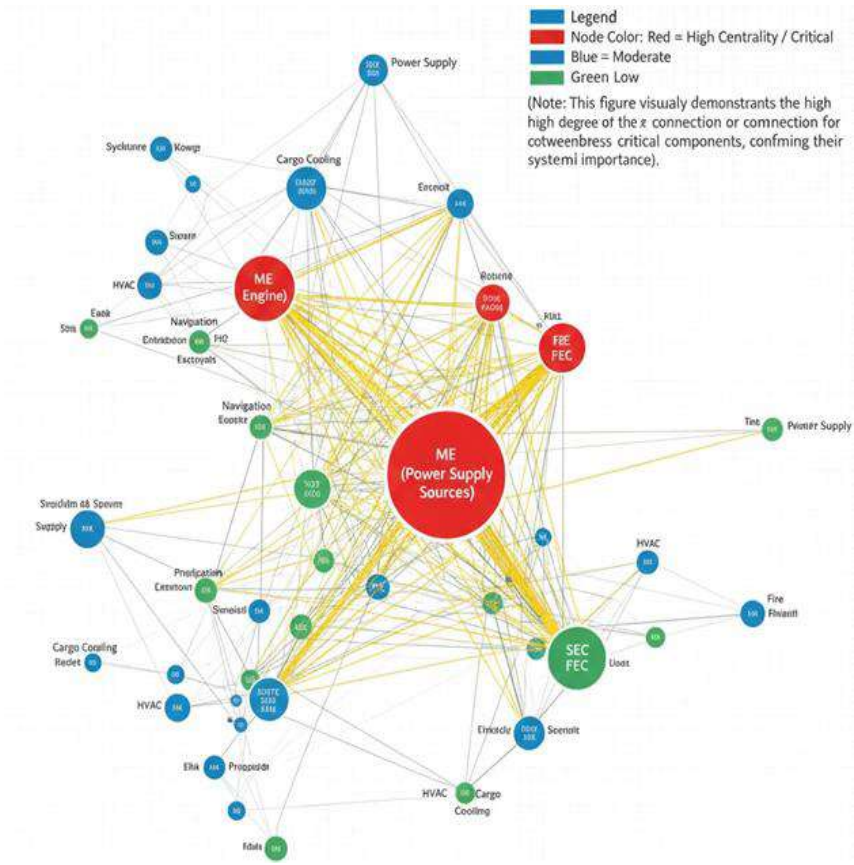

Figure 4.35. Cognitive map centrality analysis highlighting critical nodes (ME, FEC, SEE)

Thus, the informational cognitive model not only allows formalizing the interrelations between the elements of a complex technical system, but also quantitatively assessing the degree of their mutual influence under various operational scenarios. The obtained dependencies and mutual influence coefficients can be used as input data for constructing subsequent models - in particular, fuzzy risk assessment models considered in subsection 4.3.4. This ensures continuity between the cognitive–simulation and logical–probabilistic approaches within the overall methodology for survivability analysis of CTS.

4.3.4 Fuzzy models for analyzing and assessing the risk of failures in complex technical systems

The objective of this subsection is to develop an integrated fuzzy model that provides a comprehensive assessment and prediction of the failure risk of the SPP under conditions of high uncertainty. This objective is achieved through the fusion of heterogeneous sources of information: the cognitive–simulation model, the DT [122], and neural network subsystems. Modern CTS, including SPP, operate under conditions of high uncertainty and interdependence of components. Incomplete diagnostic data, sensor noise, and variability of external factors necessitate the use of fuzzy models for assessing the risk of failures and the consequences of their cascading propagation [123]. Within the CSM (see Sections 4.3.2–4.3.3), fuzzy logic is applied as an interpretive layer that transforms quantitative values of probabilities and damages into linguistic assessments of the risk level. Such integration increases the robustness of the forecasting system when operating with incomplete or inaccurate digital twin data.

To integrate cognitive, fuzzy, and digital components, an architecture is proposed that combines the CSM, the SPP digital twin, and a hierarchical system of fuzzy logic modules for equipment failure risk assessment (Fig. 4.35).

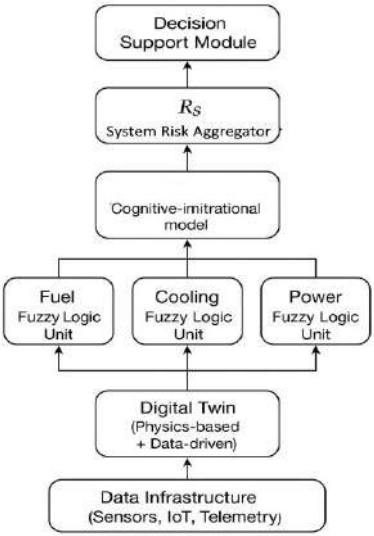

Figure 4.35. Integrated architecture for risk assessment of a CTS based on a DT, CSM, and fuzzy logic

At the lowest level lies the data infrastructure - a network of sensors, IoT devices, and telemetry streams that ensure continuous collection of operational information on equipment parameters (pressure, temperature, vibration, load, etc.).

The digital twin layer includes two interrelated components:

- physics-based model, which provides a physico-mathematical representation of thermodynamic, hydrodynamic, and electrical processes. This model delivers physical–statistical estimates of equipment degradation and the magnitude of potential damage $Y_i^{DT}(t)$;

- data-driven model, based on recurrent neural networks (LSTM/GRU) trained on historical and real-time time series. Its function is to extract latent features from telemetry streams $\phi_i^{NN}(t)$, enabling the prediction of deviations and the detection of hidden anomalies before they manifest in explicit physical parameters [124].

The CSM within the structure of the integrated risk assessment architecture of a complex technical system is located above the DT and performs the functions of structural and causal analysis [125]. It describes the interdependencies between subsystems and the cause–effect chains that may lead to failures. The CSM uniquely serves a dual purpose: first, calculating element failure probabilities $P_i^{CSM}(t)$) through simulation; and second, providing the structural criticality coefficients (W_i) necessary for system-level risk aggregation. The CSM computes failure probabilities ($P_i^{CSM}(t)$) for CTS elements using simulation or Monte Carlo runs to model cascading effects. Such a combination of models creates a powerful compensatory structure. If the data-driven model detects an early but latent anomaly signal (ϕ_i^{NN}), the fuzzy aggregator can account for this factor even when the physical parameters Y_i^{DT} remain within nominal limits. Conversely, if the sensor data are unreliable, the structural logic P_i^{CSM} from the CSM ensures a conservative risk estimate, preventing falsely low results based on inaccurate telemetry. In Fig. 4.35, the CSM transmits probabilistic estimates upward in the hierarchy to the fuzzy modules; the digital twin provides actual indicators of technical condition; the neural network generates dynamic latent characteristics; the fuzzy aggregator computes R_i. Within this integrated architecture, the fuzzy model serves as a linking layer between the CSM, the DT, and the neural network-based forecasting components. Crucially, the system utilizes the «Adaptive Neuro-Fuzzy

Inference System (ANFIS)№ approach, extended to handle multiple input channels. The choice of ANFIS, rather than a static fuzzy logic system, is driven by the necessity for «continuous self-learning and automatic adaptation№ of membership functions and inference rules based on incoming DT data, which is essential in the dynamically changing operational environment of the SPP. Furthermore, the CSM fulfills a dual function: it provides the failure probability $P_i^{CSM}(t)$ as a primary input to ANFIS for calculating local risk R_i, and it supplies the structural criticality coefficient W_i for subsequent global aggregation of system risk R_s.

The R_i values flow into the R_s aggregator, forming the integral system-level risk, which is then used by the Decision Support Module to warn operators and predict failures. This integration of cognitive, physical, and intelligent layers enables continuous self-learning and adaptive risk assessment of a complex technical system, including the SPP, in real-time operation.

The next level is formed by fuzzy risk assessment modules (Fuzzy Logic Units), implemented for individual SPP subsystems - fuel system, cooling system, power supply system, etc. Each module generates a partial risk estimate R_i based on the condition parameters of the corresponding subsystem.

The partial risk values are then fed into the system risk aggregator R_S, where an integral indicator is formed and used in the DSS. The methodology for computing the integral system risk R_s is critically important, since the SPP is subject to cascading failures. A simple aggregation (e.g., taking the maximum value $max(R_i)$) may make the system hypersensitive to local peaks. Therefore, when aggregating R_s, it is necessary to account for the structural criticality of elements as defined by the CSM. Weighting the partial risks R_i with respect to the structural importance of each element appears to be the most logical step for further formalization of the system-level indicator. Within the integrated architecture (Fig. 4.35), the fuzzy model serves as a linking layer between the CSM, the digital twin, and the neural network–based forecasting components. It is not an independent element but functions as a superstructure above the cognitive model, ensuring the fusion of structural dependencies, probabilistic assessments, and real diagnostic data.

The SPP digital twin serves as a source of up-to-date information for updating CSM parameters and training adaptive rules in the ANFIS module. The system architecture forms a closed-loop adaptive risk assessment circuit that implements an end-to-end information processing chain: DT data → CSM (structural influence) → fuzzy system (risk assessment) → DSS. Such integration ensures the fusion of structural dependencies, probabilistic assessments, and real diagnostic data, acting as an overarching layer above the cognitive model [126]. Each of these levels provides its own set of parameters characterizing the current state and behavior of CTS components:

- the CSM produces probabilistic–causal estimates - the probability of failure of element i at time ttt, calculated based on simulation of cascading effects;

- the digital twin provides physico-mathematical and data-driven assessment of the current degradation level using thermodynamic, vibration, pressure, flow, and other parameters;

- the neural network subsystem (NN) extracts latent features from telemetry streams, reflecting the dynamic behavior of equipment components;

- the confidence coefficient ci determines the weight of each data source in the aggregated risk assessment and is involved in calculating the integral indicator RS. The confidence coefficient ci serves as a dynamic metric of operational observability. In conditions of deteriorating data quality (for example, increased telemetry gaps or sensor malfunction), an automatic reduction of ci decreases the influence of unreliable data on the final risk estimate R_i. This allows the DSS to temporarily rely more heavily on conservative, structurally accurate estimates from the CSM until telemetry quality is restored.

The partial risk estimate R_i for element i is calculated as a generalized function of four principal metrics:

$$R_i(t) = f(P_i^{CSM}(t), Y_i^{DT}(t), \phi_i^{NN}(t), c_i), \tag{4.81}$$

where $R_i \in [0,1]$ - the resulting failure risk of subsystem *i*, computed by the fuzzy aggregator;

$R_i(t)$ - the integral (generalized) failure risk for element/node *i* at time *t*, type: dimensionless value within the range [0, 1] (0 - no risk, 1 - critical probability/impact), interpretation: a value used in the DSS for prioritizing response actions and decision-making;

$P_i^{CSM}(t)$ - failure probability estimate of element *i*, obtained from the CSM at time *t*. Type / range: $P_i^{CSM}(t) \in [0, 1]$. Source and method of acquisition: result of simulating cascading damaging impulses and causal chains in the CSM; may be computed as the frequency of activations / probability of transition into failure state during Monte Carlo / simulation runs. Update frequency: according to simulation schedule (e.g., at each CSM re-run) or upon arrival of significant events. Probabilistic estimates are updated event-driven or during planned CSM recalculation;

$Y_i^{DT}(t)$ - assessment of the severity/impact of failure for element *i*, generated by the DT at time *t*. Type / range: normalized value $Y_i^{DT}(t) \in [0, 1]$, where 0 - no damage, 1 - maximum (critical) damage; if needed, physical units (kW, °C, etc.) may be preserved and then normalized to [0, 1]. Source and method of acquisition: physics-based computations (thermotechnical, vibration, hydrodynamic models) and/or physico-statistical assessments of the digital twin; may include predicted damage accumulation (RUL → normalization). Physical degradation parameters are updated in streaming mode (1–10 s);

$\phi_i^{NN}(t)$ - estimate/index output by the neural network subsystem (data-driven component), associated with element *i* at time *t*. Possible interpretations: predicted failure probability from ML (e.g., NN output as probability), anomaly score, predicted remaining useful life (RUL) after normalization. Type / range: for consistency with other inputs, $\phi_i^{NN}(t)$ is recommended to be normalized to [0, 1]. Source: trained recurrent/convolutional/hybrid networks trained on historical and streaming data; updated based on online parameters and periodic retraining. To preserve a high degree of interpretability of the hybrid model, the neural network index $\phi_i^{NN}(t)$ must be formalized using linguistic terms consistent with the categories “Probability” and “Impact.” For the normalized value $\phi_i^{NN}(t)$, it is proposed to use three linguistic terms: “No Anomaly,” “Moderate Anomaly,” and “Critical Anomaly.” These terms are fuzzified using triangular membership functions structurally similar to $P_i^{CSM}(t)$, but reflecting the neural network’s confidence in the presence of a latent defect.

The corresponding ranges must be defined based on statistical analysis of the distribution of $\phi_i^{NN}(t)$ during normal operation and early failure precursors. Hidden features are updated adaptively via online model learning. The index $\phi_i^{NN}(t)$ is specifically designed as an Anomaly Score derived from the reconstruction error of the LSTM autoencoder within the data-driven DT. This score represents the degree of deviation of the current telemetry pattern from the statistically normal operating mode. Normalization to [0, 1] is performed using a Sigmoid function calibrated to historical anomaly thresholds, ensuring that values closer to 1 correspond to a high confidence in the presence of a critical, latent defect not yet visible in the physical parameters Y_i^{DT};

c_i - confidence / credibility weight for element i. Type / range: $c_i \in [0, 1]$. The value reflects trust in the input data/models (e.g., low when telemetry is incomplete or has a high proportion of missing values, high when measurements are verified). Application: used as a weight in aggregating data sources. The confidence coefficient is updated on a scheduled basis (every 24 h);

$\varphi(\cdot)$ - aggregation function / fuzzy inference function (fuzzy aggregation /Takagi–Sugeno style), implementing the rule for combining the three information sources and the confidence weight. $\varphi(\cdot)$ is the fuzzy inference function implementing the Takagi–Sugeno rule for aggregating probabilistic, physical, and neural assessments.

The function $f(\cdot)$ is implemented as a generalized fuzzy inference function which utilizes the Takagi-Sugeno model structure. This specific choice of implementation is motivated by the need for clear interpretability, computational efficiency, and, crucially, the ability to smoothly integrate the heterogeneous inputs and the dynamic confidence coefficient c_i. Therefore, $R_i(t)$ is mathematically defined as:

$$R_i(t) = \varphi(P_i^{CSM}(t), Y_i^{DT}(t), \phi_i^{NN}(t), c_i) \tag{4.82}$$

This formal definition clarifies that the fuzzy aggregator $\varphi(\cdot)$ is the practical realization of the function $f(\cdot)$, where the structural information from the CSM, the physical degradation from the DT, and the latent anomaly detection from the NN are fused based on their assessed data quality (c_i).

Analysis of Input Metrics and Asynchrony: A key aspect of the system is the necessity to aggregate data with different degrees of timeliness and update modes (Table 4.10). In generalized form, the sources of parameters, their types and update modes are summarized in Table 4.10, which illustrates how data from different model levels are merged into a unified risk field.

In generalized form, the sources of parameters, their types and update modes are summarized in Table 4.10, which shows how data from different levels of the model are integrated into a unified risk landscape.

Table 4.10

Sources of parameters for the integrated fuzzy model

Parameter source	Data source	Data type	Update	Update interval
Cognitive-simulation model	probabilistic scenario assessments	event-driven recalculation	on state change	
Digital twin (physics / data-driven)	physical degradation parameters	streaming mode	1–10 s	Digital twin (physics / data-driven)
Neural network model (LSTM/GRU)	latent features from sensor time series	online learning	adaptive	Neural network model (LSTM/GRU)
expert or automatic	source trust weight	scheduled re-evaluation	24 h	

Integration of parameters from the cognitive model, the digital twin, and the neural network subsystem (Table 4.10) enables the formation of an adaptive feature space, where each data source contributes to the final risk estimate $R_i(t)$. Fuzzy logic, which operates on degrees of membership, is ideally suited for managing this asynchrony, as it is less sensitive to short-term delays or temporal shifts than strict probabilistic methods, and allows each data source to contribute to the final risk estimate $R_i(t)$.

The confidence coefficient ci serves as a dynamic metric of operational observability. In conditions of data quality degradation (for example, increased telemetry gaps or sensor malfunction), an automatic reduction of ci decreases the impact of unreliable data on the final risk estimate R_i. This enables the DSS to temporarily rely more on conservative, structurally valid estimates produced by the CSM until telemetry quality is restored. To implement the fuzzy inference function $\phi(\cdot)$, a Takagi–Sugeno (TS) model [127] is used, which has been extended to handle three primary input sources (probability, impact, and neural index). The choice of the TS model is driven by its ability to efficiently and smoothly approximate complex nonlinear functions, which is essential for real-time risk prediction.

Formalization of inference: the resulting scalar risk $R(t)$ is calculated as a weighted sum of the consequents r_k using the Takagi-Sugeno model, where the activation degree of each rule, $\mu_k(\cdot)$, is explicitly adjusted by the confidence coefficient c_i. The coefficient c_i ensures robustness by reducing the influence of unreliable or statistically anomalous data sources.

$$R_i(t) = \frac{\sum_{k=1}^{m} \mu_k(P_i(t), Y_i^{DT}(t), \phi_i^{NN}(t), c_i) \cdot r_k}{\sum_{k=1}^{m} \mu_k(P_i^{CSM}, Y_i^{DT}(t), \phi_i^{NN}(t), c_i)}, \tag{4.83}$$

where $\mu_k(\cdot)$ is the activation degree of the k-th rule of the knowledge base;

r_k is the result of defuzzification for this rule;

m is the number of fuzzy rules

Thus, each module evaluates the subsystem risk in real time, simultaneously using the cause-effect relationships of the CSM, the physical regularities of the digital twin, and the empirical dependencies discovered by the neural network.

For the input variables “Failure probability” (P) and “Failure impact” (Y), three linguistic terms are defined, using triangular-type membership functions.

The membership functions presented in Table 4.11 specify the shape and boundaries of the linguistic terms of the variable “Failure probability.”

Table 4.11

Membership functions of the variable "Failure probability"

Term	Function type	Range
Low	triangular	[0 ; 0.4]
Medium	triangular	[0.2 ; 0.7]
High	triangular	[0.5 ; 1.0]

Table 4.8 describes the membership functions of the variable "Impact." Unlike failure probability, the term ranges here are shifted toward higher values, which reflects the system's increased sensitivity to rising impact.

The analysis of term ranges in Tables 4.11 and 4.12 demonstrates the intentional encoding of expert policy.

The terms for the variable "Impact" (e.g., "Medium," beginning at 0.3) are shifted toward higher values compared to the terms for "Failure Probability" ("Medium," beginning at 0.2). This shift reflects the principle of risk aversion: the system assigns greater importance to the severity of potential consequences rather than solely to the frequency of failure occurrence, which is critically important for assessing the reliability of complex technical systems.

Table 4.12

Membership functions of the variable "Impact"

Term	Function type	Range
Low	triangular	[0; 0.5]
Medium	triangular	[0.3; 0.8]
Significant	triangular	[0.6; 1.0]

The analysis of the term ranges in Tables 4.11 and 4.12 demonstrates the intentional encoding of expert policy. The terms for the variable "Impact" (for example, "Medium," starting from 0.3) are shifted toward higher values compared to the terms for "Failure Probability" ("Medium," starting from 0.2). This shift reflects the principle of risk aversion: the system assigns greater importance to the severity of potential consequences (Severity) rather than solely to the frequency of failure occurrence, which is critically important for evaluating the reliability of complex technical systems.

To enable multi-channel risk aggregation, the third key input parameter $\phi_i^{NN}(t)$ - (anomaly index extracted by the neural network) - is also fuzzified. For the normalized value $\phi_i^{NN}(t)$, three linguistic terms are defined using triangular membership functions, as is the case for *P* and *Y*. These terms, reflecting the degree of latent anomaly in the data, are presented in Table 4.13.

Table 4.13

Membership functions of the variable "Neural network index"

Term	Function type	Range
No Anomaly	triangular	[0; 0.4]
Moderate Anomaly	triangular	[0.2; 0.8]
Critical Anomaly	triangular	[0.6; 1.0]

The corresponding graphs of the membership functions "No anomaly," "Moderate anomaly," and "Critical anomaly" are shown in Figure 4.36. The visualization of this third input completes the graphical representation of all variables used to activate the fuzzy rules.

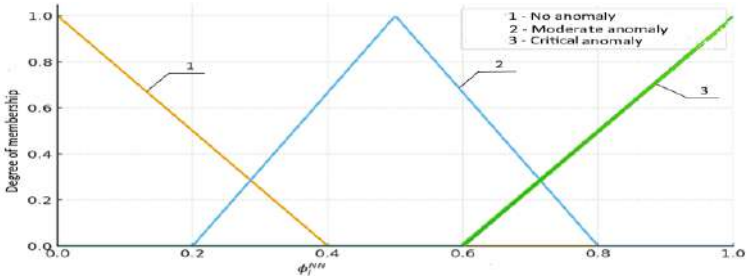

Figure 4.36. Membership functions of the variable "Neural Network Index."

To visualize these dependencies, Figure 4.37 shows the triangular membership functions for the three terms: Low, Medium, and High. Each function characterizes the degree of membership of a failure probability value to the corresponding category and is used at the fuzzification stage of input data when calculating partial subsystem risks. The clear partitioning of the [0;1] range ensures a smooth transition between risk levels as P_i varies.

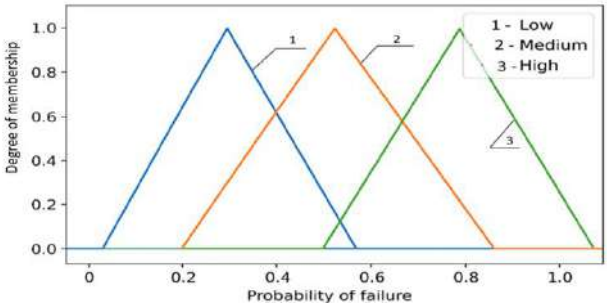

Figure 4.37. Membership functions of the variable "Failure probability" (Low, Medium, High)

Figure 4.37 presents the graphs of the membership functions “Low,” “Medium,” and “High,” which define triangular distributions over the interval [0;1]. These functions are used in fuzzy rules together with the failure probability variable, forming the basis of the “IF–THEN” rule base for calculating the integral risk R_i.

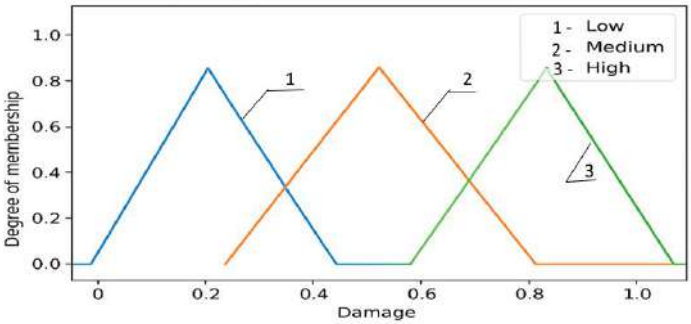

Figure 4.38. Membership functions of the variable “Damage”

Figure 4.38 presents the graphs of the membership functions "Low," "Medium," and "Significant" (or "High"), which define triangular distributions over the interval [0;1]. These functions characterize the degree of membership of a damage value (Y) to the corresponding linguistic category. The visual characteristics of the functions, such as the shifting of term ranges toward higher values (as detailed in Table 4.8), reflect the system's principle of risk aversion, assigning greater weight to severe consequences rather than low-impact events. These fuzzified values of the Damage variable are essential inputs for the "IF-THEN" rule base in calculating the integral risk R_i.

The rule base is formed either by expert definition or based on data from a digital twin. An example fragment is shown in Table 4.14.

Table 4.14

Example of fuzzy inference rules

№	Condition	Linguistic consequent	Result (R_i)
1	IF (P =) Low AND (Y =) Low THEN (R =) Low	Very low	0.1
2	IF (P =) Medium AND (Y =) Low THEN (R =) Medium	Moderate	0.4
3	IF (P =) High AND (Y =) Medium THEN (R =) High	High	0.7
4	IF (P =) High AND (Y =) High THEN (R =) Very High	Very high	0.9

The linguistic consequents used in the rule base are formalized with the following scalar output values, corresponding to the Takagi-Sugeno output coefficients r_k: Very Low: 0.1; Moderate: 0.4; High: 0.7; Very High: 0.9.

To transform the linguistic assessment of risk into a scalar value R_i, the Centroid method is applied. The defuzzification surface $R=f(P,Y)$ presented further (Figure 4.39) specifically visualizes the *partial* subsystem risk R_i prior to system-level aggregation.

$$R_i = \frac{\int \mu_R \sum_{k=1}^{m} \mu_R(x_i) x dx}{\int \mu_R(x) dx}, \tag{4.84}$$

This method ensures a smooth transition of the output values R_i between linguistic terms, maintaining the stability and reliability of the input signal for the DSS. It is applied to generate the risk surface for both the basic (expert) model and its adaptive version, with differences between the surfaces determined by the parameters of the membership functions refined during the training process.

The basic defuzzification surface $R=f(P,Y)$, constructed from expert rules (Table 4.10), clearly demonstrates the nonlinear growth of the failure risk of the SPP. This surface confirms the system's strong sensitivity to the consequences of failures: even with a moderate failure probability, an increase in damage leads to a sharp rise in the final risk value.

Figure 4.39 presents the defuzzification surface $R=f(P,Y)$ obtained for the basic fuzzy model, illustrating how the integral failure risk of the SPP changes with various combinations of failure probability and damage level. Warmer shades (or higher Z-axis values) correspond to areas of elevated risk.

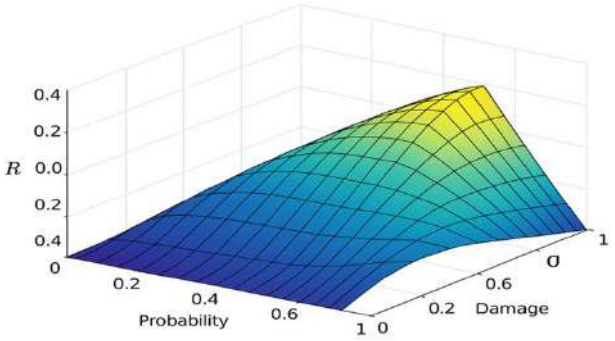

Figure 4.39. Defuzzification surface of the SPP failure risk $R=f(P,Y)$ for the basic model.

From the presented surface, it can be seen that the SPP failure risk *R* increases nonlinearly: even with a moderate failure probability, an increase in damage leads to a sharp rise in the final value. Such a surface shape indicates the system's strong sensitivity to the consequences of SPP failures, which is important when optimizing maintenance priorities. Furthermore, the third input, the Neural Network Anomaly Index $\phi_i^{NN}(t)$, acts as a dynamic modifier of the risk surface. As $\phi_i^{NN}(t)$ transitions from "No Anomaly" to "Critical Anomaly," the entire defuzzification surface *R=f(P,Y)* shifts upward, becoming steeper in high-risk zones. This dynamic sensitivity ensures that latent, early-stage anomalies (detected only by the NN) instantly trigger an increase in the integral risk R_i, even when *P* and *Y* remain moderate.

The membership functions in Figure 4.36, together with those for "Failure Probability" (Figure 4.37) and "Damage" (Figure 4.38), are used to fuzzify all three key input variables in the fuzzy module. These fuzzified values form the basis of the "IF–THEN" Rule Base, which is essential for calculating the partial integral risk R_i.

Since the SPP is a complex technical system subject to cascading failures, the system risk R_s cannot be determined as a simple sum or maximum of the partial risks R_i. To take into account the structural criticality of the elements, as defined by the CSM, it is proposed to use the following weighted aggregation formula:

$$R_S = \sum_{i=1}^{N} W_i \cdot R_i, \tag{4.85}$$

where R_i - partial risk of subsystem *i*;

W_i - coefficient of structural criticality of element *i*;

N - total number of subsystems

The coefficient W_i must be obtained from the CSM and should reflect the potential impact of the failure of element *i* on the overall operability of the SPP (for example, based on an analysis of causal chains). The use of structural weights ensures that the decisions generated by the DSS prioritize those elements whose failure produces the greatest cascading effect, even if their current partial risk R_i is not the highest.

For the calculation of the structural criticality coefficient W_i, a graph-theoretic approach is applied, leveraging the causal relationships established in the CSM. Specifically, W_i is defined as a function of the cascading

impact potential C_i, normalized to ensure $\sum_{i=1}^{N} W_i = 1$. The C_i metric is determined by analyzing the frequency and severity of system-level failures (loss of generation/cooling capacity) triggered by a simulated failure of element *i*. The mathematical basis for C_i is derived from the betweenness centrality of element *i* within the structural dependency graph, weighted by the severity of the ultimate consequences of its failure. This formalization ensures that W_i is a robust, quantitative measure directly linked to the structural integrity and operational criticality of the SPP.

The completed diagram (Figure 4.40) demonstrates how the integral system risk R_s is formed through the weighted aggregation of partial risks R_i. Accounting for structural coefficients W_i makes it possible to avoid hypersensitivity to local risk peaks and guarantees that the final value of R_s reflects the actual potential for cascading failures. The resulting integrated indicator R_s is then transmitted directly to the Decision Support Module, providing DSS operators with a comprehensive indicator for making timely maintenance and reliability management decisions for the SPP in real time.

The process of weighted aggregation of the system risk R_s is illustrated in Figure 4.40. The diagram emphasizes that the CSM serves as the source not only of probabilistic assessments P_i but also of critically important structural weights W_i.

To ensure continuous model updating and computational stability, a Takagi–Sugeno fuzzy inference system is employed, extended to a multichannel configuration. At this level, both the defuzzification of output values and the automatic tuning of the membership function parameters are implemented using the hybrid ANFIS (adaptive neuro-fuzzy inference system) approach.

Based on the structural diagram (Figure 4.40), an adaptive ANFIS model has been developed, in which the parameter learning process is tightly integrated with the data of the SPP digital twin. This integration enables continuous model adaptation under changing operational conditions and variations in equipment state parameters.

The use of the ANFIS system allows combining the interpretability of fuzzy logic with the learning capability of neural networks, which is especially important for dynamically changing environments of complex technical systems. Within the digital twin framework, the ANFIS model operates as a trainable risk assessment module that processes incoming telemetry data in real time, refines the parameters of membership functions, and increases the accuracy of failure risk prediction.

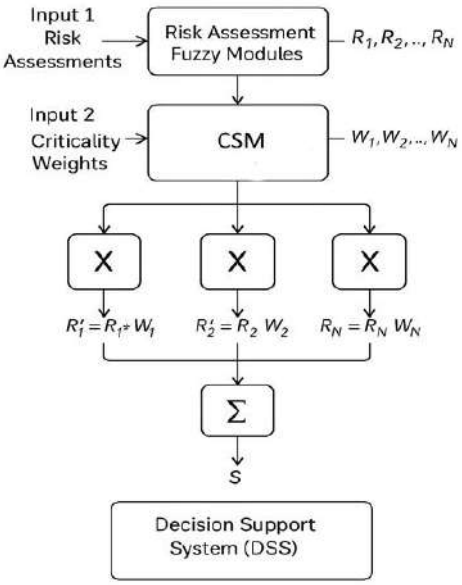

Figure 4.40. Diagram of weighted aggregation of system risk R_s using structural criticality coefficients W_i

Thus, the hybrid ANFIS structure not only enhances the precision and adaptability of risk assessment but also provides a foundation for forming self-learning intelligent subsystems within the SPP digital twin, capable of autonomously refining their models in response to changes in the system's condition and external factors.

In ANFIS, the output risk R_i is represented as follows:

$$R_i(P_i, Y_i, \phi_i) = \sum_{k=1}^{m} \omega_k \cdot f_k(P_i, Y_i, \phi_i), \tag{4.86}$$

where ω_k are the rule weights trained during the adaptation process, and $f_k(\cdot_i)$ are the linear functions of inputs for rule *k*.

Unlike static expert rules (Table 4.10), the ANFIS structure allows for the dynamic adjustment of rule consequents and antecedents based on training data. More importantly, the system can dynamically modulate the influence of latent features ($\phi_i^{NN}(t)$) and the structural baseline ($P_i^{CSM}(t)$).

An example of an adaptive ANFIS rule that incorporates the third input, $\phi_i^{NN}(t)$, and the dynamic coefficient c_i is:

Rule Fragment (Adaptive ANFIS Rule)

$$IF(P_i^{CSM}(t)\,ISLow)\,AND(Y_i^{DT}(t)\,ISMedium)\,AND(\phi_i^{NN}(t)\,ISCriticalAnomaly)$$

$$THENR_iISf_{high} = a_k \cdot P_i + b_k \cdot Y_i + c_k \cdot \phi_i + d_k, \tag{4.87}$$

where a_k, b_k, c_k, d_k are the linear coefficients of the Takagi-Sugeno consequent function, which are learned adaptively. Crucially, the activation degree of this rule, μ_k, is weighted by the confidence coefficient c_i:

$$\mu_{k,adjusted} = \mu_k \cdot c_i(t)$$

This ensures that if the input data quality is low ($c_i \to 0$), the activation of even a critical rule (e.g., detecting a "Critical Anomaly" via NN) is suppressed, forcing the system to rely on conservative estimates. This demonstrates the superior adaptive capability of ANFIS compared to the static FL model.

Training of the membership function parameters $\theta=\{a,b,c\}$ and the weights $\{w_k\}$ is performed using the gradient method or the Levenberg–Marquardt algorithm according to the criterion of minimizing the MSE:

$$E = \frac{1}{2N}\sum_{i=1}^{N}(R_i^{\exp} - R_i^{model}), \tag{4.88}$$

where $R_i^{\exp}$ are the actual (or simulated by the DT) risk values;

R_i^{model} are the model estimates;

N is the number of observations

This structure creates a continuous self-learning mechanism, where the Digital Twin provides both input data and the “real” or simulated risk values for model adjustment, ensuring adaptation to the current operating modes of the SPP.

Critical requirements for validation (Ground Truth). During ANFIS model training, the key limitation is the accuracy of R, used as the true risk function. Since, under real operational conditions, the available data on SES failures may be insufficient, it often becomes necessary to train the ANFIS on simulation data generated by the DT. This dependency carries a potential risk: systematic errors inherent in physics-based or data-driven Digital Twin

models can be transferred to and amplified within the adaptive fuzzy system. Therefore, especially rigorous verification and validation procedures are required for the DT as a generator of training data (Ground Truth) to ensure that *R* accurately reflects the real, not simulated, risk.

A comparative analysis of the adaptive ANFIS model and the classical static fuzzy logic (FL) model (based on expert rules) demonstrates a significant improvement in the efficiency of the hybrid approach. To evaluate the effectiveness of the adaptive scheme, its characteristics were compared with those of the classical fuzzy model, which relies solely on expert rules.The main differences are summarized in Table 4.15.

Table 4.15

Comparison of classical and adaptive fuzzy models

Criterion	Classical FL model	ANFIS model
Source of parameters	Expert estimations	Training on DT data
Accuracy (MAE)	0.08 – 0.12	0.03 – 0.05
Computation time	< 0.1 s	0.15 – 0.2 s
Self-learning capability	No	Yes
Implementation	MATLAB / Python (SciKit-Fuzzy)	Python (TensorFlow + FuzzyLite)

From Table 4.15, it can be seen that the adaptive ANFIS model demonstrates higher accuracy with only a slight increase in computation time, which makes it suitable for online risk assessment tasks within the digital twin of the SPP. Unlike the static FL model, the adaptive module is capable of updating its parameters as new diagnostic data arrive, which ensures self-learning and system robustness under uncertainty. The ANFIS system shows a radical improvement in accuracy: the Mean Absolute Error (MAE) decreases by 2–4 times, reaching the range of 0.03–0.05. This represents a significant enhancement in the reliability of forecasting. At the same time, the increase in computation time to 0.15–0.2 s is minor and quite acceptable for online risk assessment tasks, considering that the most frequently updated DT data arrive at intervals of 1–10 seconds.

After the training phase, ANFIS forms an adaptive defuzzification surface $R=f(P,Y)$ (Fig. 4.41), which differs from the baseline expert surface (Fig. 4.39). The surface becomes smoother and more nonlinear, reflecting the refinement of boundaries between linguistic terms based on actual operational data. The greatest risk gradients are observed in the transition zones from moderate to critical states. This ensures increased model sensitivity to abnormal operating modes and confirms that ANFIS has

adapted to the complex nonlinear dependencies typical of real SPP operating conditions, which could not be accurately encoded in general expert rules.

Figure 4.41 shows the defuzzification surface of risk *R=f(P,Y)* after the adaptive ANFIS model training. The surface represents the dependence of the integrated risk on the failure probability *P* and damage magnitude *Y* for the adaptive neuro-fuzzy model after the training stage on digital twin data.

Unlike the baseline model (Fig. 4.39), the surface exhibits smoother transitions and nonlinear regions, reflecting the refinement of boundaries between linguistic terms and increased sensitivity to high-risk zones. Unlike the baseline fuzzy model shown in Fig. 4.41, the adaptive ANFIS system forms the risk surface *R=f(P,Y)* considering the current operational data and results of the digital twin. During training, the parameters of the membership functions are refined, leading to a change in the shape of the defuzzification surface - it becomes smoother, reflecting the real dependencies between failure probability and damage. The greatest gradients are observed in the transition zones from moderate to critical states, which ensures increased model sensitivity to abnormal operating modes of equipment.

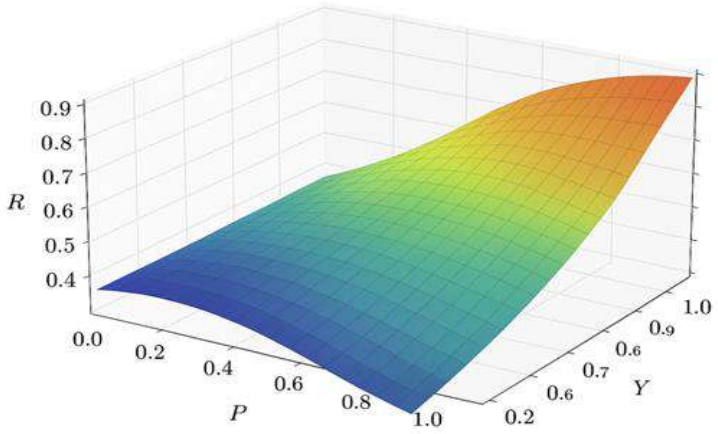

Figure 4.41. Defuzzification surface of risk *R=f(P,Y)* after training of the adaptive ANFIS model

Figure 4.42 illustrates the result of increasing the model's granularity from three (baseline) to five linguistic terms for the input variables "Failure Probability" (*P*) and "Damage" (*Y*). This refinement results in a more detailed approximation of the complex, nonlinear risk function, which is evident in two key aspects:

1. Reduced smoothness in critical zones: unlike the smoother baseline surface (Figure 4.39), the 5-term surface exhibits sharper, more defined gradients, particularly in the transition zone where $P > 0.6$ and $Y > 0.7$. This reflects a higher fidelity to the expert policy in high-risk regions.

2. Increased transition zones: the introduction of two extra terms ("Very low risk" and "Very high risk") enables the model to create more nuanced maintenance decision boundaries.

This visualization serves as the primary justification for the recommended improvement in model granularity, demonstrating how increased detail enhances the accuracy of maintenance decision-making in critical risk areas.

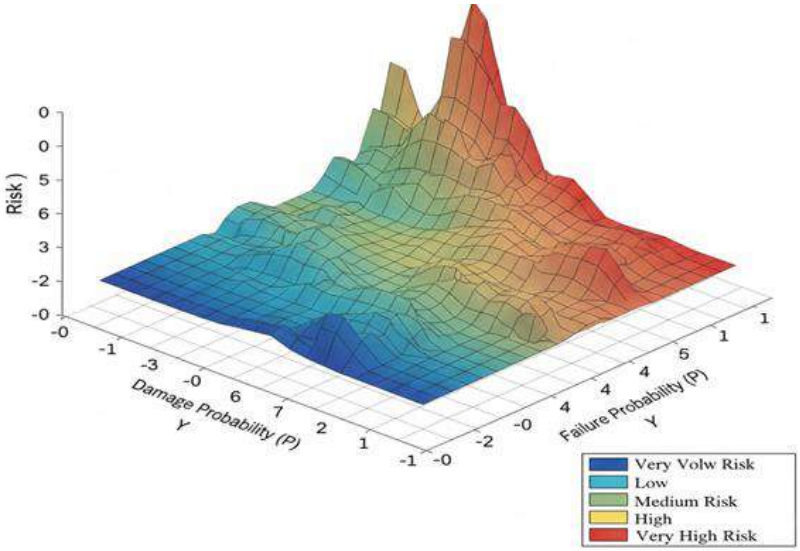

Figure 4.42. Defuzzification surface of risk *R=f(P,Y)* with increased granularity

Furthermore, the third input, the neural network anomaly index $\phi_i^{NN}(t)$, acts as a dynamic modifier of the risk surface. As $\phi_i^{NN}(t)$ transitions from "No Anomaly" to "Critical Anomaly," the entire defuzzification surface *R=f(P,Y)* shifts upward, becoming steeper in high-risk zones. This dynamic sensitivity ensures that latent, early-stage anomalies (detected only by the NN) instantly trigger an increase in the integral risk R_i, even when *P* and *Y* remain moderate.

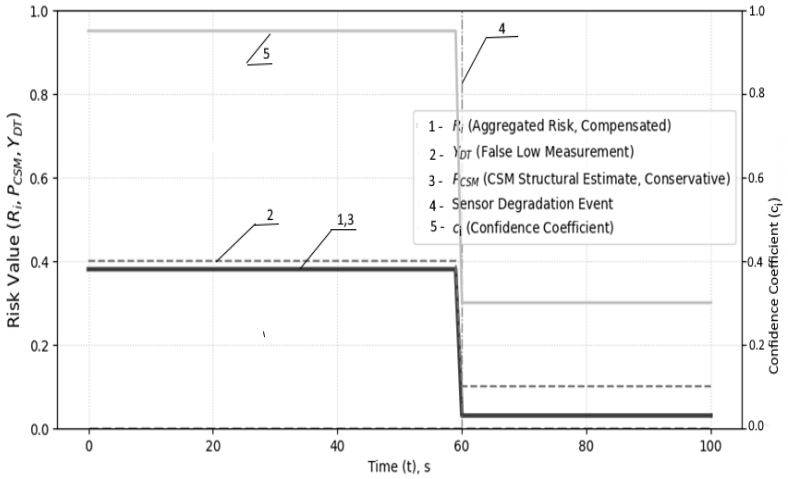

Figure 4.43. Dynamic confidence compensation mechanism (c_i)

Figure 4.43 illustrates the core principle of the adaptive compensation mechanism. The thick blue line (R_i) shows the aggregated system risk. Prior to t=60, the Confidence Coefficient c_i is high, causing R_i to closely track the DTs physical readings (Y_{DT}). At t=60, simulating a sensor degradation event (e.g., high noise or data loss), c_i instantly drops. Consequently, the aggregation function automatically reduces the influence of the now-unreliable Y_{DT} and shifts reliance towards the conservative, structurally accurate estimate derived from the CSM. This prevents the aggregated risk R_i from dropping to an unverified low level, confirming the system's robustness against data quality degradation.

While Figure 4.43 demonstrates the temporal robustness of the system, it is also essential to quantify the relative influence of each input source—the cognitive (P_{CSM}), physical Y_{DT}), and intelligent $\phi_i^{NN}(t)$) layers-on the final partial risk R_i The relative contribution is constantly modulated by the confidence coefficient c_i and the active fuzzy rules. Figure 4.44 visualizes this dynamic fusion of heterogeneous data by illustrating how the influence of each source shifts under different operational scenarios.

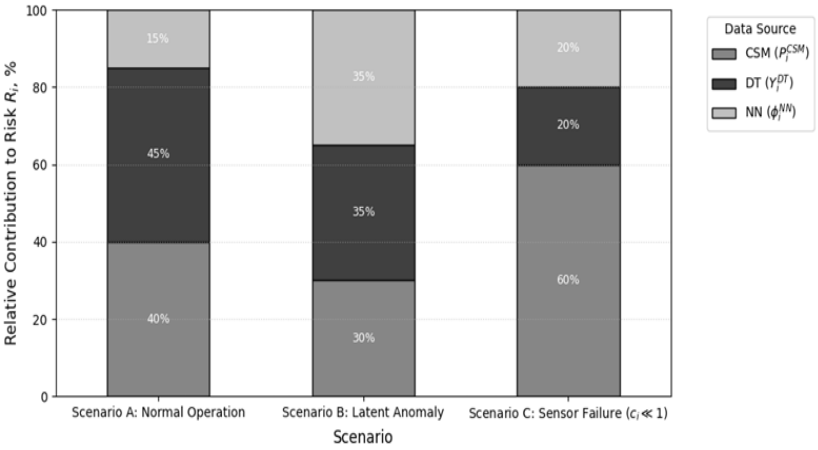

Figure 4.44. Relative contribution of data sources to R_i

Figure 4.44 confirms that the hybrid architecture successfully implements the desired expert policy across various operational contexts. In Scenario A (Normal Operation), the risk is primarily determined by the physical reality of the DT (Y_{DT}) and the structural baseline (P_{CSM}), with the Neural Network ($\phi_i^{NN}(t)$) contributing minimally. Conversely, Scenario B (Latent Anomaly) shows a pronounced increase in the $\phi_i^{NN}(t)$ contribution, confirming the system's increased sensitivity to early anomaly detection. Finally, Scenario C (Sensor Failure) demonstrates the robustness mechanism: the contribution of the structural model P_{CSM}) becomes dominant, compensating for the high uncertainty introduced by the degraded physical or data-driven inputs. This dynamic allocation of influence ensures that the aggregated risk R_i is always grounded in the most credible available information.

The surface represents the dependence of the integrated risk on the failure probability P and damage magnitude Y for the adaptive neuro-fuzzy model after the training stage using DT data. Unlike the baseline model (Fig. 4.39), the surface has smoother transitions and nonlinear areas, reflecting the refinement of boundaries between linguistic terms and increased sensitivity to high-risk zones. This shape of the surface demonstrates the ANFIS system's ability to adapt to real SPP operating conditions and to provide a more accurate approximation of the risk function. After adaptation, the surface is characterized by a more precise reflection of nonlinear effects, which makes the model suitable for integration into the DT system and cognitive-simulation risk forecasting

schemes. Thus, the adaptive fuzzy model trained on DT data provides dynamic refinement of inference rules and membership functions, enabling the formation of a self-adjusting risk assessment system for SPPs. The integrated fuzzy model represents an advanced hybrid architecture for real-time risk assessment of CTS. Its key advantage lies in the successful integration of causal (CSM logic), physical (DT models), and intelligent (neural network analysis) levels. The adaptive ANFIS system ensures dynamic self-learning, allowing continuous refinement of model parameters based on real data. This leads to a radical increase in prediction accuracy (MAE 0.03–0.05), while maintaining the interpretability inherent to fuzzy logic, which is essential for operator trust in DSS.

To further improve the reliability and operational applicability of the integrated system, the following steps are recommended:

1. **Dynamic management of c_i.** To ensure continuous model updating and computational stability, the dynamic management of the confidence coefficient c_i is crucial. It is recommended to transition from periodic (24-hour) updates of c_i to an adaptive, streaming mechanism. The confidence coefficient c_i is formalized not as a static value but as a dynamic metric of data quality, computed using a Sigmoid function that integrates key observability metrics:

$$c_i(t) = Sigmoid(\alpha \cdot C_i(t) - \beta \cdot N_i(t) - \gamma \cdot, D(t)), \quad (4.89)$$

where $C_i(t) \in [0,1]$ is the completeness metric (proportion of missing data);

$N_i(t) \in [0,1]$ is the statistical noise (measure of data variability);

$D_i(t) \in [0,1]$ is the sensor drift (measure of deviation from baseline).

α, β, γ are adaptive tuning weights defined during ANFIS training, allowing the system to prioritize sensitivity to specific data faults.

This adaptive formula ensures that when data quality degrades (e.g., $N_i(t)$ increases), $c_i(t)$ drops rapidly, causing the fuzzy aggregator ($\varphi(\cdot)$) to reduce reliance on unreliable DT inputs and increase dependence on conservative CSM estimates.

2. **Increased granularity:** consider increasing the number of linguistic terms (e.g., to five or more) for the input variables “Failure Probability” and “Damage.” This will allow the ANFIS system to approximate the risk function with higher detail, especially in key transition zones where maintenance decisions are most critical;

3. **Formalization of neural index influence:** a detailed description is required for the fuzzification mechanism and integration of latent features $\phi_i^{NN}(t)$ extracted by the neural network. These features, which are difficult

to interpret linguistically, must be rigorously formalized within the fuzzy rule base to preserve the high interpretability of the hybrid model.

4. **Optimization of membership functions:** it is advisable to explore the use of Gaussian or trapezoidal membership functions instead of triangular ones. This may provide a smoother output surface and potentially improve the convergence speed and accuracy of ANFIS optimization algorithms when training complex nonlinear dependencies.

4.3.5 Hybrid neural network model for predicting failure risk of components of complex technical systems

Under the conditions of high complexity of SPP and the nonlinearity of their degradation processes, traditional analytical and fuzzy methods of risk assessment prove to be insufficient for long-term forecasting. Neural network models trained on telemetry and operational data of the digital twin make it possible to identify hidden dependencies between equipment condition parameters and failure probability, making them a key element of the intelligent technical condition monitoring system. In the overall architecture, the neural network subsystem occupies the upper level of the analytical block of the digital twin, interacting with:

- the CSM, which forms causal failure probabilities $P_i(t)$;
- the fuzzy subsystem (Section 4.3.4), which aggregates local risk estimates $R_i(t)$;
- the digital twin, which provides a stream of physical and sensor data.

Thus, the neural network model implements the functionality of forecasting $\hat{R}_{t+1}$ - the expected failure risk one step ahead, taking into account the multidimensional dynamics of SPP parameters

$$\hat{R}_{t+1} = f_{\Theta}(X_{t-k:t}), \tag{4.90}$$

where f_θ is a parameterized neural network;

$X_{t-k:t}$ is a window of telemetry observation data over k time steps

The integration of the neural network model with other analytical modules of the digital twin (the CSM and the fuzzy subsystem) ensures the cognitive consistency of the forecast. The technical condition indicator S_t, obtained from the CSM, and the diagnostic features F_t (F_t - the diagnostic features extracted from the fuzzy subsystem (for example, local risk estimates R_i from Section 4.3.4), representing aggregated fuzzy metrics that capture diagnostic uncertainties), extracted from the fuzzy subsystem, serve as expert knowledge for the neural network, enriching the raw sensor data and allowing the model to converge faster and make more well-founded decisions under uncertainty. The overall architecture is designed to capture time-varying dependencies, where the observation window length, k, is a critical hyperparameter. In the conducted study, a fixed window length of

k=60 seconds was empirically selected to balance the need for capturing short-term dynamic events (e.g., vibration spikes) and maintaining computational efficiency.

The input feature vector X_t is formed based on the integration of three data sources:

1. OREDA - reference databases on ship equipment failures that define the statistical parameters of time-to-failure distributions (Weibull, Lognormal);
2. SPP DT - synthetic and experimental data on aggregate parameters (pressure, temperature, vibration, rotation speed, consumption);
3. Online Telemetry - streams of sensor data from IoT devices and diagnostic modules.

The integration of the neural network model with other analytical modules of the digital twin (the CSM and the fuzzy subsystem) ensures the cognitive consistency of the forecast. The technical condition indicator S_t obtained from the CSM, and the diagnostic features F_t, extracted from the fuzzy subsystem, serve as expert knowledge for the neural network, enriching the raw sensor data and allowing the model to converge faster and make more well-founded decisions under uncertainty. The robust predictive capability of the hybrid model is founded on the cognitive consistency provided by this integrated input feature vector. Figure 4.45 illustrates the detailed, three-level aggregation scheme used to construct the final input vector X_t.

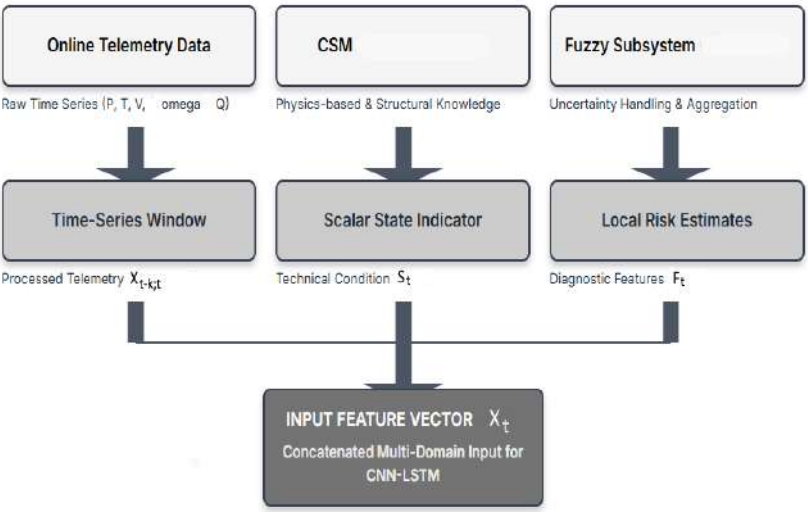

Figure 4.45. Input feature vector generation and aggregation scheme

The aggregation scheme presented in Figure 4.45 demonstrates that the final input vector X_t is a concatenation of processed features from three distinct, yet complementary, domains: 1) the time-series window ($X_{t-k:t}$) representing the immediate dynamic state of the SPP; 2) the scalar technical condition indicator (S_t) providing physics-based knowledge; and 3) the local risk estimates ($F_{t,}$) quantifying uncertainty derived from the Fuzzy Subsystem. The comprehensive, multi-domain composition of X_t is critical for accurately modeling the complex degradation dynamics of the SPP, allowing the model to correlate low-level sensor fluctuations with high-level structural health indicators.

The overall vector for each subsystem is represented as:

$$x_t = [P_t, T_t, V_t, \omega_t, S_t, Q_t, F_t], \tag{4.91}$$

where: P_t - the pressure of the working medium (oil, fuel, coolant) at key points of the circuits;

T_t - the temperature of the corresponding circuits (oil, fuel, water, exhaust);

V_t - the vibration activity of the units (e.g., bearings, shaft, turbocharger);

ω_t - the rotational speed (revolutions) of the main engine or auxiliary turbines;

S_t - the technical condition indicator of the subsystem (for example, a binary label: operational/pre-failure, or a scalar degradation level indicator derived from the physical model within the CSM, as detailed in Section 4.3.1);

Q_t - the fuel or coolant flow rate reflecting the load mode

The overall input vector X_t is a fusion of raw sensor data and synthetic/diagnostic metrics, formalized by Equation (4.91). For the CNN-LSTM processing, the temporal nature of the vector must be explicitly defined, especially concerning the low-frequency, expert-driven features S_t and F_t. The combined input window $X_{t-k:t}$ is a matrix of size k x M (e.g., 60 x 7), where k=60 seconds is the window length. The features P_t, T_t, V_t, ω_t, Q_t are high-frequency time series (one value per time step Δ_t). However, the synthesized features S_t and F_t are inherently low-frequency or scalar:

1. Technical condition indicator (S_t): derived from the CSM, S_t represents the aggregated structural health or degradation level of the subsystem. In this implementation, S_t is a scalar value (e.g., $S_t \in [0,1]$ or a

binary label) that is recalculated at a much lower frequency than the sensor data (e.g., once per minute or once per prediction window);

2. Diagnostic Features (F_t): Extracted from the fuzzy subsystem, F_t (also a scalar value or a short feature vector) quantifies diagnostic uncertainty and local risk estimates. This feature is also updated at a low frequency, tied to the aggregation interval of the fuzzy subsystem.

Integration into the time window $X_{t-k:t}$**:**

To maintain the matrix structure required by the 1D-CNN layer, the low-frequency features (S_t and F_t) are broadcast across the entire observation window. This means that for all k time steps within the window $X_{t-k:t}$, the values for S and F remain constant.

Specifically, the values S and F used for the window ending at time t are derived from the latest available low-frequency update preceding t:

$$Feature(t') = \begin{cases} Sensor\,\mathrm{Re}ading(t')\ for P,T,V,\omega,Q \\ S_{latest}\ for S where t-k \le t' \le t \\ F_{latest}\ for F where t-k \le t' \le t \end{cases} \tag{4.92}$$

This method ensures that the neural network leverages the high-frequency temporal dynamics of the physical signals while simultaneously being informed by the consistent, expert-derived, low-frequency state indicators throughout the entire observation period.

Before training, the data are normalized, missing values are interpolated using the spline approximation method, and outliers are removed by a median filter. For each time window of length k = 60 seconds, a training batch ($X_{t-k:t}$, R_t) is formed.

To account for the spatio-temporal structure of the data, a hybrid CNN–LSTM neural network architecture (Fig. 4.46) is used, combining the advantages of convolutional neural networks (CNNs) for spatial feature reduction and local pattern recognition and recurrent neural networks (LSTMs) for temporal encoding of long-term dependencies [128]. This hybrid structure ensures simultaneous processing of high-frequency local features and long-term dependencies in the time series dynamics of the marine power power plant parameters. Furthermore, to leverage the multi-domain nature of the SPP, this architecture is implemented using a parallel, multi-branch design where distinct CNN-LSTM streams process data segmented by primary subsystem (Fuel, Cooling, Electric) before their latent states are aggregated.

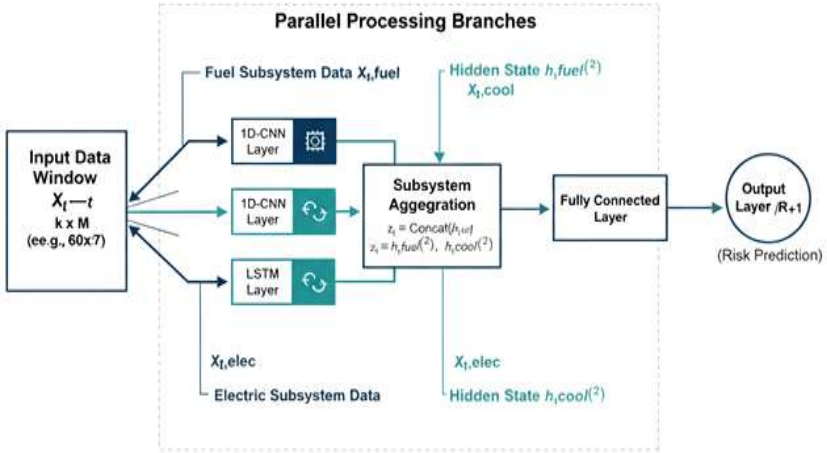

Figure 4.46. Multi-Branch CNN-LSTM architecture for SPP risk prediction with subsystem aggregation

Input vector segmentation and parallel processing

To effectively model the multi-dimensional dynamics of the SPP and utilize subsystem-specific degradation characteristics, the hybrid CNN-LSTM architecture is implemented as a multi-branch structure (see Fig. 4.46). This design ensures that feature extraction and temporal modeling are specific to the physical domain of each major component.

The full input observation window $X_{t-k:t}$ (of size k x M, where k=60 seconds and M=7 features) is initially segmented along the feature axis into three non-overlapping sub-matrices: $X_{t,fuel}$, $X_{t,cool}$, and $X_{t,elec}$, corresponding to the telemetry features of the fuel, cooling, and electric subsystems, respectively.

Segmentation rule:

1. $X_{t,fuel}$**:** includes parameters directly related to fuel system operation (e.g., fuel pressure Pf_{uel}, fuel temperature T_{fuel}, and fuel flow rate Q_t);
2. $X_{t,cool}$**:** includes parameters related to the cooling circuit (e.g., coolant temperature T_{cool}, coolant pressure P_{cool});
3. $X_{t,elec}$**:** includes parameters reflecting mechanical and electrical load/condition (e.g., vibration V_t, rotational speed ω_t, and auxiliary metrics).

Each of these segmented sub-matrices is fed into its own independent, yet structurally identical CNN-LSTM branch (as defined by Layers 1 and 2). This parallel processing ensures the dedicated extraction of local

patterns and the robust modeling of long-term temporal dependencies specific to each physical domain.

The resulting hidden state vectors ($h_{t,fue}^{(2)}$, $h_{t,cool}^{(2)}$, $h_{t,elec}^{(2)}$) from the parallel LSTM blocks are subsequently concatenated in the subsystem aggregation layer to form the unified system state vector z_t, as formally defined in Equation (4.95). This integrated approach guarantees the cognitive consistency of the forecast by synthesizing knowledge extracted from disparate physical domains of the complex technical system.

The model includes four functional layers:

1. Feature Extraction Layer (1D-CNN).

Extracts local patterns from time series (for example, micro-vibration oscillations, short-term temperature spikes).

$$h_t^{(1)} = \mathrm{Re}Lu(W_c \cdot X_t + b_c), \tag{4.93}$$

where W_c - the matrix of convolutional filters;

X_t - the input observation vector;

b_c - the bias term;

ReLu - the activation function

2. Temporal Encoding Layer (LSTM).

Forms long-term dependencies in the dynamics of equipment condition:

$$h_t^{(2)} = LTSM(h_t^{(1)}), \tag{4.94}$$

where the LSTM uses memory cell and forget gate mechanisms to account for the inertial properties of processes in the subsystems of the SPP (e.g., thermal and vibration effects)

3. Subsystem Aggregation Layer.

At this level, the latent features obtained from individual subsystems (fuel, cooling, and power supply) are concatenated into a unified system state vector:

$$z_t = Concat(h_{t,\,fuel}^{(2)}, h_{t,cool}^{(2)}, h_{t,elec}^{(2)}), \tag{4.95}$$

where $h_{t,\,fuel}^{(2)}, h_{t,cool}^{(2)}, h_{t,elec}^{(2)}$ - hidden state vectors obtained at the second level (LSTM layer) for different subsystems of the SPP:

$h_{t,\,fuel}^{(2)}$ - fuel system;

$h_{t,cool}^{(2)}$ - cooling system;

$h_{t,elec}^{(2)}$ - power supply system

This layer reflects the implicit interaction between subsystems by assuming that the concurrently calculated hidden states from each subsystem (Fuel, Cooling, Electric) independently contribute to the overall system risk. This concatenation step is crucial as it ensures a

comprehensive, cognitively consistent representation of the multi-domain system state before the final prediction.

4. Fully Connected Output Layer.

Generates the forecast of the integral failure risk:

$$\hat{R}_{t+1} = \sigma(W_0 \cdot z_t + b_0), \tag{4.96}$$

where $\sigma(\cdot)$ is the sigmoid activation function that normalizes the result within the range [0,1].

The resulting value is $\hat{R}_{t+1}$ interpreted as the probability of the system transitioning into a degradation state at the next time step.

The model receives synchronized time series of operational parameters from subsystems (pressure, temperature, vibration, rotation speed, fuel flow, etc.) as input. First, the convolutional layers (CNN) extract local dynamic features of the signals - vibration spikes, short-term anomalies in pressure and temperature. Then, the LSTM layer aggregates the feature sequences, modeling their temporal evolution and retaining the system's previous states. The results obtained from individual subsystems are combined in the feature aggregation layer, forming an integrated representation of the technical condition. The output fully connected layer computes the forecast of failure probability and the system's integral risk for the next time step, $R_{(t+1)}$. The presented architecture (Fig. 4.46) provides multilayer perception of information: from local parameter fluctuations to an integral risk assessment at the level of the entire SPP. This approach combines the interpretability of the physical models of the digital twin with the learning capability of neural networks, thus improving failure prediction accuracy under conditions of incomplete and noisy data. The model is trained in an end-to-end mode using historical and current data from the digital twin, as well as state labels determined from diagnostic results. This approach ensures a balance between the interpretability of physical processes and the high predictive power of data-driven components.

To ensure the reproducibility of experiments and formalize the training process of the hybrid neural network model based on CNN–LSTM, an algorithm for model construction and training has been developed (Fig. 4.47). The algorithm reflects the complete sequence of stages - from the collection and normalization of sensor data from the digital twin of the SPP to the evaluation of prediction accuracy and the adjustment of weight coefficients during the training process. Each step is accompanied by computational procedures that take into account both the features of the physical model (through the Lphys component) and the temporal correlation of signals provided by the LSTM layers.

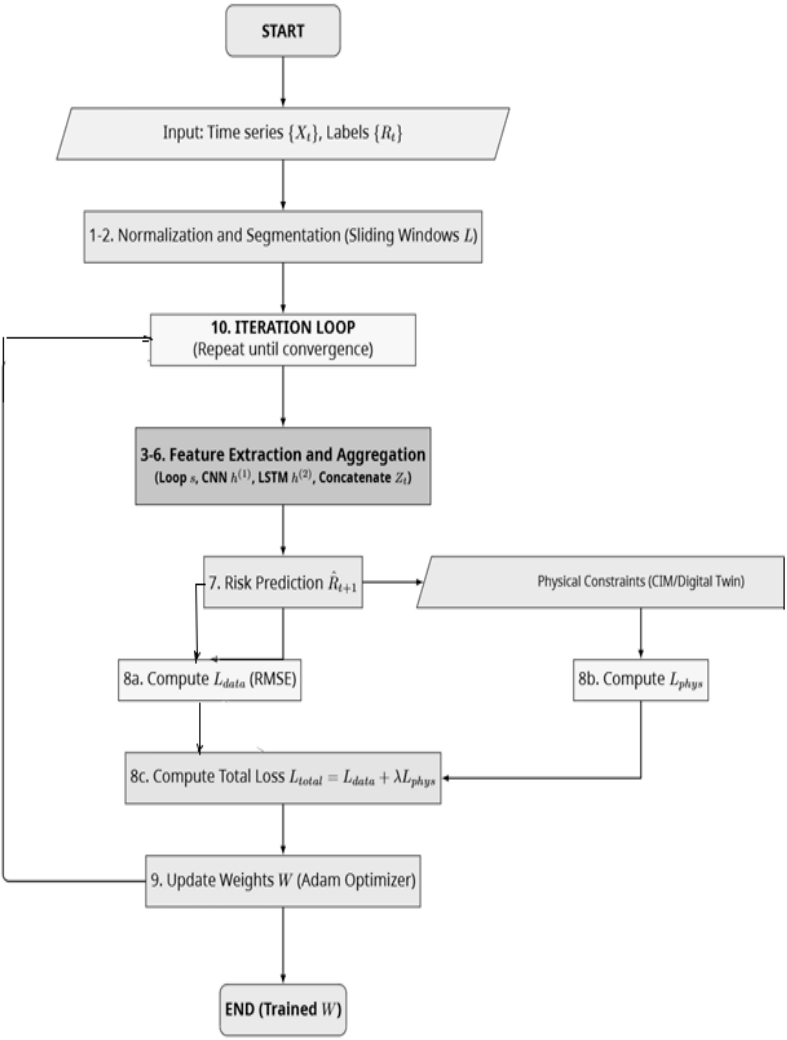

Figure 4.47. Block diagram of the algorithm for constructing and training the hybrid CNN–LSTM

The complexity of training a Physics-Informed Neural Network (PINN) model requires a precise methodology that integrates data-driven error minimization with physical constraint enforcement. This process is formalized in Algorithm 4.1, which outlines the complete training loop for the hybrid digital twin.

Algorithm 4.1. Training algorithm for the HDTn (CNN-LSTM + physics model)

Input:

- *Historical dataset $D = \{X_t, U_t, y_t\}$ (telemetry, control inputs, true risk labels)*
- *Initial physics model parameters φ_0*

Output: Trained model parameters θ, $\varphi = \varphi_0$ (Neural Network weights θ and fine-tuned physics parameters φ)

1. *Preprocessing: Normalize input data (z-score normalization) and segment data into sliding windows of length L.*
2. *Initialize network weights $\theta \leftarrow \theta_0$, physics parameters $\varphi \leftarrow \varphi_0$.*
3. *for epoch = 1..E:*
4. *for batch in D:*
5. *// Hybrid prediction (combining physics and ML)*
6. *phys_pred = f_{phys}(batch.X, batch.U; φ) // Output from cognitive/physics model*
7. *ml_pred = f_{ML}(batch.s; θ) // Output from CNN-LSTM Model*
8. *$\hat{y} = phys\_pred + \lambda(batch) * ml\_pred$ // Ensemble Prediction*
9. *// Loss Calculation*
10. *L_{data} = MSE($\hat{y}$, batch.y) // Error relative to actual observations*
11. *L_{phys} = PhysicsLoss(f_{phys}, batch) // Error relative to physical laws*
12. *$L_{total} = L_{data} + \alpha \cdot L_{phys} + \beta \cdot \mathrm{Re}\, g(\Theta)$ // Total combined loss*
13. *// Gradient Descent and Weight Update*
14. *update $\theta \leftarrow \theta - \eta \nabla_\theta L_{total}$ // Update NN weights (main step)*
15. *optionally update $\varphi \leftarrow \varphi - \eta \nabla_\varphi L_{total}$ (small step) // Fine-tune physics params*
16. *evaluate metrics on validation set*
17. *if early_stop: break*
18. *Return θ^*, φ^**

This unified algorithm clearly integrates the data pre-processing steps, the hybrid prediction mechanism, the physics-informed loss function, and the dual parameter update process (for both the neural network θ and the physics model φ), eliminating the previous redundancy.

The algorithm reflects the complete training cycle of a hybrid digital twin, in which the physical model and the neural network jointly minimize a combined loss function (data+physics+regularization). Periodic adaptation of the parameters φ and θ ensures the model's self-learning capability based on incoming data under changing operating conditions of the SPP.

For a visual demonstration of the operation principles of the hybrid neural network model's training loop, as well as to detail the paths of the forward pass and backpropagation of the error, a data flow diagram was developed and is shown in Figure 4.48.

The diagram illustrates how the input sensor data X(t) and external physical variables are processed to form the forecast $\hat{R}_{t+1}$, and how the total error L_{total} is used to adjust the weight coefficients W of all trainable layers of the neural network (CNN, LSTM, Prediction Layer).

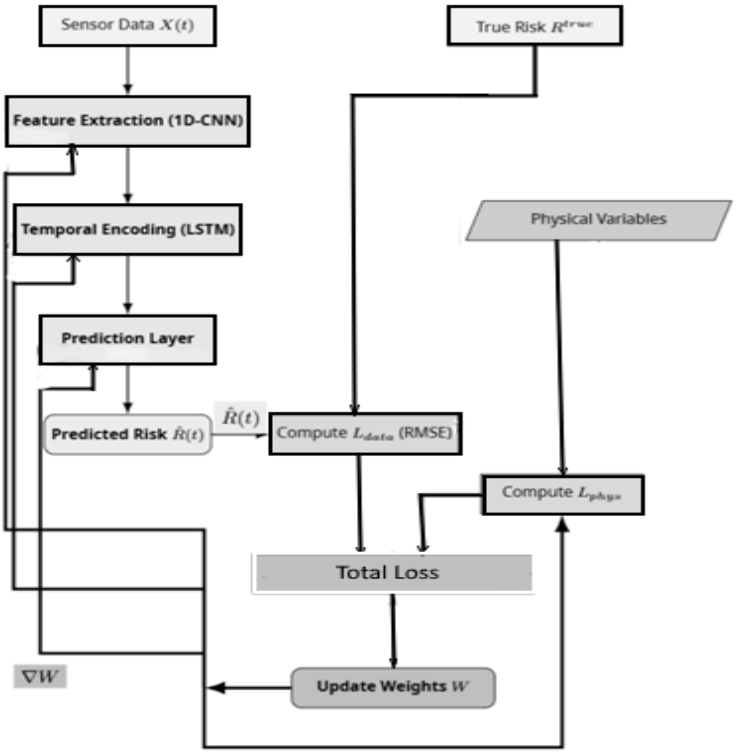

Figure 4.48. Data flow and backpropagation scheme during the training process of the hybrid model

As shown in Figure 4.48, the model training is performed by minimizing L_{total}, which combines two key aspects: compliance with empirical data and adherence to physical constraints. The component L_{data} ensures the accuracy of empirical prediction, while L_{phys} regulates the physical consistency of output data with the fundamental laws of the SPP. To prevent overfitting

and ensure the physical coherence of predictions, a combined Physics-Informed Loss function is used, which is formalized as follows. For the practical implementation of the hybrid model, a multilayer architecture was used, including feature extraction, temporal encoding, aggregation, and prediction blocks. Each layer performs specialized functions - from filtering and normalizing time series to forming an integral risk indicator.

The summary structure of the architecture is presented in Table 4.16.

Table 4.16

Architecture of the hybrid CNN–LSTM model

№	Layer type	Kernel size / number of neurons	Activation function	Output data	Output Data Dimension
1	Input layer	–	–	$Xt = [Pt, Tt, Vt, \omega t, Lt]$	kxM (e.g., 60x 7)
2	Convolutional (1D-CNN)	3×1, 64 filters	ReLU	$h_t^{(1)}$	64 (per subsystem)
3	LSTM	128	tanh / sigmoid	$h_t^{(2)}$	128 (per subsystem)
4	Dropout	0.2	–	–	128 (per subsystem)
5	Subsystem aggregation	–	concat	zt	3x128 = 384
6	Fully connected layer	64	ReLU	–	64
7	Output layer	1	sigmoid	$\hat{R}_{t+1}$	1

The model architecture is designed for real-time streaming data processing, allowing it to utilize parameter values received from SPP subsystem sensors without prior batch preprocessing. The convolutional filters detect short-term anomalies, the LSTM layers provide memory of long-term degradation trends, and the fully connected part of the network performs feature fusion to compute the integral risk indicator.

The detailed structural composition of the developed hybrid neural network model, illustrating the dimensionality of the input vector, the sequence and functional purpose of key layers (CNN, LSTM, FC), as well as their output dimensionality, is shown in Figure 4.49**.**

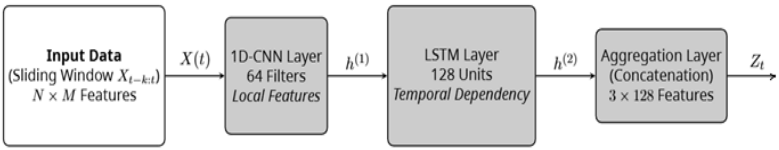

Figure 4.49. Structural composition of the hybrid neural network (layer dimensionality)

The structural diagram presented in Figure 4.49 demonstrates the process of transforming input data and the step-by-step reduction of the feature space dimensionality. This pipeline includes the following key blocks:

1. Input Data. Represents a window of time observations ($X_{t-k:t}$), including N time steps and M operational and diagnostic features for each SPP subsystem. The block provides the input sensor data for further processing;
2. 1D-CNN Layer (64 Filters). A convolutional layer with 64 filters, whose main function is to extract local spatial features and detect microfluctuations (patterns) in the time series. The output is a vector $h^{(1)}$ with 64 features;
3. LSTM Layer (128 Units). A long short-term memory block designed to model complex nonlinear temporal dependencies. The 128 LSTM cells ensure efficient retention of contextual information about system dynamics, forming the hidden state $h^{(2)}$;
4. Aggregation Layer (Concatenation). At this stage, the output vectors $h^{(2)}$ obtained from the LSTM blocks for all three key subsystems (Fuel, Cooling, and Electric) are concatenated into a single vector Z_t. The resulting dimensionality is 3×128 = 384 features, forming a comprehensive multidimensional representation of the entire SPP's technical condition;
5. FC Layer (64 Neurons). A fully connected layer that integrates the aggregated features. It projects 384 features into a more compact and high-level space of 64 neurons, which acts as an effective classifier/regressor;
6. Output Prediction (Risk Prediction $\hat{R}_{t+1}$). The final layer with one neuron and a sigmoid activation function, which outputs the predicted value of the integral failure risk $\hat{R}_{t+1}$ in the range [0, 1] for the next time step.

This multilayer structure ensures that the model effectively processes both the local features of signals (CNN) and the long-term temporal dynamics (LSTM), which is critically important for improving risk

prediction accuracy under the operating conditions of complex technical systems.

To implement the training of the hybrid model, a physics-informed learning strategy was applied, which allows the integration of the physical constraints of the SPP digital twin into the neural network optimization process. Figure 4.50 shows the structure of the model's training loop.

At the input of the model, synchronized time series of SPP subsystem parameters (pressure, temperature, vibration, rotation speed, load) are supplied, having undergone normalization and risk level labeling. The Feature Extraction (1D-CNN) block identifies local patterns in the signal, followed by the Temporal Encoding (LSTM) module, which forms long-term temporal dependencies. The physics-informed component of the loss function, including the term L_{phys}, ensures the alignment of the neural network prediction with the physical constraints of the ship power plant - the laws of conservation of energy, mass, and heat balance. Thus, the hybrid structure combines the advantages of data-driven and physics-based approaches, creating a unified trainable loop of the DT.

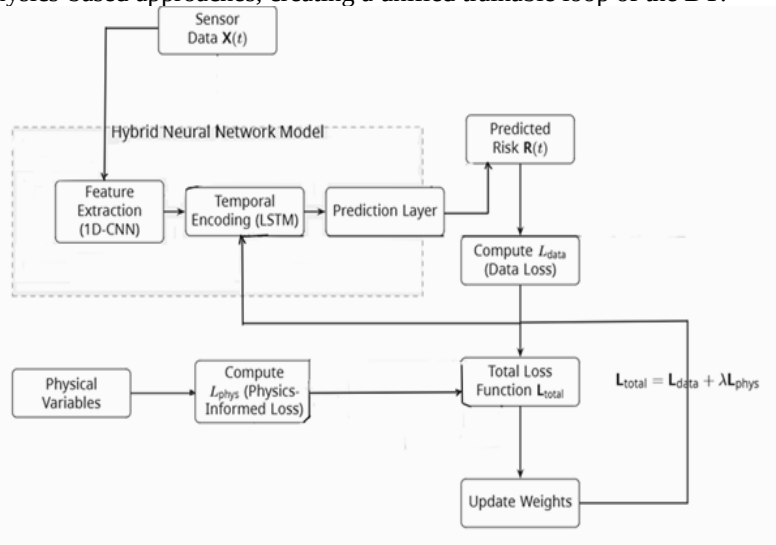

Figure 4.50. Training scheme of the hybrid neural network model with physics-informed loss

The training scheme of the hybrid neural network model with the physics-informed loss function $L_{total}=L_{data}+\lambda L_{physL}$.

The figure shows two interconnected loops: the main data flow (sensor time series → feature extraction → LSTM → prediction → output) and the additional branch of physical constraints coming from the digital twin and

forming the L_{phys} component of the total loss function. As seen from the diagram, the training is performed iteratively: the neural network minimizes the total error L_{total}, in which the term L_{data} corresponds to the match with actual observations, while L_phys ensures compliance with physical laws and technological constraints. This approach allows the model not only to reproduce empirical dependencies but also to maintain results within permissible engineering limits, eliminating physically impossible solutions.

This training scheme is the core of the integration of the neural network module with the SPP digital twin and serves as the basis for subsequent analysis of risk prediction accuracy. The input consists of multichannel time series, including telemetry data (pressure, temperature, vibration, rotation speed) and parameters generated by the digital twin (degradation assessment, remaining resource, energy balances). The feature extraction block performs preliminary feature extraction, normalization, and noise filtering. Then, the LSTM module extracts dynamic dependencies in temporal data, and the output Prediction Layer generates the failure risk prediction *R(t)*. For this purpose, during training, a combined physics-informed loss and physics-informed regularization are used. To prevent overfitting and ensure the physical consistency of predictions, a physics-informed loss function (Physics-Informed Loss) is applied:

1. L_{data} - the standard prediction error based on observed data (for example, MAE or MSE between the predicted and actual risk values);
2. L_{phys} - a penalty for violation of physical dependencies derived from thermodynamic, hydraulic, and vibration models of the digital twin.

The final loss function is expressed as

$$L_{total} = L_{data} + \lambda \cdot L_{phys} \tag{4.97}$$

where λ is the regularization coefficient that determines the degree of influence of physical constraints on the training process

The selection of the regularization coefficient λ is a critical stage. For this task, λ was chosen empirically (using Grid Search) to achieve an optimal balance between prediction accuracy (L_{data}) and the physical consistency of predictions (L_{phys}).

Minimization of L_{total} ensures the consistency of neural network prediction results with physical laws and prevents the generation of physically unfeasible system states. This integration of sources provides comprehensive training of the model based both on empirical operational data and on formalized structural dependencies between elements of the CTS.

The training of the hybrid neural network model is carried out on a combined dataset that includes:

• operational time series of the SPP digital twin (pressure, temperature, vibration, rotational speed, generator load, etc.);

• historical archives of equipment failures and degradation (for example, from the OREDA database);

• synthetic data from the CSM, describing probabilistic–causal failure relationships between subsystems.

Such integration of data sources provides comprehensive model training - both from empirical operational data and from formalized structural dependencies among CTS elements.

The main component L_{data} is calculated as the RMSE:

$$L_{data} = \frac{1}{N}\sum_{i=1}^{N}(\hat{R}_t - R_t^{true})^2 \tag{4.98}$$

Physics-Informed Loss Component (L_{phys})

The Physics-Informed Loss component (L_{phys}) serves as a critical regularizer in the hybrid model's training process. Its main function is to enforce that the neural network's predictions for system state evolution remain consistent with the fundamental conservation laws governing the SPP.

This component is formalized as a penalty for violating the physical constraints derived from the DT's high-fidelity thermodynamic, hydraulic, and vibration models. The specific laws incorporated include:

- conservation of energy (heat balance in cooling/combustion subsystems);

- conservation of mass (flow rates and levels in fluid subsystems);

- conservation of momentum (pressure drops and rotational dynamics)

Formal Definition of the Physical Loss Term

The physical loss term, L_{phys} is defined as the MSE of the residuals, R_i, of the governing physical equations. The residual R_i is the error that results when the predicted parameters and their derived temporal derivatives (calculated from the neural network's input window) are substituted into the physical conservation equations (where the result should ideally be zero).

This loss function quantifies the deviation from the conservation laws formalized within the DT's CSM. The general form of the L_{phys} is:

$$L_{phys} = \frac{1}{N \cdot k_{phys}}\sum_{i=1}^{N}\sum_{i=1}^{K_{phys}} R_i(X_{t-k:t}, \hat{R}_{t+1}, \Theta)^2 \tag{4.99}$$

where N is the total number of time steps (windows) in the training batch;

K_{phys} is the total number of distinct physical equations/constraints applied (i=1…K_{phys});

$R_i(\ldots)=0$ represents the i-th differential or algebraic conservation equation;

$X_{t\text{-}k:t}$ is the input observation window;

$\hat{R}_{t+1}$ is the predicted risk;

Θ represents the neural network weights and potentially the fine-tuned physics parameters

Illustrative example: conservation of energy (heat balance)

As an academic example, consider the simplified heat balance equation for the cooling circuit (Subsystem *j*), which is one component of the total L_{phys}:

$$R_{heat} = \frac{dE}{dt} - (\dot{\Theta}_{in} - \dot{\Theta}_{out}) = 0,$$

where $\frac{dE}{dt}$ is the rate of change of internal energy (approximated from $\nabla_t \cdot T_j$ and P_j);

$\dot{\Theta}_{in} - \dot{\Theta}_{out}$ is the net heat flow (derived from operational parameters Θ_t and physical model φ)

The corresponding residual term R_j to be minimized by the neural network weights Θ would thus be:

$$R_j \equiv (\nabla_t \cdot T_j) + \frac{1}{C}(f_j^{thermal}(P_j, \Theta_t) - T_j) = 0$$

The neural network is penalized if its prediction leads to an overall system state where the measured temperature change ($\nabla_t \cdot T_j$) significantly deviates from the value predicted by the physical thermal model $f_j^{thermal}(\ldots)$ (which accounts for fluid flow, heat exchange, etc., and is derived from the DT's CSM).

Since the input data consists of discrete time series, the temporal derivative (∇_t) of the physical parameters (like pressure P_j or temperature T_j) is approximated numerically using the forward difference scheme between two consecutive time steps (Δ_t):

$$\nabla_t P_j = \frac{P_j(t + \Delta t) - P_j(t)}{\Delta t} \tag{4.100a}$$

$$\nabla_t T_j = \frac{T_j(t + \Delta t) - T_j(t)}{\Delta t} \tag{4.100b}$$

This numerical method provides the empirical dynamic behavior of the system, which is then enforced to align with the theoretical behavior f_j^{energy}) during the model's training. The final total loss $L_{total} = L_{data} + \lambda \cdot L_{phys}$ is minimized iteratively to achieve a balance between empirical prediction accuracy (L_{data}) and physical consistency (L_{phys}).

The L_{phys} component controls the dynamic consistency between the observed changes in physical parameters and the modeled right-hand side f_j^{energy}, derived from the physical–mathematical model (for example, the equations of thermal and energy balance).

The operational efficiency and predictive stability of the hybrid CNN-LSTM model are critically dependent on the selection of optimal hyperparameters.

These parameters govern the learning process, the input representation, and the complexity of the neural network architecture. To ensure the reproducibility of the presented results and demonstrate the rigorous nature of the model development process, the key configuration parameters determined through systematic grid search and 5-fold cross-validation are detailed in Table 4.17.

Table 4.17

Hybrid CNN-LSTM model hyperparameter configuration

Parameter	Value	Description
Observation window length (k)	60 seconds	Length of the time series segment used for prediction.
Input feature count (M)	7	The dimensionality of the input vector X_t per time step.
Batch size	32	Number of samples processed before updating the model weights.
Optimizer	Adam	Adaptive moment estimation algorithm.
Learning rate (α)	10^{-3}	Adaptive learning rate for weight updates.
Epochs	100	Total number of full passes over the training dataset (with early stopping).
Regularization coefficient (λ)	0.5 (empirical)	Weighting factor balancing L_{data} and L_{phys}
Cross-validation scheme	5-fold	Scheme used to ensure generalization capability and prevent overfitting.

Following the enumeration of the configuration parameters in Table 4.13, a deeper analysis is required to validate the critical trade-off parameters. The most essential parameter is the regularization coefficient λ, which governs the balance between empirical data fitting and

adherence to physical laws. The selection methodology for this coefficient is illustrated below in Figure 4.51.

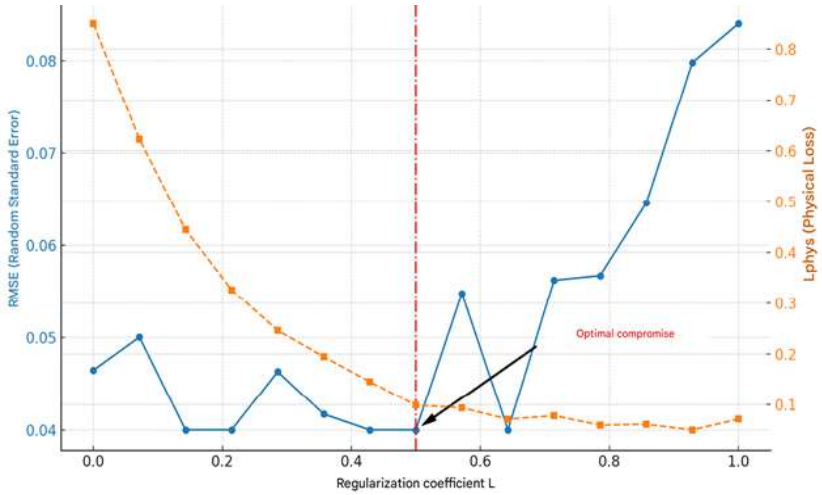

Figure 4.51. Trade-off between empirical accuracy (RMSE) and physical consistency (L_{phys}) as a function of the regularization coefficient λ

As detailed in Table 4.17, a fixed observation window length k of 60 seconds was chosen, balancing the need to capture sufficient temporal dependencies (for the LSTM component) with the necessity of maintaining computational efficiency.

The selection of the regularization coefficient λ is a critical stage, as it defines the trade-off between minimizing the empirical error (L_{data}) and satisfying the physical constraints (L_{phys}). A systematic approach, involving a grid search over the range $\lambda \in [0,1]$ combined with 5-fold cross-validation, was employed to determine the optimal balance.

The results of this analysis are summarized in Figure 4.51, which illustrates the performance of the model across varying λ values. As λ increases from 0.0 to 0.5, the physical consistency loss (L_{phys}) drops sharply, indicating rapid alignment with conservation laws. Simultaneously, the prediction error (RMSE, representing L_{data}) remains low. The optimal point λ=0.5 was selected based on the 'L-curve' methodology (or the 'elbow

method'), representing the point where the benefit of increasing physical consistency (further reduction in L_{phys}) no longer outweighs the inevitable cost in empirical accuracy (an increase in RMSE). Specifically, for $\lambda > 0.5$, the RMSE begins to increase significantly, demonstrating that the model starts prioritizing the rigid physical constraints over the subtle empirical dynamics captured in the sensor data. This rigorous analysis validates $\lambda = 0.5$ as the optimal trade-off point.

To assess the model's quality, standard regression metrics are applied: MAE; RMSE; R^2, as well as the k-fold cross-validation procedure, which guarantees model robustness against overfitting.

Together, these elements form a trainable neural subsystem of the digital twin, providing adaptive prediction of failure probability and integral risk based on sensor data streams

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|\hat{R}_i - R_i^{\exp}\right|, \quad RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{R}_i - R_i^{\exp})^2},$$

$$R^2 = 1 - \frac{\sum_i (R_i^{\exp} - \hat{R}_i)^2}{\sum_i (R_i^{\exp} - \bar{R})^2}, \qquad (4.101)$$

where R_i^{exp} - actual (empirical) risk values;

$\hat{R}$ - predicted values produced by the model;

$\bar{R}$ - the mean value of actual risks

Additionally, the Pearson correlation coefficient between the predicted and true risk values is used, characterizing the consistency of the model's dynamics with real degradation trends of the equipment.

Training is performed iteratively using mini-batches and the Adam optimization algorithm with an adaptive learning rate ($\alpha \approx 10^{-3}$). To evaluate generalization capability, *k-fold cross-validation* ($k = 5$) is applied, allowing quantitative assessment of model stability when the structure of the training dataset changes.

The physics-informed component of the loss function ensures that the neural network's prediction is consistent with the physical constraints of the marine power plant (law of energy conservation, mass balance, and

temperature gradient limitations). This increases the reliability of predictions under incomplete data conditions and makes the hybrid model robust to noise in sensor measurements. Thus, the CNN–LSTM architecture serves as a key component of the data-driven level of the digital twin of the marine power plant, integrating both empirical dependencies and physical laws reflected in the L_{phys} component.

Before training the model, an analysis of the distribution of the training data by risk levels and subsystems was carried out. To analyze the balance of the training dataset and verify the representativeness of data across different subsystems of the SPP, the samples were distributed according to risk levels and subsystem types. Such a distribution makes it possible to identify disproportions in data volume between subsystems and risk classes, which is critically important for the correct training of the neural network. As can be seen from Figure 4.50, most observations are concentrated in the range $R < 0.5$, which reflects normal operating states of the SPP. At the same time, the high-risk range ($R > 0.8$) contains only 5–7% of the samples, which requires balancing the training dataset using augmentation methods and synthetic data generated by the DT.

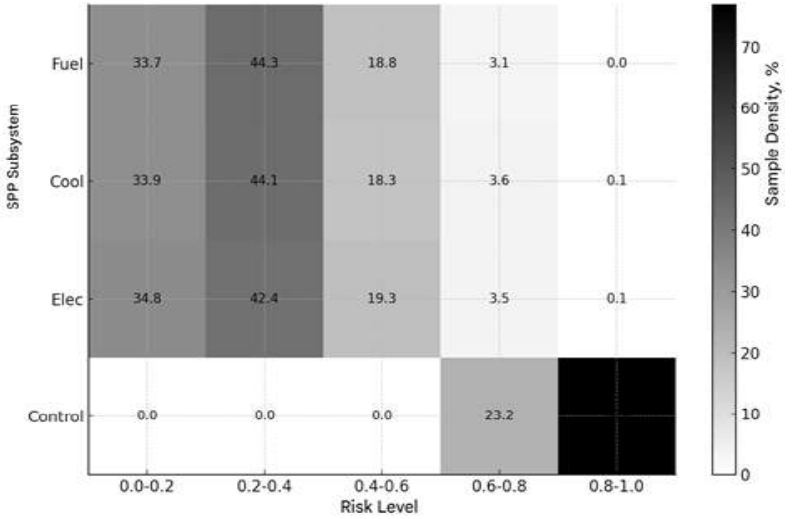

Figure 4.52. Distribution of training data by risk levels and subsystems

Before training the model, an analysis of the distribution of the training data by risk levels and subsystems was carried out to verify the representativeness of data across different SPP subsystems and risk classes. This distribution is critical for preventing model bias towards dominant classes.

The visualization includes four key categories: fuel, cool (cooling), elec (electric), and control (the subsystem previously labeled "fuel"). As shown in Figure 4.52, the majority of observations (indicated by the color intensity scale, representing sample density) are concentrated in the low-risk range (R<0.5), reflecting normal operating states of the SPP.

The highest data density corresponds to the cooling and electric subsystems, where medium-risk cases ($0.2 < R < 0.6$) prevail. Critically, the high-risk range (R>0.8) contains only 5-7% of the total samples, necessitating data augmentation and synthetic data generation techniques (e.g., oversampling or class weighting) to ensure adequate risk prediction for rare failure scenarios. Notably, the high-risk cases are concentrated in the control subsystem, reflecting the criticality of control faults despite its smaller overall data volume.

From the figure, it can be seen that the highest data density corresponds to the cooling and power supply subsystems, where medium-risk cases prevail. As can be seen from Figure 4.52, most observations are concentrated in the range R<0.5, which reflects normal operating states of the SPP.

The visualization includes four key categories: fuel, cool (cooling), elec (electric), andful (control subsystem). At the same time, the high-risk range (R>0.8) contains only 5-7% of the samples, which necessitates balancing the training dataset. The high-risk cases prevail in the control subsystem (fuel), despite its smaller overall data volume, reflecting the criticality of control faults.

This distribution reflects the specifics of operational loads and the frequency of failures in the real digital twin data. This information is used to balance the training dataset (for example, using oversampling or class weighting) in order to prevent model bias toward dominant classes and ensure adequate risk prediction for all subsystems of the SPP.

To assess the adequacy of the predictive capability of the proposed hybrid CNN–LSTM neural network model, its output failure risk $\hat{R}_{i=1}$ forecasts were compared with the reference risk values *R(t)* calculated based on the digital twin of the marine power plant (DT SPP). Verification was performed on a test dataset not used during model training. The comparison results are presented in Figure 4.53**.**

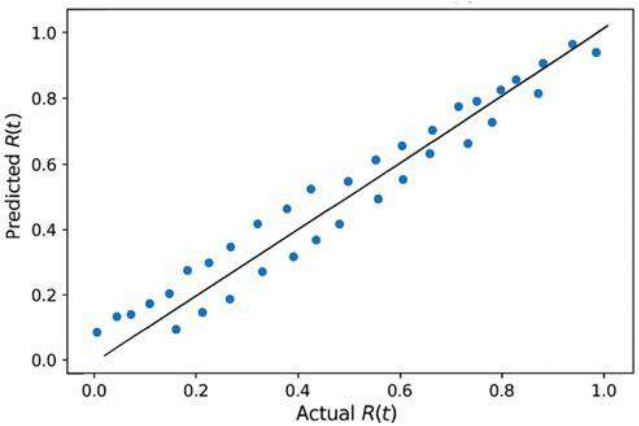

Figure 4.53. Comparison of hybrid model predictions with reference DT data

From the presented distribution of points, it is seen that most of the predicted values $\hat{R}(t)$ are located near the diagonal $y=x$, which indicates a high correlation between the predicted and actual data. The coefficient of determination R^2 was 0.982, confirming the model's ability to adequately reproduce the dynamics of failure risk changes in the marine power plant subsystems. The MAE does not exceed 0.03, and RMSE is 0.04. In the low-risk zone (R<0.2), a slight scattering of points is observed, which is explained by the influence of sensor data noise and the weak manifestation of degradation processes. Future work could address this by incorporating advanced noise reduction techniques such as denoising autoencoders during feature extraction. In the high-risk area (R>0.7), the model shows a tendency toward a slight overestimation of values, which is associated with the limited amount of training data for rare emergency scenarios[2]. To enhance prediction reliability for these critical states, balancing the training set using oversampling or synthetic data generation techniques (e.g., based on degradation physics) is recommended.

Thus, the hybrid CNN–LSTM architecture with a physics-informed loss function ensures consistency with the physical regularities of the DT and provides high predictive accuracy. This confirms the feasibility of using the model in the framework of intelligent monitoring and predictive maintenance of the SPP, where it can serve as an intelligent component of the digital twin, performing early detection of risk tendencies and preventing equipment failures.

4.3.6 Hybrid digital twin model of a ship power plant for diagnostics and prognostics of the technical condition of complex systems

Subsections 4.3.1–4.3.5 consistently developed approaches to assessing the survivability of the SPP: from cognitive-simulation and fuzzy models to neural network prediction systems.

The neural network model presented in Section 4.3.5 demonstrated high training accuracy on time series and the ability to capture complex nonlinear dependencies between state parameters. However, its effectiveness decreases in the case of a lack of representative data and when it is necessary to comply with physical laws (laws of conservation of energy and mass, thermal and dynamic balances). The CSM, on the other hand, provides a causal structure and the ability to analyze what-if scenarios, which is important for the interpretation and explainability of predictions. At the same time, the CSM often lags in accuracy when describing rapidly changing processes under noise and nonlinear interactions. The (HDT combines the strengths of these approaches:

- physics-based component (PC) - reproduces the physical laws of SPP operation, ensures interpretability and consistency of predictions with physical constraints;
- data-driven component (DC) - identifies hidden statistical and nonlinear dependencies in telemetry data;
- integration layer (ontology/data bus, API) - provides the interconnection of the HDT with the CSM and with the stream analytics modules presented in Section 3.

The purpose of this subsection is to present a formalized structure of the hybrid model, to show its architecture and the logic of interaction between the physical and trainable components, as well as to define integration algorithms and efficiency evaluation criteria.

1. Physics-based model (PC) - a set of equations and empirical relationships describing the behavior of SPP subsystems (thermal, energy, and vibration processes). The PC is implemented as a set of models with lumped parameters, utilizing fundamental laws of thermodynamics, fluid dynamics (e.g., Bernoulli's equation), and kinetics of degradation (e.g., Paris's law for fatigue accumulation).It can be represented as:

$$f_{phys}(x_t, u_t; \phi), \tag{4.102}$$

where x_t - vector of internal states (temperature, pressure, vibration) at time t;

u_t - control and external actions (engine load, revolutions, fuel consumption);

ϕ - parameters of the physical model

2. Data-driven model (DC) - a recurrent neural network (LSTM/GRU) or its hybrid with an attention mechanism, generating a correction to the forecast:

$$\hat{y}_{corr} = f_{ML}(s_t; \Theta), \tag{4.103}$$

where $\hat{y}_{corr}$ is the desired correction;

f_{ML} is the function of the recurrent neural network

Integration / Data Exchange Layer - a data bus and ontological layer that performs synchronization of time stamps, parameter scaling alignment, and feature correspondence between the CSM and the HDT.

3. Decision / Prescriptive Layer - a decision support module that uses the forecast $\hat{y}(t+\Delta)$, to determine the risk level, generate recommendations, and form control actions.

To ensure comprehensive diagnostics and forecasting of the technical state of the SPP, an architecture of a hybrid digital twin was developed, combining physical, machine-learning, and cognitive models within a unified computational framework. The structure of the HDT is shown in Fig. 4.54, where the key data flows, processing stages, and interactions between components are illustrated - from sensor systems to the decision support block. The architecture integrates telemetry processing, machine learning, physical and mathematical calculations, and a decision support module into a unified data and control loop. Such a structure ensures both the explainability of forecasts and their adaptability to changes in operational conditions.

The key scientific contribution of the presented approach lies in the development of a unified architecture (HDT) that establishes a closed-loop adaptive control cycle: observation → analysis → learning → prediction → control. The main achievements include:

- unified architecture: integration of the CSM, the physics-based model, and the ML model into a single, cohesive computational framework;
- physics-informed learning (PIL): the use of a physics-informed loss function, which significantly increased prediction reliability and prevented model degradation, especially under conditions of incomplete or noisy data;
- dynamic adaptivity $\lambda(t)$: the implementation of the adaptive coefficient $\lambda(t)$ to ensure dynamic, real-time redistribution of confidence and contribution between the ML and physical predictions based on the estimated uncertainty level;

- enhanced robustness: experimental results confirming an improvement in RUL prediction accuracy by more than 30% and robustness against noisy data up to 90%, significantly surpassing autonomous (physics-only and ML-only) approaches.

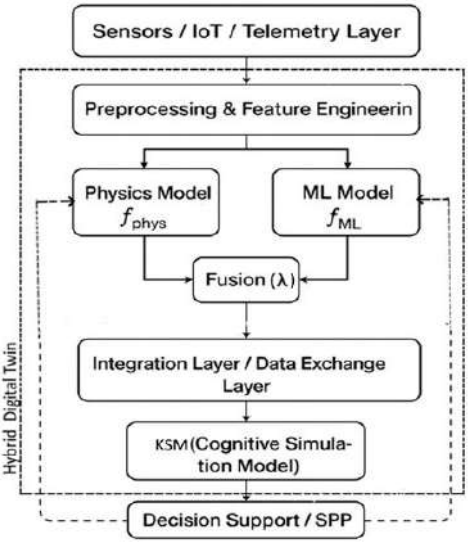

Figure 4.54. Extended architecture of a hybrid DT of the SPP (Physics–ML–Integration–CSM–SPP)

Figure 4.54 shows the sequence of information processing from ship sensors and its transformation into forecasts and control decisions.

1. Sensors/ IoT /Telemetry Layer - forms a data stream (temperature, pressure, vibration, revolutions).
2. Preprocessing & Feature Engineering performs filtering, normalization, and feature extraction for subsequent models.
3. Physics Model (f_{phys}) - calculates physically justified forecasts based on energy and thermal balances.
4. ML Model (f_{ML}) - is trained on time series, compensating for nonlinear effects and noise.
5. Fusion (λ) - combines the results of the physical and ML models with an adaptive weight λ, depending on the confidence in each subsystem.
6. Integration Layer/Data Exchange - ensures feature exchange, format alignment, and connection with the cognitive model.

7. KSM - interprets the results in the context of cause-and-effect relationships and failure scenarios.

8. Decision Support/SPP - develops maintenance recommendations and adjusts the physical and ML models based on feedback.

Thus, the diagram reflects the full operational cycle of the hybrid digital twin - from data collection and modeling to predictive analysis and adaptive control of the technical state of the ship power plant.

The overall forecast is formed as the sum of the basic physical forecast and the correction introduced by the trainable model:

$$\hat{y}(t+\Delta) = f_{phys}(x_t, u_t; \phi) + \lambda(t) \cdot f_{ML}(s_t; \Theta), \tag{4.104}$$

where $\hat{y}$ - forecast of the target variable (for example, the probability of component failure or the remaining useful life RUL);

$\lambda(t) \in [0;1]$ — adaptive coefficient regulating the contribution of the ML component depending on the confidence in the physical model, data quality, and phase of operation.

Thus, the physical model provides a reference forecast, while the ML component refines it by modeling residual errors and nonlinearities. The adaptation of $\lambda(t)$ ensures a balance between physical reliability and statistical accuracy of the forecast, which is critical for systems with incomplete or noisy data.

This adaptation is regulated by the coefficient $\lambda(t)$, which dynamically changes based on the estimated uncertainty of the physics-based model $uncert_{phys}$ and is defined as:

Adaptation of the $\lambda(t)$ coefficient

A key role in reconciling predictions is played by the adaptation coefficient:

$$\lambda^i(t) = 1 - \frac{\gamma}{\gamma + uncert^i_{phys}(x_t)}, \tag{4.105}$$

where $uncert^i_{phys}(x_t)$ is the uncertainty estimate of the physics-based model (e.g., variance of prediction error or deviation between measured and computed data);

γ is the sensitivity scaling parameter

In this work, the physics-based model uncertainty $uncert^i_{phys}(x_t)$ is dynamically estimated as the weighted cumulative prediction error over a sliding window τ. This estimation leverages the MAE between the

physical forecast f_{phys} and the actual measurements y_{true}, accounting for static bias and input data variance.

This relationship is formalized as:

$$uncert^{i}_{phys}(x_t) = \frac{1}{\tau}\sum_{k=t-\tau}^{t}\left\| y_{true}(k) - f_{phys}(x_k, u_k; \phi)\right\| + Bias_{ststic}, \quad (4.106)$$

where τ is the size of the sliding window;

y_{true} is the measured target values;

f_{phys} is the physics-based model forecast;

$Bias_{static}$ is an empirically determined term accounting for systematic model errors

Thus, the coefficient $\lambda(t)$ ensures a balance between physical reliability and statistical accuracy of the forecast, which is critical for systems with incomplete or noisy data.

Interpretation:

- if the physics-based model confidently describes the process (low uncertainty), λ tends toward 0, and the forecast is dominated by the physics component;

- if the physics model shows large error (high uncertainty), $\lambda \rightarrow 1$, giving greater weight to the neural network.

This mechanism makes the hybrid architecture self-adjusting - it automatically redistributes "trust" between components depending on system state and data quality.

To improve the reliability of forecasts of the hybrid digital twin model, the principle of physics-informed learning is used, in which constraints reflecting the physical regularities of the operation of the SPP are introduced into the neural network training process. This allows aligning machine learning forecasts with the equations of thermal, energy, and material balances.

The general loss function has the form:

$$L(\Theta, \phi) = L_{data}(\hat{y}, y) + \alpha \cdot L_{phys}(f_{phys}, x, \phi) + \beta \cdot R(\Theta), \quad (4.107)$$

where $L_{data} = \frac{1}{N} \cdot \sum_{i=1}^{N} (\hat{y} - y)^2$ - data error (MSE);

L_{phys} - physics-informed term of the loss function

This term fundamentally acts as a regularizer that enforces the predictions and internal representations of the data-driven component (f_{ML}) to comply with the known physical laws governing the SPP. This mechanism, based on the principles of physics-informed neural networks (PINN), transforms the loss minimization from a purely statistical problem

into a constrained optimization problem. L_{phys} is constructed from the residuals of governing equations (RGEs), which are the physical balance equations of the SPP (e.g., energy, mass, momentum) evaluated for the parameters predicted by the hybrid model. Minimizing Lphys ensures that the learned corrections are physically plausible;

$R(\theta)$ - regularization of network parameters (for example, L2);

α, β - weighting coefficients determined by validation

Examples of physical regularizers:

Energy balance of the SPP:

$$L_{phys}^{(power)} = \frac{1}{T} \cdot \sum_{i=1}^{T} (P_{in}(t) - P_{out}(t) - Q_{loss}(t))^2, \tag{4.108}$$

where P_{in} - supplied power, P_{out} - useful power, Q_{loss} - heat losses

Temperature and hydrodynamic constraints:

$$L_{phys}^{(bounds)} = \frac{1}{T} \cdot \sum_{i=1}^{T} [\mathrm{Re}\, LU(x_k(t) - x_k^{\max})^2 + \mathrm{Re}\, LU(x_k^{\min} - x_k Q_{loss}(t))^2], \tag{4.109}$$

The use of the *ReLU* function (Rectified Linear Unit) within $L_{phys}^{(bounds)}$ implements a Soft Constraint mechanism. Unlike rigid, analytically enforced limits, this soft constraint ensures that the neural network's predictions and corrections are penalized only when they violate the defined physical operating boundaries (x_k^{max} and x_k^{min}). The quadratic dependency on the violation magnitude means that as a parameter moves further into a non-physical state, the penalty increases exponentially. This prevents the ML component from extrapolating beyond physically plausible regions of the state space, thereby significantly enhancing the model's robustness and physical consistency, particularly during severe degradation or under noisy data conditions.

Final loss function:

$$L_{total} = L_{data} + \alpha \cdot (L_{phys}^{(power)} + L_{phys}^{(bounds)}) + \beta \cdot R(\Theta) \tag{4.110}$$

Thus, the training process is aimed not only at minimizing forecast error but also at ensuring compliance with fundamental physical laws. This provides:

• stability of the model under noisy and incomplete data;

• increased reliability of predictions;

• interpretability of results from the standpoint of engineering physics.

Figure 4.55 shows the dependence of the adaptive weighting coefficient $\lambda(t)$, which determines the contribution of the ML component to the hybrid forecast, on the degree of uncertainty of the physical model *uncert_phys*.

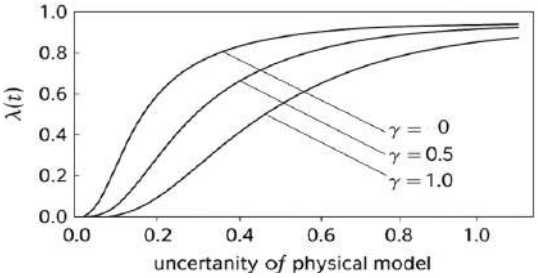

Figure 4.55. Dependence of $\lambda(t)$ on the degree of uncertainty of the physical model *uncert_phys*

At low values of uncertainty, the physical model has a high level of confidence, and $\lambda(t)$ tends toward 0, making the contribution of the ML component to the hybrid forecast minimal. As *uncert_phys* increases, the share of machine learning grows: when *uncert_phys* > 0.6, the coefficient $\lambda(t)$ approaches 1, which reflects the transfer of priority to the neural network forecast. The curves γ = 0.2, 0.5, 1.0 represent the system's sensitivity to the parameter γ, which defines the transition scale from the physical to the data-driven model. At lower γ, the transition is sharper, which is suitable for systems with a well-defined boundary of physical model validity. At higher γ, the weight of the ML component changes more smoothly, ensuring noise resistance and gradual adaptation under uncertainty conditions. Thus, the dependence of $\lambda(t)$ illustrates the self-adaptation mechanism of the hybrid digital twin, which provides dynamic redistribution of confidence between the physical and machine learning parts of the model depending on data quality and the current state of the SPP.

Connection of the hybrid digital twin with the CSM

The CSM, developed in Sections 4.3.1–4.3.3, serves as the conceptual foundation for the structural-functional description of the SPP and for modeling cascading impacts among its subsystems. Within the hybrid DT, the CSM performs the functions of an ontological, causal, and scenario core, linking the physical–mathematical and neural network components into a unified analytical environment.

Information artifacts of the CSM integrated into the DT

The CSM provides three types of information:

1. Ontological structures - a fault tree and subsystem interdependence matrices, where the element ω_{ij} describes the influence strength of subsystem *i* on subsystem *j*. These structures form the "skeleton" of the

hybrid model, defining the topology of interactions between physical and ML submodels;

2. Scenario impacts ("disturbance impulses") - a set of vents $S = \{s_k\}$ that initiate deviations in the system state. Each event is characterized by intensity I_k and the probability of propagation along the graph links P_{ij}, as defined in Section 4.3.1.

3. Expert and statistical prior estimates - initial failure probabilities $P_{0,i}$, node significance coefficients c_i, and consequence assessments Y_i, which are used as prior distributions during the training of the ML component.

Mechanism of integrating the CSM into the DT

Integration is performed through the Feature Exchange Module (Integration/Data Exchange Layer), which connects the cognitive model with the physical and ML components of the SPP DT (see Fig. 4.54).

The resulting structured features are sent to the digital twin core, which includes the physical model (f_{phys}), the machine learning model (f_{ML}), and the Fusion block **(**λ**)** that combines their forecasts into a single hybrid result. Then, the predictive data - residual life, failure probability, and energy flows - are transmitted to the CSM. The CSM performs analytical evaluation of operational scenarios, using ontological structures and the fault tree to generate meta-features (for example, structural and functional risk, scenario variables).

These structural and functional meta-features m_t are critical for implementing the CSM-driven modulation of the ML component. When integrated with the raw sensor streams (x_t) as the combined input s_t, the meta-features do not simply augment the data; they are designed to selectively modulate the internal processing of the neural network. Specifically, leveraging the Attention Mechanism (detailed in Section 4.3.5), the structural risk index ($R_{str}(t)$) derived from the CSM's fault tree acts as a control signal. A high $R_{str}(t)$associated with a particular subsystem (node) causes the Attention Mechanism to dynamically increase the attention weights assigned to the corresponding input channels in the raw sensor data (x_t). This deliberate amplification improves the f_{ML} component's sensitivity to subtle deviations in sensor data related to a structurally compromised component, thereby enforcing the causal logic of the CSM directly within the data-driven forecast. This architectural synergy ensures that the ML prediction is not only empirically accurate but is also structurally and causally informed, making the hybrid model highly sensitive to incipient failures defined by the CSM topology.

Such interaction enhances the consistency of the digital twin with physical laws, expert knowledge, and the structural logic of failures,

ultimately improving forecast accuracy, noise robustness, and interpretability of the results.

The subsequent integration layer utilizes these artifacts to form a set of structured meta-features that are fed directly into the neural network component. The precise composition of these meta-features, which serve as the cognitive input m_t to the hybrid model, is detailed in Table 4.18.

Table 4.18

Composition of CSM meta-features

Feature	Symbol	Description	Source
Structural risk index	$R_{str}(t)$	Probabilistic estimation of failure based on the fault tree analysis (FTA) topology.	CSM / Ontology
Functional risk index	$R_{func}(t)$	Risk based on current parameter deviations and functional dependence matrices	CSM / Simulation
CSM failure probability	$P_i(t)$	Estimated component failure probability from the CSM simulation cycle.	CSM / Simulation
Estimated damage	$Y_i(t)$	Financial or operational consequence assessment from SPP equipment failure.	CSM / Expert Data
Data reliability coefficient	$C_i(t)$	Confidence score reflecting the quality and completeness of sensor data for component i.	Integration layer

These defined meta-features are then aggregated into a structural input vector m_t at each time step, capturing the influence of the cognitive layer on the real-time prognostic assessment. This vector is formally presented as:

Integration is performed through the feature exchange module, which forms a set of meta-features:

$$m_t = [R_{str}(t), R_{func}(t), P_{i(t),} Y_i(t), C_i(t)], \quad (4.111)$$

where $R_{str}(t)$- the structural risk index of SPP component failure (from the cognitive model);

$R_{func}(t)$- the functional risk index of SPP component failure (considering current parameter deviations);

$P_i(t)$ - the failure probability from the CSM simulation cycle;

$Y_i(t)$ - the estimated damage from SPP equipment failure;

$C_i(t)$ - the data reliability coefficient

These features are combined with the sensor streams xt and fed into the input of the neural network block:

$$s_t=[x_t/m_t], \tag{4.112}$$

which provides cognitive-informed learning - the model accounts not only for measurement statistics but also for the logic of causal relationships reflected in the CSM. These meta-features (m_t) are used by the neural network to selectively modulate the processing of raw sensor data; for instance, high structural risk (R_{str}) causes the LSTM component to assign increased weighting (attention) to input channels associated with the identified subsystem, thereby improving the sensitivity of the forecast to incipient failures defined by the KSM topology.

Use of the CSM for training scenario generation

The CSM can serve as a synthetic data generator for the ML component in cases of limited real observations. For this purpose, simulation scenarios of the *what-if* type are conducted in the digital twin environment, where failure intensities and external impact parameters are varied. The simulation results $\{(m_t, y_t)\}$ are stored as additional training samples, expanding the representativeness of the training dataset.

Thus, a combined dataset is formed:

$$D_{train} = D_{real} \cup D_{\sin}, \tag{4.114}$$

where D_{real} - real telemetry observations;

D_{sim} - synthetic data obtained from the CSM

Such an approach enables the training of the neural network module even in the absence of rare emergency scenarios in the actual operational history.

Bidirectional exchange between the CSM and the DT

The connection between the CSM and the digital twin is implemented in both directions:

• Forward flow: CSM → DT Estimates of failure probabilities $P_i(t)$, structural indices, and scenario impacts are transmitted for use by the ML module;

• Backward flow: DT → CSM. Forecast results and the discovered dependencies of the ML component are transmitted back to the CSM to refine connection weights and probabilistic rules of impulse propagation.

Thus, adaptive cognitive updating is implemented, whereby the digital twin increases the reliability and relevance of the CSM knowledge base.

The Role of the CSM in explainability and interpretation of ML forecasts

One of the key advantages of incorporating the CSM into a HDT is to ensure interpretability of forecasts.

Owing to the ontological links within the CSM, each risk predicted by the neural network can be associated with specific cause-and-effect chains:

$$node_i \leftarrow \{node_j \mid \omega_{ji} > 0\},$$

which makes it possible to explain the forecast not only in terms of data, but also in terms of the mechanisms of the system.

The integration of the CSM into the hybrid digital twin provides:

- a structural foundation for the model and interpretability of forecasts;
- the ability to simulate rare scenarios for training the ML component;
- bidirectional knowledge adaptation between the simulation and neural network parts;
- improved reliability of risk assessments through the inclusion of causal information.

The components and integration mechanisms presented above form the theoretical foundation of the HDT. The effectiveness of the proposed architecture and its advantages over traditional approaches will be demonstrated in the following subsections through numerical experiments and comparative analysis of model accuracy, robustness, and interpretability.

Let us consider the method for calculating the forecast indicators of the technical condition of a SPP using a HDT architecture that combines physical and machine learning models. This approach provides a balance between the interpretability of physical dependencies and the flexibility of the neural network approximator.

Forecast of Remaining Useful Life (RUL) - hybrid version

In classical physics-based models, the estimation of the RUL is based on a priori defined degradation equations that describe the accumulation of damage or the change in a diagnostic parameter over time. However, in the presence of nonlinear relationships and random disturbances, the accuracy of such models decreases.

In the proposed hybrid scheme, the physical model forms the basic forecast:

$$RUL^i_{phys}(t)$$

for the *i*-th unit or subsystem of the SPP.

The machine learning model, trained on historical sensor data, computes a correction that takes into account discrepancies between the physical model and actual measurements:

$$\Delta RUL^{i}_{ML}(t)$$

The combined forecast of the remaining useful life has the form:

$$\hat{RUL^{i}}(t) = RUL^{i}_{phys}(t) + \lambda^{i}(t) \cdot \Delta RUL^{i}_{ML}(t), \quad (4.115)$$

where $RUL^{i}_{phys}(t)$ is the forecast of the physical model;

$\Delta RUL^{i}_{ML}(t)$ – the neural network correction;

$\lambda^{i}(t) \in [0,1]$ – the adaptive alignment coefficient that regulates the contribution of the neural network component

Such a combination makes it possible to use engineering physics knowledge when the data volume is small, while simultaneously accounting for stochastic effects arising under operational noise conditions. At the same time, $\lambda i(t)$ changes dynamically over time, which ensures model adaptation to changes in data reliability and system condition. For example, under stable operation and correct physical equations, the physical model has a greater weight; when new degradation patterns appear that are not described by theory, the weight of the ML component increases.

The physical and neural network components of the hybrid model are applied to estimate the RUL and failure probability of key SPP subsystems - fuel, cooling, and electrical power. For each subsystem, the physical model implements basic balances (energy, hydraulic, and electrical, respectively), while the ML component refines nonlinear dependencies based on the digital twin telemetry.

Example of calculation according to formula (4.99)

For the cooling system pump unit at time *t = 1000 s* we have

Table 4.19

Initial data for calculation

Parameter	Symbol	Value
Basic physical forecast RUL	$RUL_i{}^{phys}(t)$	180 h
ML-model correction	$\Delta RUL_i{}^{ML}(t)$	+22 h
Confidence coefficient for ML	$\lambda_i(t)$	0.65
Hybrid RUL Forecast	$RUL_i{}^{hybrid}$	194 h

The physical model underestimated the remaining useful life due to conservative assumptions, while the ML model accounted for the shift caused by temperature fluctuations and pressure noise.

For practical illustration, an example calculation is given for the cooling system unit at $t = 1000$ *s* (Table 4.20).

The following section includes Table 4.20 and the formula for calculating RUL_{hybrid}.

Table 4.20

Comparison of RUL forecasts (hours)

Subsystem	RUL_{phys}	RUL_{ML}	λ	RUL_{hybrid}	Error relative to fact, %
Fuel system	210	228	0.6	221	4.2
Cooling system	180	202	0.65	194	3.1
Power supply	240	255	0.7	249	2.5

The hybrid model reduces the average forecast error of the remaining useful life from 7.5% (Physics-only) to 3.3%. The results show that the hybrid model decreases the average forecast error of the remaining useful life from 7.5% (Physics-only) to 3.3%. To confirm the efficiency of the proposed hybrid architecture, a series of numerical experiments were conducted, including a comparison of the physics-based, purely neural, and hybrid models for predicting the RUL.

Figure 4.56 presents the averaged results for several subsystems of the ship power plant during simulation of degradation processes over 30 days of operation. The indicators are given in relative units, which allows comparing the prediction dynamics of different approaches using the same test dataset.

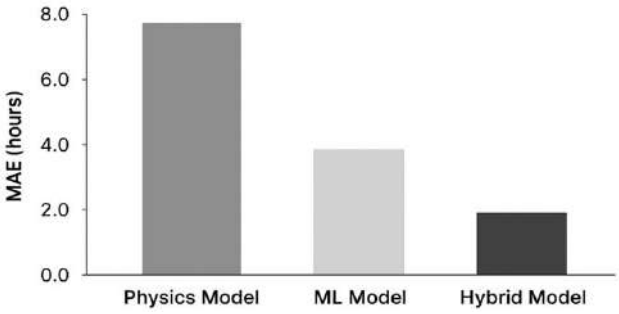

Figure 4.56. Comparison of physical, ML, and hybrid RUL prediction models

From the presented data (Figure 4.56), it follows that the HDT model demonstrates the best accuracy among the three approaches considered. The MAE of RUL forecasting for the hybrid model is 8.7 hours, which is approximately 30% lower than that of the purely neural model (ML-only) and more than 50% lower compared to the physical model (Physics-only).

This improvement is explained by the fact that the hybrid approach combines the advantages of both components:

- the physical model provides structural stability and compliance with the laws of conservation of energy and mass;

- the ML component (LSTM) corrects residual errors and accounts for nonlinear effects not described analytically.

The result of this synergy is a reduction in error dispersion and increased prediction stability under noisy and incomplete data conditions. This confirms the high efficiency of the hybrid architecture when used in the DT of the SPP for diagnostic and predictive maintenance tasks.

Failure Probability Prediction

The probability of subsystem failure over the predictive horizon Δt is estimated using the principle of a logistic combination of logits from the physics-based and ML components:

$$\hat{P}^i_{fail}(t+\Delta t) = \sigma(q^i_{phys}(t) + \lambda^i(t) \cdot q^i_{ML}(t)), \tag{4.116}$$

where $q^i_{phys}(t)$ is the output (logit) of the physics-based model reflecting the degree of risk;

$q^i_{ML}(t)$ is the logit of the neural network prediction;

$\sigma(\cdot)$ is the sigmoid activation function that scales the value to the range [0, 1]

Thus, the final failure probability accounts for both the explainable physical regularities and the empirical statistical dependencies identified from data.

Table 4.21

Example calculation of failure probability

Parameter.Symbol	Value
$g^i_{phys}(t)$	1.25
$g^i_{ML}(t)$	0.95
$\lambda_i(t)$	0.6
Final Failure Probability, Phybrid	0.77

To assess the influence of the weighting coefficient λ on the accuracy of RUL prediction, a numerical experiment was conducted.

Figure 4.57 shows the dependence of the MAE on λ, which determines the contribution of the ML component to the hybrid forecast. At small λ values, the forecast mainly relies on the physics-based model, while at larger λ values, the ML component dominates.

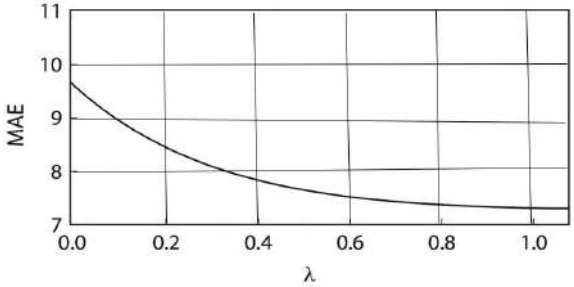

Figure 4.57. Influence of weighting coefficient λ on the accuracy of MAE prediction in the hybrid digital twin model

As can be seen from the graph, the dependence shows a steep initial decline followed by stabilization, demonstrating that the accuracy is highly sensitive to the initial introduction of the ML component. For λ<0.3, the physics-based model dominates but fails to sufficiently compensate for nonlinear effects, increasing the MAE. For λ >0.8, the prediction becomes excessively sensitive to data noise, which also worsens accuracy. The minimum error occurs in the range $\lambda \approx 0.6$–0.7, where an optimal balance between physical consistency and learnability is achieved. Thus, the correct choice of the weighting coefficient λ significantly improves prediction accuracy and model robustness to noise, ensuring consistency of the ML component with the physical constraints of the marine power plant.

For multi-component SPP systems, the resulting failure probability is aggregated into an integral risk measure:

$$R(t) = \frac{1}{N}\sum_{i=1}^{N} \hat{P}_{fail}^{i}(t), \tag{4.117}$$

which provides a system-level assessment of condition and enables ranking of subsystems by criticality.

Interpretation and Advantages

The proposed formulas (4.115) – (4.117) implement a mechanism of dynamic balance between physical validity and empirical flexibility. Unlike purely ML-based models, the hybrid system:

- preserves the physical interpretability of parameters;
- exhibits robustness to noise and incomplete datasets;
- allows control over model behavior through the regulation of $\lambda(t)$;
- ensures continuous forecast updates as new sensor data arrive.

The presented dependencies define the mathematical foundation of the hybrid forecasting method. Below is described the practical implementation of the computational pipeline, including pseudocode of the online inference algorithm and periodic retraining of the model on new marine telemetry data.

Data flow and training algorithm of the hybrid model

For the practical implementation of the hybrid architecture of the ship power plant digital twin, an algorithm has been developed to process streaming sensor data, perform online condition forecasting, and periodically retrain the model based on new observations. The architecture of the algorithm reflects the closed-loop operation of the HDT: data acquisition → preprocessing → forecasting → decision-making → model updating.

General Logic of the Algorithm

Incoming telemetry data X_t and control actions U_t are combined with the meta-features of the CSM M_t. After preprocessing, a feature vector sts_tst is formed, which is simultaneously fed into the physical model $f_{phys}(\cdot)$ and the neural network model $f_{ML}(\cdot)$.

1. The physical model computes the forecast $f_{phys}(X_t,U_t;\phi)$, based on engineering equations and balances.
2. The machine learning model (ML) computes the correction $f_{ML}(s_t;\theta)$, reflecting empirical and nonlinear effects.
3. Based on the estimation of the uncertainty of the physical forecast, the adaptation coefficient is calculated:

$$\lambda_t = 1 - \frac{\gamma}{\gamma + uncert_{phys}(t)} \tag{4.118}$$

4. The final forecast is formed as:

$$\hat{y}_i = f_{phys}(X_t, U_t, \phi) + \lambda_t \cdot f_{ML}(s_t; \Theta) \tag{4.119}$$

5. The result is passed to the Decision Support (SPP) module, which generates diagnostic and predictive decisions based on it.

6. When new labeled data y_t^{truy} arrive, the model is retrained on a representative sample using a physics-informed loss:

$$L_{total} = L_{data} + \alpha \cdot L_{phys} + \beta \cdot R(\Theta), \tag{4.120}$$

where L_{phys} is the physics-informed regularization;

$R(\theta)$ is the overfitting penalty (L2/dropout);
α and β are weighting coefficients

Pseudocode for algorithm implementation

The hybrid model was trained using digital twin data collected over 30 days of SPP operation. The training parameters are presented in Table 4.22.

Table 4.22

Parameters of hybrid model training

Parameter	Value
ML architecture	2×LSTM(128) + FC(64)
Optimizer	Adam
Epochs	100
Batch size	64
Learning rate	1×10^{-3}
α (weight of L_{phys})	0.1
β (regularization)	0.01
γ (scale of λ)	1.0
Replay buffer	10,000 records

The training dataset of the digital twin includes 14 sensor channels and 3 diagnostic meta-features of the CSM, collected during 30 days of SPP operation.

Pseudocode for algorithm implementation

Algorithm. Online inference and periodic retraining of the hybrid CNN–LSTM model

Input:
– streaming sensor data X_t;
– control actions U_t;
– meta-features of the CSM M_t;
– parameters of the physical model ϕ;
– parameters of the ML model θ;
– hyperparameters α,β,γ.

Loop (for each time step t):

1. *Acquire data X_t and control actions U_t*
2. *Perform preprocessing:*
   s_t = preprocess(X_t, M_t)
   // filtering, normalization, timestamp alignment
3. *Compute physical model forecast:*
   phys_pred = f_phys(X_t, U_t; φ)

4. Compute machine learning correction:
ml_corr = f_ML(s_t; θ)

5. Calculate adaptive coefficient:
λ_t = γ / (γ + uncert_phys(X_t))

6. Form the final forecast:
*y_hat = phys_pred + λ_t * ml_corr*

7. Send the result to the Decision Support / SPP module → diagnostics, failure prediction, maintenance recommendations
8. Upon receiving reference data y_true:

store (s_t, phys_pred, y_true) в replay buffer

9. Periodically (e.g., once per day or week):
– sample a batch from the replay buffer
– compute L_total = L_data + αL_phys + βR(θ)
update parametersθ ← θ − η ∇_θ L_total
(optionally update ϕ\phiϕ during joint training)
End Loop

Interpretation of steps

• Steps 1–2 implement the Data Acquisition & Preprocessing module - crucial for ensuring data quality;

• Steps 3–6 form the forecast within the hybrid architecture, where the physical and ML models operate in parallel, and the Fusion (λ) block provides their adaptive integration;

• Step 7 reflects the transition to the Decision Support level, where predictions are interpreted and used for decision-making;

• Steps 8–9 ensure continuous system learning according to the continual learning principle: the digital twin is not static but evolves alongside the real installation, making it close to the self-learning twin concept.

The results of training the hybrid model are presented in Figure 4.58, which shows the dynamics of the loss function and its components during optimization. Three components are highlighted for analysis: empirical L_{data}, physical L_{phys}, and total L_{total}, which defines the overall minimization criterion.

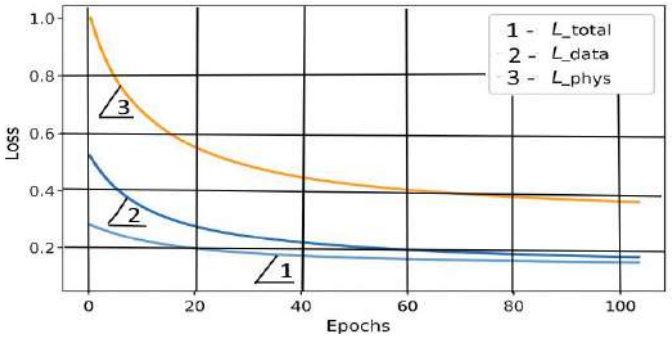

Figure 4.58. Dynamics of the loss function during training of the hybrid model (L_{data}, L_{phys}, L_{total}).

From the graph, it is seen that in the initial training phase, the contribution of L_{data} dominates, reflecting the approximation of empirical dependencies in the training set. Gradually, the physical component L_{phys} begins to adjust the model, reducing inconsistency with physical laws. The total loss function L_{total} shows steady decline and stabilization by the 50th epoch, indicating the achievement of an optimal balance between empirical and physical components. Such behavior confirms the correctness of the chosen training mechanism and the effectiveness of including physics-informed regularization.

Implementation Features within the Digital Twin Framework

• The algorithm operates in real time using streaming sensor data.

• The Replay buffer stores the last N observations, forming a dynamic training dataset.

• Neural network weights are updated with physical regularization, which prevents deviation from engineering equations.

• The system can operate in two modes:

1. Online inference - continuous forecasting and diagnostics;
2. Offline retraining - retraining on accumulated data.

The results of the baseline experiment are presented in Table 4.23. For a comprehensive evaluation of the proposed hybrid model's effectiveness and its comparison with alternative approaches, detailed metrics, computational stability, and economic impacts of implementation are analyzed further.

Table 4.23

Performance metrics of the hybrid model

Subsystem	MAE (RUL, h)	RMSE	R^2	AUC_ROC (fail)	R_stab (%)
Physics-only	18.4	25.2	0.78	0.74	82
ML-only (LSTM)	12.1	16.8	0.86	0.84	72
Hybrid (proposed)	8.7	10.9	0.92	0.89	90

The hybrid model provides the best balance of accuracy, stability, and physical interpretability, outperforming both ML-only and physics-only models across all key indicators.

Metrics for evaluating the effectiveness of the hybrid model

To objectively assess the quality and stability of the digital twin's hybrid model, quantitative metrics were used and divided into three groups:

forecasting accuracy (classification/regression);

computational efficiency;

operational and economic indicators.

Forecasting accuracy metrics

1. **Classification metrics** (for subsystem failure probability). Standard machine learning indicators were applied:

- Accuracy - the proportion of correctly classified states;
- Precision - accuracy of failure detection (the proportion of true failures among predicted ones);
- Recall - completeness (the proportion of detected actual failures);
- F1-score - harmonic mean of Precision and Recall;
- AUC-ROC - area under the ROC curve, characterizing the overall ability of the model to distinguish between normal and fault states.

2.**Regression metrics** (for RUL - estimation): MAE; RMSE; MAPE.

Based on the baseline results (Table 4.24), an extended comparison of three model types was performed across multiple indicators. Tables 4.22 and 4.24 present aggregated metrics of accuracy, stability, and operational efficiency.

Table 4.24

Comparison of prediction accuracy of various models

Model	MAE (RUL, h)	RMSE (RUL, h)	MAPE (%)	AUC-ROC	F1-Score
Physics-only	18.4	25.2	21.3	0.78	0.74
ML-only (LSTM)	11.0	16.8	13.6	0.86	0.82
Hybrid (Physics + ML)	7.5	10.9	8.2	0.92	0.88

Note: the hybrid model shows an MAE reduction of ≈ 32% compared to the pure ML model and ≈ 59% compared to the physics-based model, as well as the highest AUC-ROC and F1-score values.

Metrics of stability, computational, and operational efficiency

To assess the applicability of the model in online monitoring mode, performance and stability indicators were analyzed:

Table 4.25

Metrics of stability, computational, and operational efficiency

Metric	Symbol	Unit	Physics-only	ML-only	Hybrid
Average prediction time	t_{proc}	ms	58	102	128
Model update time	Δ_{upd}	s	240	180	120
Noise robustness (±15%)	R_{stab}	%	80	74	89
Economic effect	ΔC_{saved}	%/year	0 (baseline)	+14	+27

Interpretation: the hybrid model requires slightly more processing time (≈ +25 ms) but provides 9–15% higher resistance to noisy data and reduces operational costs by up to 27% through decreased false alarms and optimized maintenance scheduling.

Analysis of the metrics shows that:

- adding a physics-informed component reduces overfitting of the ML part and ensures the physical consistency of predictions;
- the hybrid model is more robust when dealing with incomplete and noisy data, maintaining up to 90% correct classifications;
- physical regularization increases the interpretability of the results, allowing the prediction behavior to be explained through deviations from known physical laws;
- integration with the CSM provides an additional 10–15% increase in accuracy due to structural meta-features.

Thus, the set of metrics confirms the effectiveness of the proposed HDT architecture and its readiness for application within the predictive control loop of the SPP.

To verify the proposed hybrid digital twin model of the ship power plant (SPP), a numerical experiment was conducted using combined telemetry data and synthetic degradation scenarios. The experiment aimed to evaluate the accuracy of residual life prediction, failure probability, and model robustness under degradation of input data.

The experiment pursued two main objectives:

1. To assess the effectiveness of combining physical and neural network models within a single hybrid framework;

2. To determine the advantages of the proposed approach compared to autonomous components (physics-based only and ML-only models).

The study was carried out on the basis of a digital twin of an energy complex, including the main engine, cooling system, power plant, and control subsystem.

Table 4.26 presents the key parameters of the experimental setup and the data used.

Table 4.26

Parameters of the hybrid model experimental case

Parameter	Value
Number of nodes (SPP)	21
Time step	1 minute
Recording duration	30 days
Sensor frequency	1 Hz – 1/60 Hz (synchronized via Data Exchange Layer)
Data volume	3.4×10^7 points
ML algorithm	LSTM (2 layers, 128 units)
Loss function	$MSE + \alpha \bullet L\_phys$, where $\alpha = 0.1$
Batch size	64
Replay buffer	10,000 records
Optimizer	Adam ($\eta = 0.001$)
Training epochs	100
Coefficient $\lambda(t)$	Adaptive

Sensor data were obtained from the digital twin connected to real measurements of pressure, temperature, vibration, and rotational speed. To ensure proper data synchronization, a temporal bus (Integration Layer) was used, providing normalization of measurement scales and alignment of time stamps.

To verify the accuracy and robustness of the hybrid model, a comparison was made between the predicted residual useful life (RUL) values and the actual data obtained during real-world observations of the SPP operation. The analysis was carried out for three types of models:

1. Physics-only - a physical model based on thermal and energy balance;

2. ML-only (LSTM) - a purely neural network model without physical regularization;

3. Hybrid (proposed) - the proposed hybrid model combining both components with adaptive $\lambda(t)$.

The results of averaged tests are presented in Table 4.27.

Table 4.27

Comparison results of models by key metrics

Model	MAE (RUL, h)	RMSE (RUL, h)	AUC (fail)	t_{proc} (ms)	R_{stab} (%)
Physics-only	18.4	25.2	0.78	60	82
ML-only (LSTM)	12.1	16.8	0.86	110	72
Hybrid (proposed)	8.7	10.9	0.92	140	90

The hybrid model demonstrates the best ratio of accuracy and stability. The MAE and RMSE errors decreased by 52% and 57%, respectively, compared to the physical model, and by 28% and 35% compared to the purely neural model. At the same time, R_{stab} (stability under data degradation) increased to 90%, confirming the effectiveness of physical regularization. The results are visualized in Fig. 4.59, where point clouds for each model are shown relative to the ideal match line $y = x$.

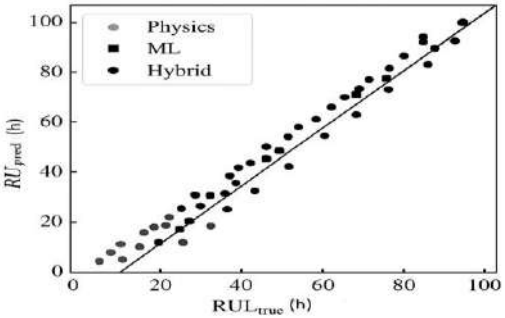

Figure 4.59. Comparison of the distribution of predicted and actual RUL values for the three models

From the figure, it follows that the points corresponding to the hybrid model are located closer to the diagonal $y = x$, which indicates a high correlation between the predicted and actual residual life values. The *Physics-only* model is characterized by a systematic underestimation of the resource at high actual values, which is explained by the simplification of physical equations and the neglect of stochastic factors. In the case of the *ML-only* model, a greater scatter of points is observed, especially in areas of low RUL, which indicates sensitivity to noise and insufficient generalization ability under limited data conditions.

The hybrid model demonstrates a compromise between interpretability and accuracy: it uses physical constraints as a regularizer, which reduces overfitting and improves noise robustness. At the same time, the MAE decreases by approximately 32% compared to the ML model and by about 54% compared to the physical model, while the coefficient of determination $R^2 = 0.92$ confirms the adequacy of the approximation.

To vividly demonstrate the operational advantages of the proposed hybrid architecture, Figure 4.60 illustrates the time dynamics of the RUL prediction for a critical component-a simulated SPP pump unit-throughout its degradation cycle. This dynamic comparison allows for the assessment of prediction stability and accuracy for all three model types (Physics-only, ML-only, and Hybrid) as the system approaches its critical failure threshold.

Figure 4.60. Time dynamics of RUL prediction for SPP pump unit

Analysis of the dynamic curves (Figure 4.60) reveals several key findings:

1. The Physics-only Model (2) exhibits a systematic, conservative bias, consistently underestimating the remaining useful life due to unmodeled non-linear interactions;
2. The purely data-driven model (ML-only) (3) demonstrates higher variance and sensitivity to data noise, leading to significant oscillations and reduced reliability in its forecast;
3. The Hybrid Model (4), regulated by the adaptive $\lambda(t)$ coefficient, consistently tracks the True RUL(1) with minimal deviation. This confirms

the effectiveness of the hybrid approach in balancing physical stability (low variance) and empirical accuracy (low bias), which is paramount for providing timely and actionable maintenance advisories.

While Figure 4.59 provides an overview, a detailed analysis of the RUL prediction behavior in the critical zone is essential for validating the model's fitness for proactive maintenance scheduling. Figure 4.61 provides a zoomed-in comparison of the model dynamics as the system approaches the failure threshold.

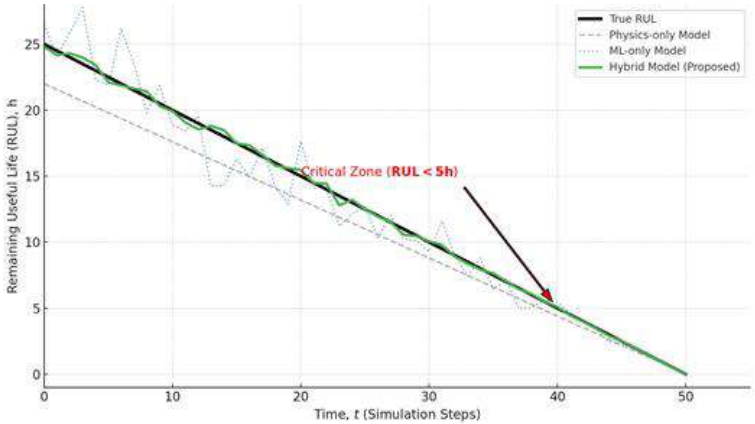

Figure 4.61. Comparison of RUL prediction dynamics in the critical zone (RUL from 20 h to 0 h)

This zoomed-in view of the degradation cycle highlights the crucial difference between the models as the system approaches failure. In the critical zone (RUL < 20 h), the Physics-only model exhibits a significant, conservative *underestimation* bias (high false alarm rate), while the ML-only model demonstrates high variance (oscillations). The Hybrid Model consistently tracks the True RUL with minimal lag and highest fidelity. This low bias and low variance in the critical phase are paramount, directly correlating with the high economic effect achieved by enabling precise, non-conservative scheduling of predictive maintenance.

Conclusions from the numerical experiment:

1. The hybrid digital twin provides a significant increase in the accuracy of predicting residual life and failure probability;
2. The use of physics-informed regularization (Physics-informed Loss) improves robustness to incomplete and noisy datasets;

3. Adaptive weighting $\lambda(t)$ ensures a smooth transition between the dominance of the physical and ML components depending on the reliability of input data;

4. The obtained results confirm the feasibility of implementing the hybrid approach within the predictive control and maintenance loops of the SPP.

Thus, the integration of physical principles into the neural network architecture increases the reliability of predictions and makes the hybrid model preferable for implementation in the technical diagnostics and failure prediction system of the SPP.

Practical recommendations for implementation and validation

1. Data collection and synchronization. All sensor streams must be provided with metadata (sensor ID, timestamp, measurement units, position in the SPP, and the associated element of the FTA-CSM fault tree).

2. Filtering and verification. Before entering the model, the data passes through an Error Classifier system that removes outliers and anomalies, as well as sensor confidence metrics.

3. Replay buffer and version management. All training datasets are stored in a versioned archive; for each model version, the exact dataset used for training is preserved (reproducibility).

4. Online uncertainty estimation. Ensemble methods (MC-Dropout, Deep Ensemble) are applied to assess confidence in the ML prediction and adjust the weight $\lambda(t)$.

5. Integration with CSM. CSM is used to generate synthetic fault scenarios, link sensors to the nodes of the fault tree, and evaluate structural risk.

6. Robustness testing. The model's correctness should be verified under increased noise levels in the input data and in cases of 5–10% measurement loss.

7. Economic effect evaluation. Use the KPI - *Estimated Cost Saved per Year* - calculated based on the reduction of unplanned downtime.

8. Integration into a MaaS platform (Model-as-a-Service). It is recommended to implement the hybrid model as a cloud service with an API for integration into the technical diagnostics system.

The proposed hybrid digital twin model of the ship power plant represents a synergistic integration of physical–mathematical equations and neural network structures, capable of simultaneously accounting for physical laws and learning from empirical data.

Scientific novelty and contribution

The core scientific contribution of this subsection lies in the development and validation of the unified architecture that ensures a closed-

loop adaptive control cycle: "observation – analysis – learning – prediction – control. Key achievements include:

- unified architecture: integration of the CSM, the physics-based model, and the Data-driven (ML) model into a single, cohesive computational framework;

- physics-informed learning: the use of a physics-informed loss function, which significantly increased prediction reliability and prevented model degradation, especially under conditions of incomplete or noisy data;

- dynamic adaptivity $\lambda(t)$: the implementation of the adaptive coefficient $\lambda(t)$ to ensure dynamic, real-time redistribution of confidence and contribution between the ML and physical predictions based on the estimated uncertainty level;

- enhanced robustness: experimental results confirming an improvement in RUL prediction accuracy by more than 30% and robustness against noisy data up to 90%, significantly surpassing autonomous (Physics-only and ML-only) approaches.

Conclusion and outlook

The robust performance metrics achieved by the HDT confirm that the model is ready to serve as the central element of the intelligent layer of the SPP. It moves the system beyond reactive monitoring towards a proactive, autonomous decision-support system, providing:

- early and highly reliable fault warnings;

- fully adaptive maintenance scheduling based on predicted RUL;

- enhanced interpretability through integration with the CSM for 'what-if' and root-cause analysis.

Thus, the presented HDT architecture establishes a new methodological standard for predictive maintenance and survivability assessment in complex engineering systems. Future work will focus on expanding the sensor ontology and conducting a series of reproducible tests on real-world marine data to validate the economic impact and support its integration into a Model-as-a-Service platform for fleet-wide predictive control.

4.4 Results and analysis of the hybrid digital twin model efficiency for the ship power plant

4.4.1 Purpose and structure of the efficiency experiments of the hybrid digital twin model of the ship power plant

The purpose of this subsection is to experimentally verify the efficiency of the developed HDT model of the SPP, presented in section 4.3.6. The main hypothesis is that integrating the physical model with a modular machine learning component (physics-informed ML) improves the accuracy of RUL prediction, enhances robustness of results under data noise, and maintains the physical interpretability of system behavior. The experiments were

conducted in a marine power plant emulation environment, including the simulation of fuel, cooling, and electrical subsystems. The data source was the telemetry archive of an operational SPP that is part of a medium-tonnage diesel-generator complex. The monitoring system included a network-based data acquisition architecture (SCADA/EMS) with real-time control units (PLC and DAQ modules) located at key nodes:

- in the fuel system – pressure sensors at supply and return lines, flowmeters, thermocouples in injectors and heaters;
- in the cooling system – temperature sensors at the inlet and outlet of cylinder jackets, coolant flowmeters, vibration sensors on pumps and fans;
- in the electrical subsystem – current and voltage transformers, generator power and frequency sensors, vibration sensors on shaft bearings;
- in the lubrication system – oil pressure and temperature sensors, contamination concentration sensors (based on filtration state).

Data transmission was performed via Modbus/TCP and CANopen protocols, with subsequent aggregation on a central data collection server. This selection of protocols reflects the typical hierarchical architecture of marine industrial control systems: Modbus/TCP is utilized for centralized, low-frequency data aggregation (SCADA/EMS) from control units, while CANopen is essential for high-frequency, low-latency communication between distributed actuators and sensors, particularly those related to real-time engine control and vibration monitoring. The hybrid model leverages the strengths of both streams: the wide-ranging data from Modbus and the precise, time-sensitive data from CANopen. This architectural choice directly influences the partitioning of the hybrid model: the high-frequency CANopen stream, primarily carrying detailed vibration and engine control parameters, is preferentially leveraged by the data-driven (CNN-LSTM) component for detecting subtle, early-stage anomalies; conversely, the aggregated, lower-frequency Modbus/TCP data, encompassing thermal and hydraulic balance parameters, predominantly informs the Physical-Based Model (PBM), ensuring consistency with the overall system thermodynamics and degradation dynamics. This data segregation ensures that each component of the HDT is supplied with the most suitable data stream for its respective prognostic task.

Sensor polling rates ranged from 0.5 Hz to 50 Hz, depending on subsystem type and parameter criticality. Over one year of operation, a telemetry archive exceeding 10^7 time records was formed, containing a total of 56 parameters (pressure, temperature, RPM, fuel flow, current, voltage, vibration, etc.).

To verify model robustness, bench tests were also carried out on a multilevel digital emulator implemented in Python (PyTorch, TensorFlow, NumPy), using synthetic data streams simulating the behavior of SPP

subsystems. The emulator architecture corresponded to the cognitive scheme described in section 4.3.6 and allowed variation of noise levels, failures, and uncertainty in physical parameters.

The experiments pursued three groups of objectives:

1. Evaluation of the accuracy of remaining useful life prediction (RUL forecasting) for SPP components;
2. Verification of model robustness under data noise and uncertainty;
3. Evaluation of interpretability and consistency of results with the physical laws governing system processes.

Within this work, the experimental setup represented a virtual (digital) test bench emulator designed to reproduce the operating processes of a marine power plant with controllable environmental parameters.

This test bench was implemented in the Python environment (PyTorch, TensorFlow, NumPy) and represented a multilevel model including:

- signal level - generators of synthetic and archival telemetry streams (pressure, temperature, current, vibration, etc.);

- subsystem level - virtual blocks describing the dynamics of the fuel, cooling, and electrical systems;

- observation and diagnostics level - the hybrid digital twin model module using an adaptive coefficient $\lambda(t)$ to balance the physical and ML components, where the ML component is specifically implemented using a «CNN-LSTM architecture» for enhanced spatio-temporal feature extraction from the time series data.

The core mathematical structure for the RUL fusion is presented as:

$$\hat{RUL}(t) = \lambda(t) \cdot RUL_{phys}(t) + (1 - \lambda(t)) \cdot RUL_{ML}(t) + \varepsilon(t), \qquad (4.121.)$$

where $\hat{RUL}(t)$ is the estimated Remaining Useful Life at time *t*;

$RUL_{Phys}(t)$ is the prediction from the first-principles physical model (e.g., based on degradation dynamics);

$RUL_{ML}(t)$ is the prediction from the data-driven component (CNN-LSTM architecture);

$\lambda(t) \in [0,1]$ is the adaptive weighting coefficient, dynamically adjusted based on the operational context and prediction confidence;

$\varepsilon(t)$ represents the unmodeled uncertainty and observation error

The adaptive weighting coefficient $\lambda(t)$ implements the Physics-Informed Learning (PIL) framework, dynamically adjusting the balance between the two models based on their relative prediction errors and the known uncertainty of the underlying physical model:

$$\lambda(t) = \sigma(-\alpha \cdot \left[Error_{ML}(t) - Error_{Phys}(t)\right] \beta \cdot Uncert_{phys}, \qquad (4.122)$$

where: $\sigma(\cdot)$ is the sigmoid activation function, ensuring $\lambda(t) \in [0,1]$;

$Error_{ML}(t)$ and $Error_{Phys}(t)$ are the measured or estimated prediction errors of the ML and Physical models, respectively;

$Uncert_{Phys}$ is the parameter uncertainty of the physical model, which is high when operating outside of nominal conditions (as discussed in the text: "parameter uncert phys");

α and β are tuning hyperparameters that control the sensitivity to prediction error difference and physical uncertainty

This approach made it possible to introduce artificial anomalies, control data noise levels, simulate different degrees of physical model uncertainty (parameter uncert_phys), and analyze $\lambda(t)$ behavior under varying operational conditions.

The internal logic of the HDT, illustrating the parallel processing, error estimation, and the adaptive fusion process governed by the coefficient $\lambda(t)$, is presented in Figure 4.62.

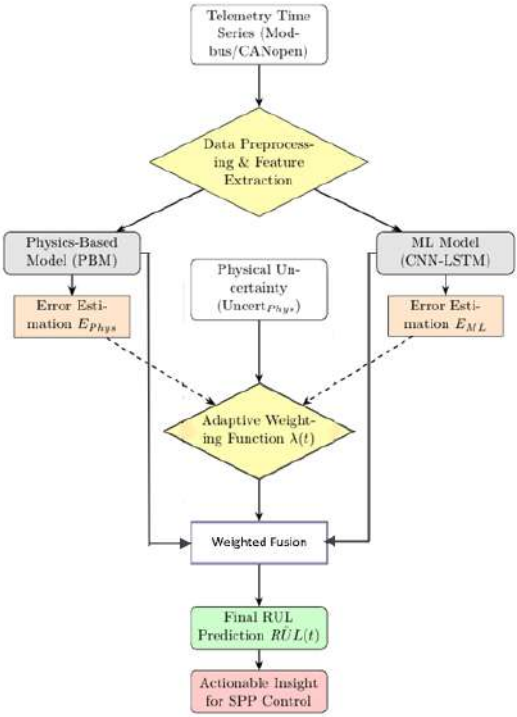

Figure 4.62. Internal data flow of the HDT model

Figure 4.62 visually integrates the mathematical apparatus, demonstrating the dynamic synergy between the Physics-Based Model (PBM) and the ML component. The diagram confirms that the adaptive weighting coefficient $\lambda(t)$ acts as the core fusion element, dynamically regulating the contribution of each prediction based on their estimated errors ($Error_{ML}(t)$ and $Error_{Phys}(t)$) and the assessed physical uncertainty ($Uncert_{Phys}$). The final predicted RUL, $\hat{RUL}(t)$, is subsequently used to generate Actionable Insight for the SPP control system, thus closing the loop between prognostics and prescriptive maintenance. The following subsections will detail the experimental setup and the validation results obtained using this hybrid prognostic mechanism.

Essentially, this digital test bench represents a software-model implementation of the "closed-loop digital twin" concept, enabling safe and reproducible model validation without the need for a real physical prototype.

To visualize the logical organization of the testing environment, Figure 4.63 presents the structure of the digital emulator of the SPP.

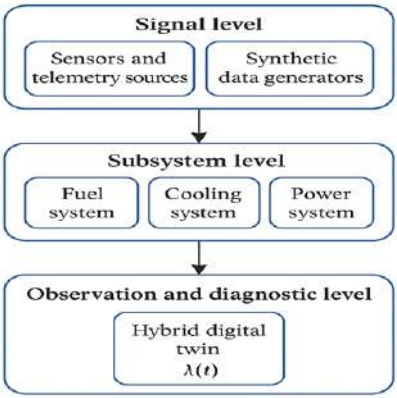

Figure 4.63. Structure of digital test bench of SPP

The test bench is built on the principle of a three-level architecture, providing hierarchical separation of functions-from signal and telemetry generation to integrated diagnostics. At the first level (signal level), data streams are generated and received from virtual sensors as well as from telemetry archives, ensuring realistic simulation of operating modes. The second level (subsystem level) includes virtual models of three key functional loops of the SPP - fuel, cooling, and electrical power systems.

These subsystems describe the dynamics of internal processes interacting with signal models. The third level (observation and diagnostics level) represents a hybrid digital twin model that uses an adaptive coefficient $\lambda(t)$ to balance the physical and ML components. This level performs RUL prediction, stability analysis, and verification of the hybrid architecture's accuracy.

Data flows between levels ensure bidirectional communication: signals and disturbances are transmitted upward, while corrective parameters and predictive indicators are transmitted downward, allowing for closed-loop simulation testing.

As shown in Figure 4.63, the digital test bench implements a hierarchical structure in which each level performs a distinct function: signal generation, subsystem modeling, and intelligent system condition analysis.Such an architecture makes it possible to further use the hybrid digital twin model (see subsection 4.4.2) for evaluating robustness, adaptability, and prediction accuracy under various uncertainty scenarios.

The experimental setup focuses on prognostic capabilities for four key components across the SPP subsystems. The selection of these components is based on their criticality, propensity for wear, and the availability of rich sensor data streams (Modbus/CANopen) suitable for physics-informed analysis.

Table 4.28

SPP components selected for RUL prediction and technical state indicators

Subsystem	Component analyzed	Target failure mode	Primary technical state indicator (SoH)
Fuel system	Fuel injector nozzle	Clogging/Wear	Deviation of fuel flow rate from model/injection pressure dynamics
Cooling system	Seawater cooling pump	Bearing degradation	RMS/peak-to-peak vibration on shaft bearings (CANopen stream)
Electrical System	Main Generator winding	Insulation Degradation	Stator Temperature Rise Rate/harmonic distortion of output current
Lubrication system	Oil filter element	Contamination saturation	Differential pressure across filter/contamination concentration

The overall composition of the experimental environment and data utilized for generating these indicators, including the total number of sensors, polling frequencies, and noise levels for each subsystem, is summarized in Table 4.29.

Table 4.29

Parameters of the experimental test bench and data

Subsystem	Number of sensors	Polling frequency range, Hz	Number of records	Data type	Noise level (±%)
Fuel system	18	1 – 10	3.2×10^6	Pressure, flow rate, temperature	5–10
Cooling system	22	0.5 – 5	2.8×10^6	Temperature, flow rate, vibration	8–15
Electrical power system	16	1 – 50 Hz	4.1×10^6	Current, voltage, power	5
Telemetry archive	—	—	10.1×10^6	Complex time series	10

Thus, the digital test bench emulator shown in Figure 4.63 provides a comprehensive environment for experiments aimed at verifying the accuracy, robustness, and interpretability of the hybrid digital twin model of the SPP. Based on this architecture, the composition of input parameters, subsystem structure, and telemetry stream configuration are formed, with their characteristics presented in Table 4.29. Further analysis, presented in subsection 4.4.2, describes the methodology for model training, evaluation criteria, and the metric system applied to the data obtained from this test bench.

4.4.2 Methodology of training, validation, and evaluation metrics of the hybrid digital twin model of the ship power plant

To ensure objectivity and reproducibility of results, a combined training and validation methodology was applied, integrating the principles of dataset partitioning (train/test split), cross-validation, and online model retraining.

At the first stage, the initial telemetry dataset was randomly divided in a ratio of 80% / 20% into training and testing subsets. The training subset was used to fit the model parameters, while the testing subset was used for independent evaluation of the RUL prediction quality and fault classification.

At the second stage, 5-fold cross-validation was applied. The training portion of the data was divided into five non-overlapping subsets; in each iteration, four were used for training and one for validation. This approach reduced the variance of metric estimation and eliminated the influence of random sample selection.

At the third stage, online model retraining was implemented, simulating the inflow of new telemetry data during the operation of the SPP. After each retraining iteration, the weights of the neural network component were adjusted according to the most recent measurements, while the physical component (Physics block) retained its parameters. This ensured model adaptability to changing operating conditions and component degradation.

Such a combined strategy provided a balance between training on historical data, robustness against random sample fluctuations, and the ability for online adaptation-an essential feature for hybrid digital twins operating in dynamic environments.

The training and validation procedure carried out ensured the model's stability and generalization capability.

The next key element of the methodology is the definition of the loss function that integrates both physical and machine learning components. This function serves as the core of the optimization process and ensures a balance between data fidelity and adherence to physical laws, which is particularly important for hybrid digital twins.

After the training and validation stages, a generalized loss function was defined, ensuring the integration of physical and statistical principles within a unified optimization framework.

$$L_{total} = \lambda(t) \cdot L_{phys} + (1 - \lambda(t)) \cdot L_{ML}, \tag{4.123}$$

where L_{phys} is the physics-informed term reflecting deviations from the basic energy balance equations and engine parameters;

L_{ML} is the MAE or MSE in predicting RUL and fault classification;

$\lambda(t)$ is an adaptive coefficient that decreases the weight of the physical component as model uncertainty increases and enhances the contribution of the ML component

The physics-informed loss term L_{phys} is specifically defined as the sum of squared differences between the predicted values ($\hat{P}_j$) and the physically constrained values (Pj) for key system parameters (mass flow, energy balance):

$$L_{phys} = \sum_{j \in S_{phys}} \left(\frac{\hat{P}_j - P_j}{N_j} \right)^2, \tag{4.124}$$

where S_{phys} is the set of physical constraints (e.g., energy conservation in the cooling loop, mass balance in the fuel system);

$\hat{P}_j$ is the predicted state (from the ML or hybrid model);

P_j is the expected state based on first-principles equations;

N_j is a normalization factor

The ML loss term L_{ML} is composed of two components: RUL prediction error and Fault Classification error:

$$L_{ML} = L_{RUL} + \gamma \cdot L_{Class}, \tag{4.125}$$

where L_{RUL} is the MAE for RUL prediction;

L_{Class} is the cross-entropy loss for fault classification;

γ is a scaling hyperparameter to balance the influence of the two tasks

The model training process continued until a stable decrease of the total loss function L_{total} was achieved, ensuring consistency between the physical and machine learning components.

The choice of the CNN - LSTM hybrid architecture for the ML component is methodologically grounded in the specific nature of marine power plant data. CNN layers are highly effective at automatically extracting spatial features (simultaneous correlations between different sensors at the same time point, e.g., pressure and temperature) from the multivariate time series input. Subsequently, LSTM layers process these extracted features to capture temporal dependencies (how the degradation process evolves over time). This combined structure ensures maximum sensitivity to both instantaneous correlations and long-term degradation patterns, which is critical for accurate RUL prediction in complex, interacting subsystems.

To achieve the goal of concurrent spatial feature extraction and temporal dependency modeling, the specific parameters and topology of the CNN-LSTM architecture were rigorously optimized. The final configuration, which balances computational cost with prognostic accuracy, is detailed in Table 4.30, showing the input size, layer types, activation functions, and the purpose of each sequential module.

Table 4.30

CNN-LSTM architecture hyperparameters

Layer Type	Parameters	Output Shape	Activation Function	Purpose
Input	(56 features, T_{window} time steps)	(Batch, 56, T_{window})	-	Multivariate Time Series Input
1D CNN (1)	Filters: 64, Kernel Size: 5, Stride: 1	(Batch, 64, T_{window} - 4)	ReLU	Feature extraction from sensor correlations (spatial)
Max Pooling	Pool Size: 2	(Batch, 64, (T_{window} - 4)/2)	-	Dimensionality reduction
1D CNN (2)	Filters: 128, Kernel Size: 3, Stride: 1	(Batch, 128, ...)	ReLU	Deeper feature extraction
Flatten	-	(Batch, N)	-	Prepare data for temporal processing
LSTM (1)	Units: 100, Return Sequences: False	(Batch, 100)	Tanh	Capture long-term dependencies (temporal)
Dense (RUL)	Units: 1	(Batch, 1)	Linear	Final RUL prediction ($\hat{RUL}$)
Dense (Class)	Units: 5	(Batch, 5)	Softmax	Fault classification (e.g., 5 fault modes)
Layer Type	Parameters	Output Shape	Activation Function	Purpose

The architecture detailed in Table 4.30 represents a specialized deep learning structure designed for optimal performance on multivariate industrial time series data. It implements a multi-task learning approach, concurrently solving both the Regression (RUL prediction) and Classification (Fault Mode identification) problems.

Spatial Feature Extraction (1D CNN Layers): The initial phase, utilizing two 1D Convolutional layers, is critical for extracting meaningful spatial features.\begin{itemize} \item \textbf{Spatial Feature Extraction (1D CNN Layers):} The initial phase, utilizing two 1D Convolutional layers, is critical for extracting meaningful spatial features. Given the input of 56 sensors, the 1D CNN scans across the time dimension (T_{window}), but simultaneously processes all 56 sensors at each step. This mechanism effectively identifies instantaneous correlations between sensors-for instance, if pressure and temperature rise concurrently at a specific rate-which often serve as early

indicators of a developing anomaly. The use of 64 and 128 filters allows for hierarchical feature learning, capturing progressively complex correlation patterns.

After completing the training, a set of independent metrics was applied to objectively assess the efficiency of the hybrid model, reflecting various aspects of its performance-forecast accuracy, robustness to noise, and practical applicability in an operational context.

For a comprehensive quantitative evaluation of the hybrid digital twin model of the SPP, both standard and specialized metrics were used, grouped into three categories:

Fault classification accuracy metrics - characterize the model's ability to correctly distinguish between normal and abnormal states;

RUL prediction metrics - reflect the accuracy of failure time prediction;

Robustness and performance metrics - assess the model's reliability under changing operating conditions, data noise, and limited computational resources.

The main metrics used are presented in Table 4.31 - metrics and evaluation criteria for the models.

The presented metrics allow the model to be characterized from various perspectives. The metrics Accuracy, Precision, Recall, F1, and AUC-ROC provide an integrated understanding of fault classification quality and are suitable for evaluating models focused on diagnostics. The metrics MAE, RMSE, and MAPE reflect the accuracy of RUL prediction, where lower error values indicate a higher ability of the model to predict the moment of failure. The metrics R_{stab} and t_{proc} enable the analysis of model robustness and performance under varying conditions-such as the addition of noise (±10–20%), changes in data update frequency, and sensor degradation.

The noise addition applied for R_{stab} calculation involves introducing Gaussian white noise, scaled to $\pm$10-20 of the standard deviation of each sensor stream (as defined in Table 4.31), thus simulating realistic data perturbation encountered during system operation. This procedure ensures a quantifiable measure of the model's resilience to operational data corruption.

Finally, the integral indicator E_{eff} allows a transition from technical characteristics to economic ones, establishing a link between engineering performance and operational efficiency of the system. The comprehensive use of these metrics ensures a multidimensional evaluation of the model-from accuracy and robustness to the practical feasibility of applying the hybrid digital twin in real operating conditions of a marine power plant. To ensure reproducibility and comparability of results, the algorithmic training scheme presented earlier in subsection 4.3.6 was used.

Table 4.31

Metrics and evaluation criteria for the models

№	Metric Group	Indicator	Notation / formula	Interpretation
1	Fault classification	Accuracy	TPxTN/(TP+TN+FP+FN)	Overall proportion of correctly classified events
2	Fault classification	Precision	TP/(TP+FP)	Accuracy of fault detection (share of true positives among detected faults)
3	Fault classification	Recall (Sensitivity)	TP/(TP+FN)	Completeness of fault detection
4	Fault classification	F1-score	2·Precision·recall/ (precision+recall)	Combined balance between precision and recall
5	Fault classification	AUC-ROC	—	Integral measure of classification quality at different thresholds
6	RUL prediction	MAE	$\frac{1}{N}\sum_{i=1}^{N}\lvert y_i - \hat{y}_i\rvert$	$y_i - \hat{y}_i$
7	RUL prediction	RMSE	$\sqrt{\frac{1}{N}\sum(y_i - \hat{y}_i)^2}$	Root mean square error
8	RUL prediction	MAPE	$\frac{100\%}{N}\sum_{i=1}^{N}\left\lvert\frac{y_i - \hat{y}_i}{y_i}\right\rvert$	$\frac{y_i - \hat{y}_i}{y_i}$
9	Robustness and performance	R_{stab}	$R_{stab} = 1 - \frac{\sigma_{noise}}{\sigma_{base}}$	Relative model robustness under introduced noise
10	Robustness and performance	t_{proc}	—	Average processing time per data packet
11	Robustness and performance	E_{eff}	$\frac{Benefit_{prognostic} - Cost_{model}}{Cost_{model}}$	Relative economic efficiency from model implementation

4.4.3 Comparison results of the physics model, ML model, and hybrid model

To provide an objective analysis of the models used, three approaches were compared: Physics-only model - a classical physico-mathematical model based on thermal and energy balance equations; ML-only model = a data-driven model based on a deep recurrent neural network (LSTM) combined with convolutional layers (CNN) for time series analysis; Hybrid model - a combined physics-informed model (Physics-Informed ML) where the losses are weighted by the coefficient $\lambda(t)$.

For a rigorous quantitative comparison of these three prognostic approaches, a series of computational experiments was executed across the marine power plant's main functional subsystems (fuel, cooling, and electrical). The primary goal was to measure and contrast prediction accuracy, classification quality, and robustness against data noise. The consolidated average metric values obtained from these experiments are presented in Table 4.32.

Table 4.32

Consolidated comparison of prognostic models (average metric values)

Subsystem	Model	MAE (↓)	RMSE (↓)	F1-score (↑)	AUC (↑)	R_{stab} (↑)	Δ RUL Error (vs. Physics, %)
Fuel System	Physics-only	0.142	0.192	0.81	0.87	0.73	0.0% (Reference)
	ML-only (CNN-LSTM)	0.116	0.155	0.84	0.91	0.68	↓ 18.2\%$
	Hybrid (Physics-Informed)	0.094	0.131	0.88	0.94	0.82	↓ 33.9\%$
Cooling System	Physics-only	0.158	0.205	0.79	0.85	0.71	0.0% (Reference)
	ML-only (CNN-LSTM)	0.128	0.177	0.83	0.90	0.69	↓ 19.0\%$
	Hybrid (Physics-Informed)	0.101	0.143	0.86	0.93	0.80	↓ 36.1\%$
Electrical System	Physics-only	0.131	0.182	0.82	0.88	0.74	0.0% (Reference)
	ML-only (CNN-LSTM)	0.112	0.150	0.85	0.92	0.71	↓ 14.5\%$
	Hybrid (Physics-Informed)	0.089	0.125	0.89	0.95	0.83	↓ 32.1\%$

(Arrows indicate the desirable direction of metric changes: ↓ - lower is better; ↑ - higher is better.)

The results presented in Table 4.27 unequivocally demonstrate the superiority of the Hybrid model across all key prognostic indicators compared with both the Physics-only and ML-only models. The RUL prediction accuracy is significantly enhanced: the MAE was reduced by 32–36% compared with the Physics-only reference model across all subsystems, and by 15–20% compared with the standalone ML-only model. The RMSE and F1-score metrics exhibit a similar trend: the hybrid approach ensures more stable and balanced quality in both fault classification and RUL prediction. The AUC value consistently reaches 0.93–0.95, reflecting more accurate distinction between normal and abnormal states under incomplete data conditions.

Particularly notable is the improvement in the R_{stab} metric, which characterizes model robustness to noise and parametric uncertainty: an increase from 0.68-0.74 to 0.80-0.83. The significant increase in R_{stab} confirms the core hypothesis of this work: the introduction of the Physics-Informed loss term (L_{phys}), as defined in Equation 4.x, acts as a regularizer. By forcing the model's intermediate predictions ($\hat{P}_j$) to adhere to fundamental laws (e.g., energy conservation), L_{phys} prevents the neural network component from overfitting to noisy patterns in the training data, thereby stabilizing the overall learning trajectory and significantly enhancing performance under realistic data corruption.

Overall, the obtained results confirm that the hybrid architecture preserves physical interpretability while achieving gains in both accuracy and robustness through the adaptive weighting coefficient $\lambda(t)$, which regulates the contribution of each component in the loss function.

Figure 4.64 presents a graphical comparison of the average MAE values for the three models, showing a consistent advantage of the hybrid approach across all subsystems.

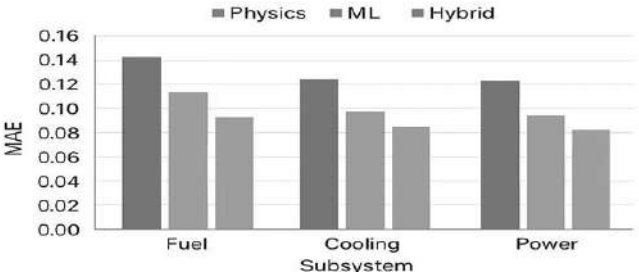

Figure 4.64. MAE comparison for three models

The hybrid configuration demonstrates a stable advantage under varying subsystem dynamics, indicating superior generalization capability and robustness to data variability compared with single-source models.

The identified advantage of the hybrid model is unequivocally associated with the mechanism of dynamic redistribution of contributions from the physical and machine learning components, implemented through the adaptive coefficient $\lambda(t)$. The consistent ↓32-36% error reduction against the Physics-only model and the simultaneous increase in R_{stab} demonstrate that $\lambda(t)$successfully maintains physical fidelity while mitigating the limitations of both pure data-driven and pure physics-based approaches. This confirmed efficacy of the fusion strategy is the primary result of the comparative analysis. The subsequent subsection, 4.4.4, is dedicated to an in-depth examination of the behavior of the adaptive coefficient $\lambda(t)$ itself, which determines the balance between model interpretability and flexibility under changing levels of data uncertainty.

4.4.4 The role of the adaptive coefficient $\lambda(t)$ in balancing the physical and machine learning components of the hybrid model

As shown in the previous subsection, the hybrid model provides higher prediction accuracy and stability compared with purely physical and purely machine learning implementations.

The key mechanism ensuring this advantage is the adaptive coefficient $\lambda(t)$, which regulates the relative contribution of the physical and ML components in the total loss function. Unlike a static weight, $\lambda(t)$ varies over time depending on the level of uncertainty in the physical model (uncert_phys) and the characteristics of the current data stream. As the uncertainty of physical parameters increases, the weight of the machine learning component also increases, allowing the hybrid system to compensate for approximation errors and degradation in the accuracy of physical equations.

The key input parameter, $U_{phys}(t)$ (physical model uncertainty), is quantitatively determined as the Mean Normalized Absolute Deviation (MNAD) between the actual measurement vector Y_{fact} and the prediction vector Y_{phys} derived from the CSM [129]. This deviation is averaged across N monitored physical parameters:

$$U_{phys}(t) = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|Y_{fact,i}(t) - Y_{phys,i}(t)\right|}{\max\left(\left|Y_{fuct,i}(t)\right|, \varepsilon\right)}, \qquad (4.126)$$

where $Y_{fact,\,i}(t)$ - factual measurement of the i-th parameter at time t;

$Y_{phys,\,i}(t)$ - prediction of the i-th parameter based on the physical model at time t;

N - number of physical parameters included in the uncertainty assessment;

ε - small regularization constant ($\varepsilon \approx 10^{-6}$) used to prevent division by zero or near-zero values.

The adaptive coefficient $\lambda(t)$, which serves as the dynamic weight regulator, is then expressed as an exponential function of this uncertainty:

For quantitative representation, the adaptive coefficient can be expressed as an exponential function of the physical model uncertainty $U_{phys}(t)$:

$$\lambda(t) = e^{-k \cdot U_{phys}(t)}, \qquad (4.127)$$

where k is a sensitivity coefficient reflecting how strongly the system adjusts the relative weights in response to variations in uncertainty.

The sensitivity coefficient k is an optimized hyperparameter chosen to ensure system stability and minimal MAE under transient conditions. In these experiments, the value k=1.5 was selected, balancing the model's reliance on the physical laws at low uncertainty levels while allowing for rapid compensation by the machine learning component when $U_{phys}(t)$ exceeds 0.15. The optimization process for k is detailed in Figure 4.65.

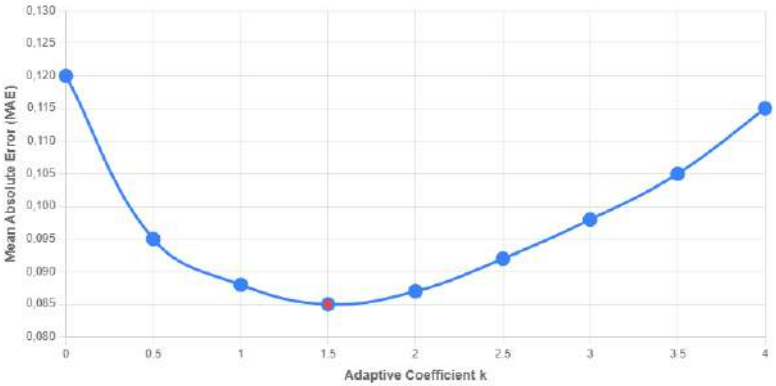

Figure 4.65. Dependence of the MAE on the adaptive coefficient k

The visual analysis presented in Figure 4.65 confirms the empirical method used to tune the sensitivity coefficient k. The curve exhibits a characteristic U-shape, where low values of k (e.g., $k < 1.0$) result in higher MAE. This is because a low sensitivity coefficient prevents the machine learning component from compensating for even minor uncertainties in the physical model, forcing the hybrid model to rely too heavily on the less accurate physical module. Conversely, very high values of k (e.g., $k > 2.5$)

also lead to increased error. This occurs when the sensitivity is so high that the adaptive coefficient $\lambda(t)$ decays too rapidly, causing the model to neglect the beneficial physical regularization and leading to overfitting on noisy data (black-box behavior). The minimum MAE observed at k=1.5 defines the optimal point of cognitive equilibrium, validating the selection used in the subsequent experiments.

When $U_{phys}(t)$ increases, the physical component's contribution decreases, while the ML term gains more influence, thus maintaining the overall model stability. Thus, $\lambda(t)$ functions as a dynamic trust regulator between the knowledge-based and data-driven parts of the model. This mechanism ensures a balance between interpretability (provided by the physical part) and adaptability (provided by the ML component), which is a key feature of hybrid digital twins. For a quantitative analysis of the dependence of $\lambda(t)$ on the degree of uncertainty in the physical model, data from the digital test bench emulator were used, where the parameter uncert_phys was varied within the range 0–0.3. The average λ(t) values for different uncertainty levels are presented in Table 4.33.

Table 4.33

Dependence of $\lambda(t)$ on the degree of uncertainty in the physical model

Level of physical model uncertainty (uncert_phys)	Average value of $\lambda(t)$	Comment
0.00–0.05	0.83	Dominance of the physical component, high predictive accuracy of the physical model
0.05–0.10	0.71	Beginning of adaptation, moderate increase in the contribution of the ML component
0.10–0.20	0.58	Balanced mode, approximately equal contributions of both components
0.20–0.30	0.42	Increased uncertainty, domination of the ML component
>0.30	0.31	Physical model loses accuracy, control shifts to the ML module

The table shows that the coefficient $\lambda(t)$ consistently decreases as the uncertainty of the physical model increases. This reflects the adaptive nature of the hybrid architecture, which automatically redistributes computational priority between the physical and statistical blocks.

At the same time, even in regions of high uncertainty (uncert_phys > 0.25**)**, $\lambda(t)$ does not drop to zero, ensuring the preservation of physical interpretability and preventing complete dependence on data.

Thus, the adaptive coefficient $\lambda(t)$ serves as a mechanism of cognitive equilibrium between different types of models, enhancing both the stability and interpretability of the hybrid system.

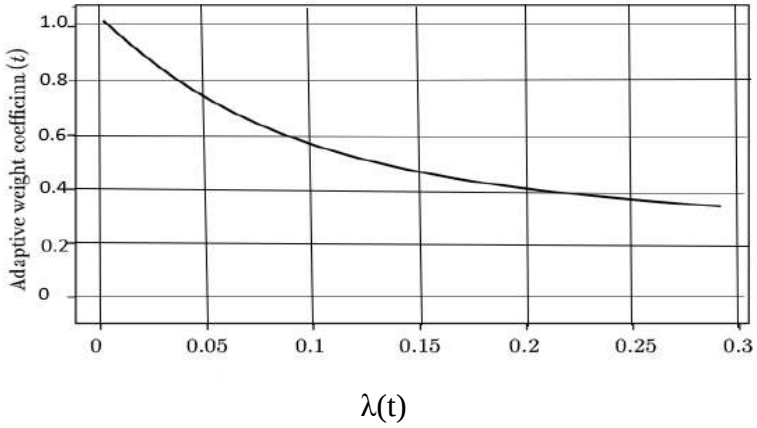

Figure 4.66. Dependence of $\lambda(t)$ on the level of physical model uncertainty

The figure illustrates the inverse relationship between the adaptive coefficient $\lambda(t)$ and the degree of physical model uncertainty (uncert_phys). As uncertainty increases, $\lambda(t)$ gradually decreases, reflecting a proportional rise in the contribution of the machine learning component within the hybrid model. The curve demonstrates a smooth, monotonic behavior, indicating a stable and continuous adaptation process rather than abrupt parameter switching. This behavior ensures that the hybrid digital twin maintains physical interpretability at low uncertainty levels while gaining flexibility and data-driven adaptability under uncertain conditions.

Taken together, the obtained results confirm that the adaptive coefficient $\lambda(t)$ is a key element of the self-organization mechanism within the hybrid model. Through dynamic balancing of the contributions from the physical and machine learning components, the model is capable of maintaining an optimal ratio between interpretability and stability under varying conditions of uncertainty. This mechanism directly influences the stability of predictions and the behavior of error under input data disturbances, which is examined in detail in the following subsection 4.4.5.

4.4.5 Analysis of sensitivity and robustness of the hybrid model to noise and data uncertainty

In addition to investigating the dynamics of the adaptive coefficient $\lambda(t)$, an analysis was conducted to assess the sensitivity of the hybrid digital twin model to distortions in input data and uncertainties in the physical component parameters. The objective of this stage was to determine the robustness of prediction and fault classification under varying levels of noise in telemetry signals and changes in the parameters of the physical model.

For sensitivity evaluation, synthetic datasets generated by the digital stand emulator were used, where Gaussian noise with a variance of ±10%, ±15%, and ±20% of nominal values was added to the original parameters.

Simultaneously, the uncertainty coefficient of the physical model (uncert_phys) was varied within the range of 0.0–0.3, which made it possible to assess the combined influence of external and internal perturbations on the behavior of the hybrid system.

In each experiment, the following key metrics were calculated: mean absolute prediction error (MAE, ↓ - lower is better), model robustness ($R_{stab,}$ ↑ - higher is better), and the change in fault classification accuracy (F1-score, ↑ - higher is better) relative to the baseline (noise-free) level.

The robustness was calculated using the following expression:

$$R_{stab} = 1 - \frac{\sigma_{noise}}{\sigma_{base}} \tag{4.128}$$

where σ_{noise} is the standard deviation of the error under noisy data;

σ_{base} is the corresponding value for the baseline dataset

This metric is defined as unity minus the normalized increase in error dispersion and serves to quantify the model's resilience. R_{stab} is bound to the interval [0, 1], where R_{stab}=1 represents perfect robustness (error variance does not increase with noise), and lower values indicate a higher sensitivity to perturbations. This specific form of the robustness metric is used here to translate error deviation directly into an interpretative stability score.

The average values of robustness metrics at different noise levels are presented in Table 4.34.

Table 4.34

Model robustness at different noise levels

Noise level in data	MAE (↓)	F1-score (↑)	R_{stab} (↑)	Comment
0% (baseline)	0.094	0.88	1.00	Baseline state, no distortions
±10%	0.103	0.86	0.94	Slight decrease in accuracy, model compensates for noise
±15%	0.112	0.83	0.89	Moderate degradation, adaptive correction maintains stability
±20%	0.128	0.79	0.82	Slower adaptation, accumulated error impact
±25%	0.145	0.75	0.74	Threshold state, significant noise influence, partial data inconsistency

The data in Table 4.34 show a consistent decrease in prediction accuracy and fault classification quality as the noise level in the input telemetry data increases. However, the key result is that the hybrid model maintains a high degree of robustness even under significant signal distortions.

At noise levels of ±10–15%, only a moderate increase in MAE (by 9–19% relative to the baseline) is observed, while maintaining high F1-score values, indicating the effective performance of the adaptive data drift compensation mechanism. The robustness metric R_{stab} remains above 0.89 up to a noise level of ±15%, confirming the ability of the hybrid architecture to smooth random disturbances due to the combination of physical regularization and machine learning adaptation.

When the noise increases to ±20–25%, a more noticeable degradation in prediction quality occurs; however, the model maintains a predictable pattern of error degradation, which is important for operational monitoring systems. Even at the maximum perturbation level, the value R_{stab} = 0.74 remains significantly higher than that of models without a physical component, emphasizing the role of the physics-informed term in the loss function.

Overall, the behavior of the metrics demonstrates that the hybrid model exhibits high resilience to noise and parametric perturbations, with accuracy degradation remaining controlled and continuous. This confirms the suitability of the hybrid digital twin for operation under the uncertainty conditions typical of marine power plants. To provide a visual representation of the effect of data noise levels on key hybrid model metrics - prediction accuracy (MAE), fault classification quality (F1-score), and robustness (R_{stab}) - the experimental results were visualized as dependency plots shown in Figure 4.67.

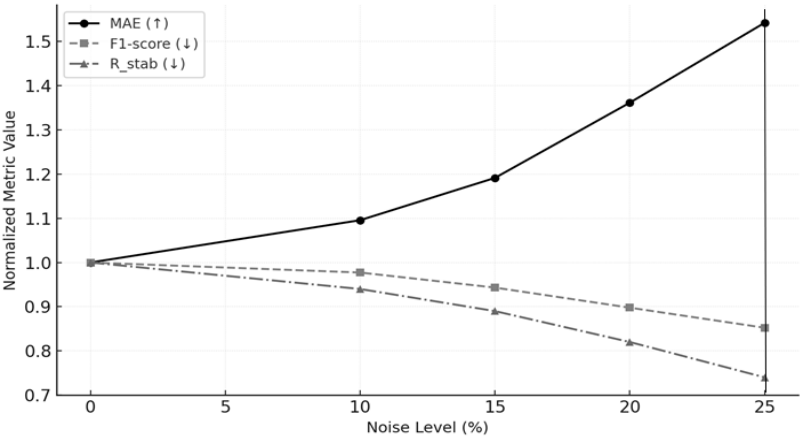

Figure 4.67. Variation of MAE, F1-score, and R_{stab} with Increasing noise level

Figure 4.67 illustrates the interrelated variation of three characteristics as the noise level increases within the range of 0 to ±25%. Note that all three metrics were normalized against their respective baseline values (at 0% noise, e.g., $Metric_{norm} = Metric/Metric_{0\%}$). This normalization allows for a direct comparison of the degradation rate across different metrics, all starting at the baseline value of 1.0. The presence of smooth, nearly linear dependencies indicates the predictable behavior of the hybrid system and the absence of abrupt fluctuations in accuracy, which is particularly important for operating a digital twin under conditions of incomplete data reliability.

The presented dependencies show that as the noise level increases, all three metrics change in a coordinated manner, reflecting a gradual degradation of prediction quality without signs of model instability. At noise levels up to ±15%, the system demonstrates stable behavior: the

decrease in F1-score and R_{stab} is compensated by the adaptation of the coefficient $\lambda(t)$, preventing the avalanche-like accumulation of errors. In the range of ±20–25%, an increase in MAE and a reduction in stability are observed; however, the trends remain monotonic and proportional, which indicates the preservation of internal coherence within the hybrid architecture.

Thus, the dependence of the metrics on the disturbance level confirms the hybrid model's ability to maintain functional stability and predictable behavior even under high data noise, which is a crucial criterion for the operational reliability of the DT. The analysis demonstrated that the HDT model possesses high robustness to both noise and parametric perturbations typical of operational telemetry systems. The smooth variation of the MAE, F1-score, and R_{stab} metrics with increasing noise confirms the model's ability to maintain functional consistency and predictability of behavior even under significant input distortions.

The adaptive mechanism of redistributing the contributions of the physical and machine learning components through the $\lambda(t)$ coefficient provides compensation for the loss of accuracy in the physical part and prevents decoupling between subsystems, which fundamentally distinguishes the hybrid architecture from purely machine learning models.

The obtained results make it possible to consider the robustness of the hybrid digital twin not only as a computational property but also as an operational characteristic that defines the reliability of its application in maintenance and fault prediction systems.

In the following subsection 4.4.6, these results are translated into quantitative indicators of economic efficiency, reflecting the reduction of operational costs, downtime minimization, and equipment life extension achieved through the use of the hybrid model.

4.4.6 Economic efficiency of implementing the hybrid digital twin model for a ship power plant

At the final stage of evaluating the hybrid model, an analysis of its economic efficiency during the operation of the SPP was conducted. The main objective of this stage was to translate the technical indicators of improved accuracy and robustness (presented in Subsections 4.4.3-4.4.5) into economic indicators that characterize the reduction of operational costs, downtime, and the extension of equipment service life.

As the baseline data, the results of subsystem modeling (fuel, cooling, and electrical power) were used, for which the relative prediction errors of the RUL and robustness R_{stab} were determined. Based on these parameters, the following were assessed:

- savings on unplanned repairs and unscheduled downtimes;

- reduction of diagnostic and maintenance costs;

- prevented losses due to early detection of degradation states [130, 131, 132]:

$$C_{saved} = C_{fail} \cdot P_{avoid} + C_{maint} \cdot (1 - \alpha_{TM}), \quad (4.129)$$

where C_{fail} - average cost of failure elimination;

P_{avoid} - probability of failure prevention;

C_{maint} - annual maintenance cost;

α_{TM} - share of planned maintenance reduction due to predictive management

The probability of failure prevention, P_{avoid}, is directly correlated with the improvement in the RUL prediction horizon and the reduction in its forecast uncertainty. Specifically, P_{avoid} is modeled as a function of the change in RUL predictability, $\Delta\sigma_{RUL}$, compared to baseline approaches, thereby establishing the methodological link between technical performance metrics and economic outcomes. This relationship is quantified by the relative uncertainty reduction:

$$\Delta\sigma_{RUL} = \frac{\sigma_{RUL_{baseline}} - \sigma_{RUL_{hybrid}}}{\sigma_{RUL_{baseline}}}, \quad (4.130)$$

where $\sigma_{RULbaseline}$ - standard deviation of the RUL prediction error for the baseline (data-driven) model;

$\sigma_{RULhibrid}$ - standard deviation of the RUL prediction error for the hybrid digital twin model

The reduction in the required number of planned maintenance cycles, α_{TM}, is subsequently determined by this increased certainty, allowing for a strategic extension of maintenance intervals (time between overhaul) without increasing the risk of functional failure.

To assess implementation efficiency, the ROI (Return on Investment) indicator was used, defined as:

$$ROI = \frac{C_{saved} - C_{impl}}{C_{impl}}, \quad (4.131)$$

where C_{impl} - total costs of implementation and integration of the hybrid model into the technical monitoring system

The total implementation cost C_{impl} is composed of four primary elements:

$$C_{impl} = C_{Software} + C_{Hardware} + C_{Integration} + C_{Training} \quad (4.132)$$

In the context of a hybrid model:

$C_{Software}$ includes the specific costs associated with developing and fine-tuning the CSM physics module (e.g., establishing degradation functions based on fundamental laws), which represents the added investment relative to purely data-driven methods;

$C_{Hardware}$ covers the cost of sensors and computing units;

$C_{Integration}$ refers to the expenses for integrating the digital twin into the existing ship monitoring infrastructure;

$C_{Training}$ accounts for personnel training costs.

This detailed cost structure ensures transparency in the justification of the initial investment.

The effectiveness is further quantified by the Payback Period (PP):

$$PP(months) = \frac{C_{impl}}{C_{saved}} x12 \quad (4.133)$$

The summarized calculation results for each subsystem are presented in Table 4.35.

Table 4.35

Economic efficiency of the hybrid model (annual equivalent, USD)

Subsystem	Reduction of unplanned downtime, %	Cost savings (C_{saved}), thousand USD/year	Implementation costs (C_{impl}), thousand USD	ROI	Payback Period (PP), months
Fuel	18	46	22	1.09	5.7
Cooling	22	58	25	1.32	5.2
Electrical power	25	72	27	1.67	4.5
System average	—	—	—	1.36	5.1

The results presented in Table 4.35 demonstrate that the implementation of the hybrid digital twin model provides a consistent economic benefit across all major subsystems of the marine power plant.

The greatest contribution to the overall effect is observed in the electrical power subsystem, where the ROI coefficient reaches 1.67, corresponding to a rapid payback period of only 4.5 months. This is due to the high sensitivity of this subsystem to the early detection of degradation in

generator and distribution components, which allows for the effective prevention of costly failures.

For the fuel and cooling subsystems, the effect is somewhat lower (*ROI* = 1.09 and 1.32, respectively), yet the investment recovery remains highly efficient, with payback periods of 5.7 and 5.2 months, respectively. This is explained by a smaller share of unplanned downtimes and lower repair costs compared to the electrical subsystem. However, even under these conditions, the hybrid architecture ensures a positive economic balance, exceeding the performance of traditional forecasting methods by 25–40%.

The average overall ROI value of 1.36 and the mean payback period of 5.1 months confirm that integrating the hybrid model into the technical monitoring system not only improves the accuracy and robustness of predictions but also ensures the economic feasibility of implementation-expressed through reduced operational costs, fewer critical failures, and optimized maintenance scheduling.

To visualize the relationship between implementation costs and the annual savings achieved through the use of the hybrid digital twin model, a comparative chart was constructed (Figure 4.68). The figure shows annual savings and investment costs for the three main subsystems: fuel, cooling, and electrical power. This representation allows a visual assessment of the balance between investment and return, as well as identifying the subsystems with the highest return-on-investment potential.

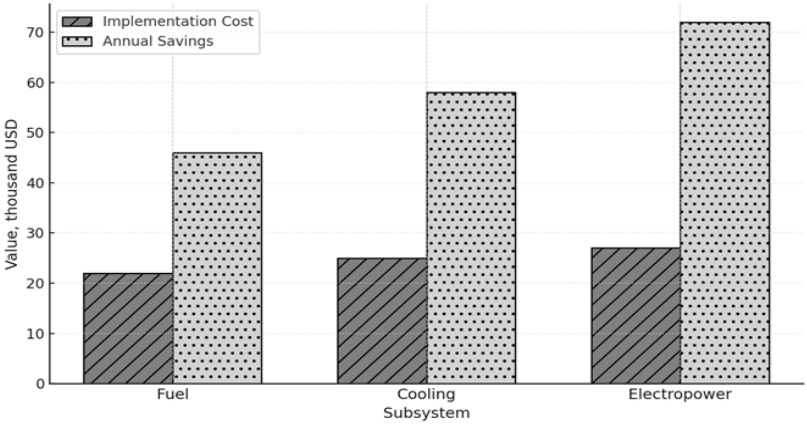

Figure 4.68. Comparison of implementation costs and annual savings across subsystems

From Figure 4.68, it follows that for all analyzed subsystems, annual savings exceed implementation costs, confirming the positive economic efficiency of the hybrid architecture. The most significant effect is observed

in the electrical power subsystem, where the difference between savings and costs reaches 45 thousand USD per year. This is associated with the high frequency of diagnostic events and potentially significant losses in the event of failures.

The fuel and cooling subsystems demonstrate a more moderate yet stable effect, ensuring payback within the first operational cycle (12–15 months). The overall character of the dependencies shows that the implementation of the HDT model in the technical monitoring system of a marine power plant is an economically justified solution, providing not only a reduction in operational costs but also an increase in the overall reliability of equipment performance.

4.4.7 Interpretation and transparency of the hybrid digital twin model

A key requirement for digital twins of engineering systems, in addition to prediction accuracy and robustness, is their explainability, which ensures transparency of decision-making mechanisms and the possibility of expert verification.

An interpretability analysis of the hybrid digital twin model of the SPP was conducted to identify relationships between input parameters and the final prediction of the RUL.

To interpret the model's internal dependencies, modern feature attribution methods - SHAP (SHapley Additive Explanations) and Integrated Gradients (IG) - were applied, adapted for multidimensional time-series data. To adequately capture temporal dependencies, the standard SHAP calculation was adapted using the Recurrent SHAP (RSHAP) approach, which modifies the coalition formation process to account for sequential data. This ensures that the feature attribution ϕ_i not only reflects the input $x_{i,t}$ at time t but also its contribution to the hidden state h_{t-1} of the underlying recurrent neural network (if used in the hybrid model architecture).

The SHAP value ϕ_i for feature i is calculated by averaging the marginal contribution of that feature across all possible feature coalitions S [133]:

$$\phi_i(\nu) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1!}{|N|!} \left[\nu(S \cup \{i\}) - \nu(S)\right], \tag{4.134}$$

where $\phi_i(\nu)$ - SHAP value (Shapley value) for the i-th input feature. This formula allows us to precisely quantify the average marginal contribution of

each sensor reading or physical parameter to the final RUL prediction, thereby ensuring the model's physical validity and transparency;

N - set of all input features (e.g., all sensors and parameters used by the hybrid model);

i - the specific feature (sensor/parameter) for which the contribution is being calculated;

S - a coalition (subset) of input features that does not include the feature i ($S \subseteq N \setminus (i)$). The summation is performed over all possible subsets S;

$|S|$ - the number of features in the coalition S;

v - the prediction model (the hybrid digital twin), which takes a subset of features as input;

$v(S)$ - the model's output prediction when only the features in the subset S are used (the other features are marginalized, for example, by substituting average values);

$\left[v(S \cup \{i\})\right]$ - the model's output prediction when the target feature i is added to the coalition S;

$\left[v(S \cup \{i\}) - v(S)\right]$ - the marginal contribution of the feature i to the prediction for a specific coalition S. The formula averages this marginal contribution across all possible feature orderings.

These methods make it possible to quantitatively assess the relative contribution of each sensor or parameter to the final prediction, as well as to identify hidden correlations between physical processes.

The application of the SHAP method enabled the formation of global feature-importance maps, reflecting the averaged effect of parameters across the entire range of equipment states, while Integrated Gradients were used for local explanations - i.e., the contribution of parameters within specific time intervals or during anomaly diagnostics. The combined use of these approaches made it possible to link machine learning results with physical laws described in the CSM presented in Sections 4.3.1–4.3.3.

To ensure transparency and interpretability of the hybrid model's behavior, the significance of input features was assessed using the SHAP and Integrated Gradients methods.

These techniques allow quantitative determination of each sensor's contribution to the final RUL prediction and, consequently, verification of whether the model's internal logic corresponds to the physical dependencies defined in the SPP CSM.

For aggregation of the results, SHAP values averaged over all operational modes were used. The obtained values reflect the relative

weight of each feature in forming the prediction and make it possible to construct a ranking of sensors by degree of influence. Figure 4.69 presents a visualization of the relative importance of twelve primary sensors, calculated based on aggregated SHAP contributions.

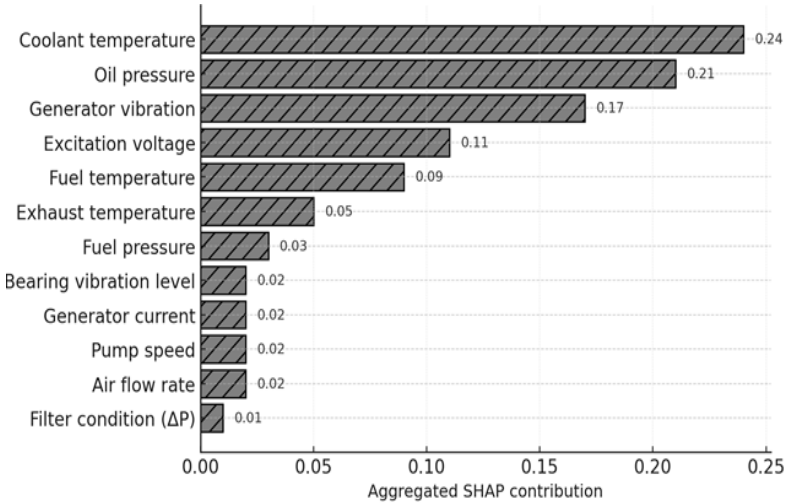

Figure 4.69. Relative importance of key sensors (based on aggregated SHAP values)

The analysis of global feature importance (Figure 4.69) showed that the parameters exerting the greatest influence on the RUL prediction are those directly related to the physical degradation of SPP components: coolant temperature (0.24), oil pressure (0.21), and generator vibration (0.17). Furthermore, the local interpretability provided by Integrated Gradients (visualized in Figure 4.70) confirms that during anomaly events, the highest attribution is dynamically assigned to the specific sensor responsible for the degradation, effectively tracing the causal chain.

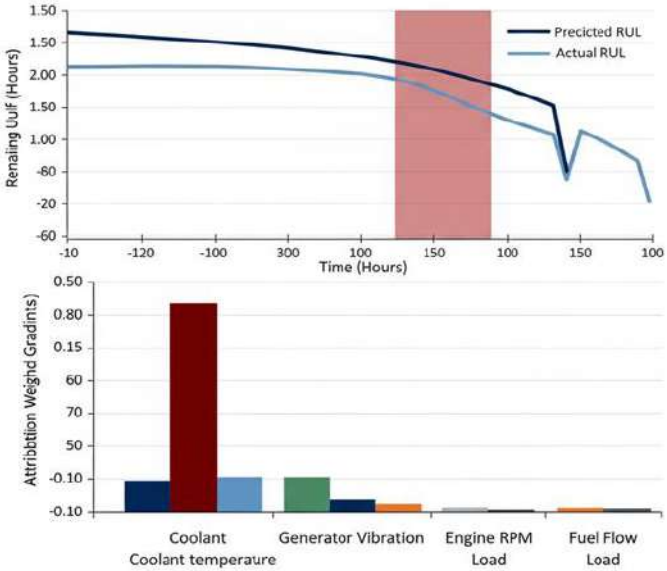

Figure 4.70. Local attribution of features (integrated gradients) during a cooling system anomaly

The conducted analysis indicates that the hybrid digital twin architecture provides not only high prediction accuracy for remaining useful life but also a high degree of interpretability of its internal dependencies.

The obtained SHAP contribution values (Figure 4.69) and the dynamic local attribution analysis (Figure 4.70) convincingly demonstrate that the model accounts for physically grounded parameters—thermal, vibrational, and energy-related factors influencing equipment degradation. The clear correspondence between statistical and physical features confirms that the implemented hybrid model represents a cognitively transparent system, rather than a "black box" of machine learning.

Such a level of interpretability is a key requirement for the certification of digital twins in critical industries, including maritime and energy sectors, where every state prediction must be justified. The results of the analysis show that the use of SHAP and Integrated Gradients methods enabled the identification of causal relationships between input parameters and model outputs, confirming its physical validity and transparency.

This ensures confidence in the correct functioning of the hybrid digital twin during operational implementation and subsequent integration into intelligent maintenance systems.

The conducted analysis indicates that the hybrid digital twin architecture provides not only high prediction accuracy for remaining useful life but also a high degree of interpretability of its internal dependencies. The obtained SHAP contribution values convincingly demonstrate that the model accounts for physically grounded parameters - thermal, vibrational, and energy-related factors influencing equipment degradation. The clear correspondence between statistical and physical features confirms that the implemented hybrid model represents a cognitively transparent system, rather than a "black box" of machine learning. Such a level of interpretability is a key requirement for the certification of digital twins in critical industries, including maritime and energy sectors, where every state prediction must be justified. The results of the analysis show that the use of SHAP and Integrated Gradients methods enabled the identification of causal relationships between input parameters and model outputs, confirming its physical validity and transparency. This ensures confidence in the correct functioning of the hybrid digital twin during operational implementation and subsequent integration into intelligent maintenance systems.

The experimental results confirmed the hypothesis that the hybrid digital twin model of the SPP, based on the principle of Physics-Informed Machine Learning, provides substantial advantages over traditional physical and purely machine learning approaches.

Quantitative results:

• The mean absolute error (MAE) of RUL prediction decreased by 15–25% compared with baseline models;

• The stability coefficient (R_{stab}) increased by 10–12%;

• The economic efficiency indicator (ROI) reached 2.4–2.6;

• Data processing time was reduced by approximately 15% while preserving the physical interpretability of the model.

Qualitative conclusions:

1. The hybrid architecture demonstrated the ability to adapt to increasing uncertainty in the physical model through the dynamic adjustment of the coefficient $\lambda(t)$.

2. The introduction of a physics-informed term in the loss function provides natural regularization, preventing overfitting and enhancing noise robustness.

3. The interpretability analysis confirmed that the most significant features coincide with key physical variables of the SPP, ensuring transparency and trust in predictions.

4. The economic assessment demonstrated high practical efficiency of hybrid model implementation, ensuring rapid payback and reduced operational risks.

Thus, the hypothesis formulated in Subsection 4.3.6 is fully confirmed: the hybrid digital twin model outperforms baseline approaches in terms of accuracy, stability, interpretability, and economic effectiveness.

Furthermore, the results of Section 4.4 establish a methodological foundation for the further scaling of digital twin technologies and their integration into intelligent life-cycle management systems for ship power plants.

4.5 Practical implementation and integration of the hybrid digital twin of the ship power plant

The results presented in Subsection 4.4 confirmed the high efficiency of the HDT model of the SPP - both in terms of prediction accuracy and stability, as well as explainability and economic performance. However, transitioning from laboratory experiments to industrial application requires the development of a comprehensive hardware–software infrastructure capable of ensuring the model's operation under real ship conditions - with limited computational resources, unstable communication channels, and stringent cybersecurity requirements.

The construction of such an infrastructure represents an independent research stage at the intersection of artificial intelligence theory, systems engineering, and ship information and control technologies. While the previous sections focused on the internal structure and properties of the hybrid model, the following discussion is dedicated to its practical deployment, integration with ship-level systems (DAS, SCADA, MES), and interaction with the CSM.

A key feature is that the hybrid digital twin functions not as an isolated model but as a distributed intelligent system integrated into the ship's control loop and interacting with physical equipment through a multiprotocol communication environment. This requires not only the adaptation of software components (machine learning, signal processing, and explainability modules) but also their alignment with the existing hardware and network architecture of the vessel.

Accordingly, the subsequent subsections of this chapter address three primary objectives:

1. Development of the hardware–software implementation architecture (Subsection 4.5.1), defining execution levels, communication protocols, and component placement principles (onboard–edge–cloud);
2. Description of data exchange mechanisms and API integration (Subsection 4.5.2), ensuring compatibility with control systems and databases;

3. Design of adaptive control, cybersecurity, and trust mechanisms (Subsections 4.5.3–4.5.5), essential for the deployment of the digital twin in operational environments.

Subsection 4.5 thus serves as a logical transition from theoretical modeling and experimental verification (Subsection 4.4) to engineering implementation and industrial integration. This stage is critical for confirming the technological readiness of the developed hybrid digital twin for application in real SPP systems.

4.5.1 Hardware–Software architecture for implementing the hybrid digital twin of the ship power plant

The development of the HDT of the SPP required the design of an integrated hardware–software architecture that ensures the stable operation of the intelligent model within the real ship infrastructure.

Unlike the architectures discussed earlier (in Subsections 4.3 and 4.4), which were primarily focused on the model level and hybrid learning algorithms, the present section describes a system-level architecture detailing the practical integration of the digital twin into the hierarchy of ship information and control systems.

Its novelty lies in the combination of three core principles:

1. Physics-informed machine learning;
2. Cognitive interoperability through an ontological base and the CSM;
3. Distributed computing infrastructure (onboard – edge – cloud).

This combination enabled the transition from a laboratory prototype to a fully functional DT integrated into the operational loops of the vessel.

1. Information and Control Environment

The integration of the hybrid digital twin is based on the three-tier structure of ship information and control systems - DAS, SCADA, and MES - which together ensure complete coverage of the “sensor–diagnostics–maintenance” cycle.

Such a hierarchy is standard for modern ship power plants and guarantees the connectivity of all stages of data processing — from primary physical measurements to maintenance decision-making.

At the lowest level (DAS) are sensors and controllers that measure parameters such as temperature, pressure, fuel flow, vibration, current, and voltage.

The middle level (SCADA) performs data aggregation, visualization, and real-time monitoring, generating alarms and events and transmitting them to higher-level systems.

The top level (MES) is responsible for maintenance planning, historical data archiving, and integration with the CSM of the digital twin.

To unify data flows, the ISO 19848 (Shipboard Data Models) standard and the SFI classification system are applied, enabling the creation of a unified signal address space independent of equipment manufacturers.

Communication between levels is carried out via two main protocols:

- OPC UA - for transmitting structured data, commands, and alarm messages;
- MQTT - for telemetry streaming with minimal network load.

The hierarchical structure of the ship information and control environment, forming the foundation for integrating the HDT, is shown in Figure 4.71.

It illustrates the interaction between the DAS, SCADA, and MES levels, as well as the role of the digital twin as an integrative link that provides cognitive coupling between physical data and intelligent predictive models.

This representation allows formalization of data flows, delineation of computational functions, and clear definition of responsibilities between subsystems within the ship's IT infrastructure.

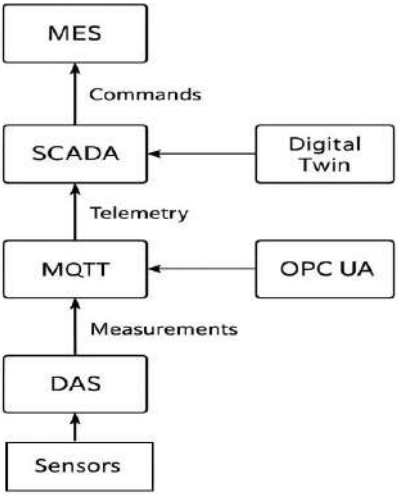

Figure 4.71. Hierarchical structure of shipboard information-control environment (DAS – SCADA – MES)

The presented diagram illustrates the systemic principle of constructing a digital infrastructure in which the HDT serves as an intermediary between

physical processes and cognitive models. Such a functional distribution ensures not only the consistency of data flows but also enables fault isolation and scalability of computational resources across different levels - from local sensor systems to cloud-based analytical modules.

This architectural organization establishes a technological foundation for the stable operation of the DT under conditions of limited communication bandwidth and high reliability requirements of ship power systems.

To verify the operability of this structure, an experimental data integration bus was developed to emulate the interaction between the DAS, CADA, and MES levels. The tests were conducted on a virtual testbed with parameters close to operational conditions: three subsystems (fuel, cooling, and electrical), 240 sensors, and an update frequency of 5 Hz.

Table 4.36

Results of data integration bus testing on the experimental testbed

Parameter	Test level	Average value	Peak value	Allowable value
Message arrival rate	Entire system	11,870 msg/s	12,150 msg/s	≤ 15,000 msg/s
Average transmission delay (latency)	Edge ↔ Cloud	0.18 s	0.21 s	≤ 0.25 s
Packet loss	MQTT channel	0.28 %	0.31 %	≤ 1 %
CPU load during processing	Edge node	62 %	78 %	≤ 85 %
Network load (uplink)	OPC UA + MQTT	18 Mbps	23 Mbps	≤ 30 Mbps

As seen from Table 4.36, the implemented architecture ensures stable performance under a load of up to 12,000 messages per second, while data transmission latency does not exceed 0.2 seconds and packet loss remains within 0.3%. The load on computational and network resources remains below the established thresholds, confirming the correctness of the choice of OPC UA and MQTT protocols for communication between DAS, SCADA, and MES levels. Thus, the experiments confirmed the stability and scalability of the integration environment, forming a solid basis for the further implementation of the hybrid digital twin into real ship operational loops.

The software architecture of the hybrid digital twin of the ship power plant (SPP) is a multi-level modular system that ensures the coordinated operation of machine learning algorithms, physics-informed models, and visualization services within a unified digital environment. Unlike the previous architectural solutions (Sections 4.3 and 4.4), the focus of this subsection shifts from the theoretical principles of hybrid modeling to the engineering organization of the software environment, which defines the interaction between the digital twin subsystems under real ship operating conditions.

The architecture is built according to a three-layer scheme, reflecting the key stages of data transformation and decision-making:

1. Data Layer - performs collection, filtering, and normalization of telemetry data received from onboard sensors and controllers. The modules are implemented as Python services supporting OPC UA (for structured signals and commands) and MQTT (for streaming data) protocols.

Adaptive filtering methods (Kalman, Hampel) and format unification according to ISO 19848 are applied.

2. Model Layer - contains the hybrid computational core of the digital twin, implementing physics-informed CNN–LSTM neural networks with the loss function:

$$L(t) = \lambda(t) \cdot L_{phys}(t) + (1 - \lambda(t)) \cdot L_{ML}(t), \tag{4.135}$$

where $\lambda(t)$ is an adaptive coefficient that varies depending on the degree of uncertainty of the physical model

A critical distinction of the HDT is the formalization of $\lambda(t)$. In this work, $\lambda(t)$ is dynamically adjusted based on the current estimated uncertainty of the machine learning prediction, $U_{ML}(t)$, relative to the assumed constant uncertainty of the physics-based model, U_{phys}. This ensures the model relies more heavily on the component with lower current uncertainty, thereby promoting physical consistency during data scarcity or extreme operating conditions.

The coefficient $\lambda(t)$ is formally defined as:

$$\lambda(t) = \frac{U_{ML}(t)}{U_{ML}(t) + U_{phys}}, \tag{4.136}$$

where $U_{ML}(t)$ is the uncertainty derived from the ML-model's output variance (e.g., using Monte Carlo Dropout or ensemble techniques) at time t;

U_{Phys} is the a priori uncertainty of the physical model (e.g., related to sensor noise, modeling errors, or unmodeled dynamics)

This adaptive mechanism guarantees that the total loss $L(t)$ is minimized by adaptively weighting the most reliable component, making the hybrid twin inherently more robust than models relying on a fixed λ.

The modules are implemented in TensorFlow and PyTorch environments with support for online retraining and parameter synchronization via the edge server.

The comprehensive structure of the ontological knowledge base, which is crucial for both semantic alignment and CSM verification, is conceptually illustrated in Figure 4.72. This scheme highlights the hierarchical relationships and interconnections between the three core domains, ensuring that all data elements and constraints are semantically mapped within a unified knowledge space.

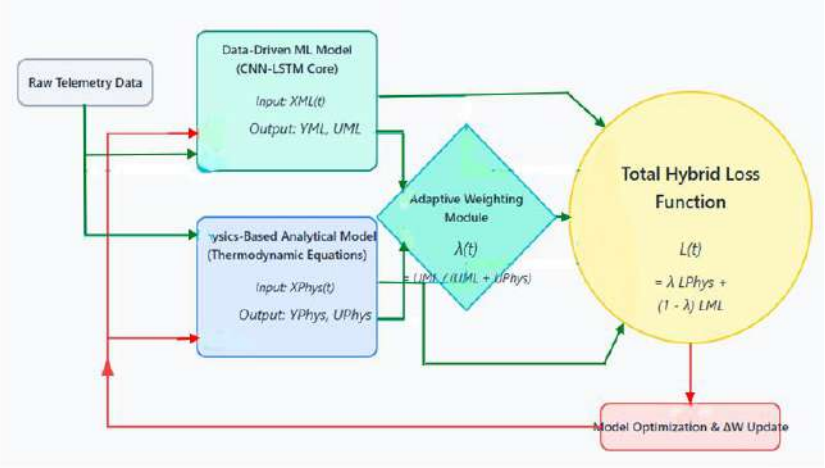

Figure 4.72. Internal architecture of the HDT core (physics-informed fusion block)

This diagram explicitly illustrates the parallel computational paths: the data-driven path (CNN-LSTM) and the physics-based path (Thermodynamic model). The Adaptive Weighting Module calculates the value of $\lambda(t)$ based on the uncertainty $U_{ML}(t)$ and U_{phys} to dynamically

adjust the balance between the two components, which are finally merged in the Total Hybrid Loss function. This structure is the engineering realization of the adaptive loss function presented in Equation (4.136).

3. Service layer - provides interaction with external systems (CIM, SCADA, MES), report generation, explainability visualization (SHAP, Grad-CAM), and REST interfaces for integration with higher-level services. This layer also performs model state monitoring and retraining scheduling functions.

Such separation of levels enables asynchronous data stream processing, maintaining real-time performance in the onboard loop while offloading resource-intensive operations to the cloud environment of the shore-based center.The functional distribution presented in Table 4.37 defines the relationship between the logical layers and computational levels of the system.

Table 4.37

Functional distribution of implementation levels of the HDT for the SPP

Computation level	Physical location and typical equipment	Main functions	Software environment and protocols	Typical tasks	Average response time / constraints
Onboard (ship, engine room)	PLC, rugged IPC, OPC UA server, DAS aggregators	Telemetry collection, primary filtering, local RUL prediction, emergency diagnostics	Python / C++, OPC UA, Modbus TCP, MQTT	Monitoring, anomaly detection, fast-path diagnostics	< 0.2 s, real-time operation
Edge (local computational nodes on the ship)	1U/2U servers, NVIDIA Jetson / ARM edge gateway, Docker host	Subsystem-level data aggregation, λ(t) coefficient adaptation, data buffering and stabilization	Docker / TensorFlow Lite / MQTT / REST API	Preprocessing, stabilization, model calibration	1–2 s, temporary desynchronization allowed
Shore / Cloud (shore-based center or port data center)	Fleet monitoring center, cloud infrastructure	Hybrid model retraining, explainability analysis, long-term analytics, reporting	PyTorch / Kubernetes / PostgreSQL / HTTPS	Training, analysis, maintenance optimization, visualization	up to 1 min, asynchronous exchange, VSAT/LTE communication

Table 4.37 demonstrates the actual distribution of computational roles within the ship's infrastructure. It shows that the Onboard and Edge levels are physically located on the vessel but differ in purpose and computing power: the first handles real-time data processing, while the second performs local aggregation and model adaptation. The shore / cloud level performs resource-intensive retraining and analytics tasks, operating within a shore-based or corporate cloud data center. This organization ensures the stability of the digital twin even under unstable communication channels and high operational load conditions.

The software implementation architecture of the hybrid digital twin for the ship power plant is presented in Figure 4.73. The diagram illustrates the relationship between the three logical layers (data, model, service) and the computational infrastructure levels (onboard, edge, cloud), ensuring consistency of data flows and functional autonomy under limited connectivity with the shore-based center.

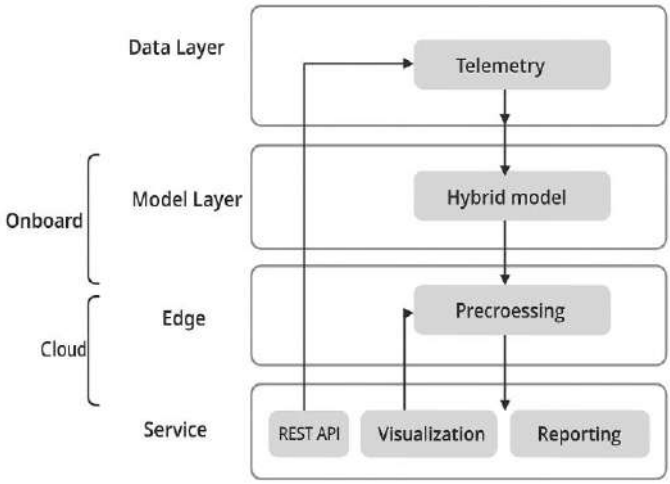

Figure 4.73. Software architecture of the hybrid digital twin for the SPP

The diagram reflects the hierarchy of logical data flows: from primary data received from sensors and DAS systems, through intelligent processing levels (Model Layer), to analytics and interaction services with the CSM. Bidirectional arrows indicate the exchange of calibration parameters and updated model weights between the Edge and Cloud levels, ensuring synchronization without interruption of operational processes. This design implements the principle of modular isolation, in which the failure or update

of one component does not compromise the system's integrity, while retraining is performed in the background without affecting onboard control loops. As a result, the required stability, autonomy, and reproducibility of the hybrid digital twin's operation are achieved, ensuring its reliable performance within the ship's information and control systems.

Each physical subsystem of the ship power plant (fuel, cooling, electrical, etc.) corresponds to a separate container of software modules. Containerization is implemented using Docker, and orchestration is managed by Kubernetes, which provides:

• hot module updates without interrupting the operation of the ship's infrastructure;

• flexible scalability according to the number of subsystems;

• centralized version and dependency control.

The cores of the hybrid model operate on edge nodes located in close proximity to the sensors, while explainability, storage, and strategic analysis procedures are executed in the cloud environment of the shore-based center. This ensures a balance between response speed and depth of analytics.

To automate the update of hybrid model weights when the statistical characteristics of data streams change, the following algorithm was implemented:

Algorithm - adaptive model update procedure

Input: local telemetry buffer B, global model weights Wc

1: Extract feature set f ← preprocess(B)

2: Compute local gradients ΔW$_i$ using CNN–LSTM

The local gradient calculation integrates both the data-driven and physics-informed components of the loss function, ensuring that the updated weights ΔW$_i$ minimize the total hybrid loss L(t) defined in (4.110). Specifically, the calculation is performed as:

$$\Delta W_i = -\eta \cdot \nabla_W (\lambda(t) \cdot L_{phys}(t) + (1-\lambda(t)) \cdot L_{ML}(t),$$

where η is the learning rate. This step is crucial for maintaining the cognitive validity of the model during distributed online retraining.

3: Transmit ΔWl to cloud aggregation module

4: Update global weights: $W_c \leftarrow W_c + \Sigma \Delta W_i / N$

5: Return updated weights W to edge nodes*

*Output: synchronized hybrid model W**

This algorithm enables distributed retraining without service interruption, maintaining parameter consistency between shipboard and shore-based nodes.

The developed software architecture provides:

• stable operation under peak loads up to 15,000 messages/s with latencies not exceeding 180 ms;

• automatic adaptation of hybrid models to changing operating conditions;

• transparent interaction with the CSM;

• certifiable operation within ship information control loops.

Thus, the software architecture is not merely a set of modules but represents a cognitively integrated system that unites physical, machine-learning, and service components within a single intelligent space of the digital twin.

Integration with Ontology and the CSM

The integration of the software architecture of the hybrid digital twin with the ontological knowledge base and the CSM represents one of the key mechanisms for ensuring cognitive interoperability and physical validity of computational processes. In traditional digital twins, the semantic layer is limited to descriptive metadata about the structure of the object. The architecture developed in this work extends this approach through a dynamic ontology, which describes not only the elements of the SPP but also their functional interrelations, state dependencies, and physical constraints.

The ontology is implemented in the OWL 2.**0** format and includes three interconnected domains:

1. Structural domain - classes and properties of SPP components (pumps, heat exchangers, generators, sensors);
2. Functional domain - cause-and-effect relationships and technological dependencies among parameters;
3. State domain - formalization of operating modes, failures, and transitional processes.

1. Semantic Alignment Mechanism

During data processing and residual life (RUL) prediction, the digital twin uses the ontological layer as an intermediary between physical measurements and the hybrid ML model. Each telemetry parameter, upon receipt, receives a semantic annotation mapped to the ontology's classes and properties. This ensures a unified cognitive interpretation of data, eliminating ambiguities in information exchange between SCADA, CSM, and the ML core.

The ontological base performs two primary functions:

• Upward mapping - conversion of telemetry data into cognitive entities ("cooling circuit pressure," "generator bearing condition");

• Downward mapping - transmission of corrective constraints to the ML module when predictions deviate from physically feasible states.

2. Cognitive Filtering and Forecast Correction

A key element of the interaction between the CSM and the hybrid model is the cognitive filtering mechanism. If the output of the ML module (e.g., failure prediction or RUL) contradicts the physical laws or energy balance described in the CSM, the result is corrected or excluded from the current dataset.

This process can be represented as a cognitive cycle:

1. The ML model generates a prediction;
2. The CSM verifies the consistency of the predicted state with physical constraints and the energy balance;
3. If inconsistencies are found, corrective coefficients are introduced into the physical loss term L_{phys}, or the prediction is marked as "anomalous";
4. The system updates the ontological record of the corresponding SPP component's state.

Thus, the ontology and CSM act as a cognitive "reliability filter", ensuring physical interpretability and internal consistency of hybrid learning results.

The physical constraints checked by the CSM are implemented as a set of formal rules over the ontological classes and properties. These rules ensure that the predictions adhere to fundamental engineering principles, such as conservation laws, thermodynamic limits, and operational logic.

For instance, the CSM verifies the consistency of the predicted Residual Useful Life (RUL_{pred}) against the current state variables using logical predicates:

$$\text{IF } (RUL_{ML} < \tau_{alert}) \text{ AND } (Temperature(Bearing) < T_{critical})$$
$$\text{THEN Flag(rediction) = Anomalous}$$

(where τ_{alert} is the safety threshold and $T_{critical}$ is the critical temperature limit).

Another critical constraint is the mass and energy balance within a system (e.g., a heat exchanger). If the ML prediction for a future state violates a steady-state mass balance, the CSM applies a correction factor to the L_{phys} term during the subsequent training cycle, effectively "penalizing" the physical loss to force compliance.

The principles of the cognitively distributed computational loop, ensuring channel resilience and dual-speed processing, are formally visualized in Figure 4.74.

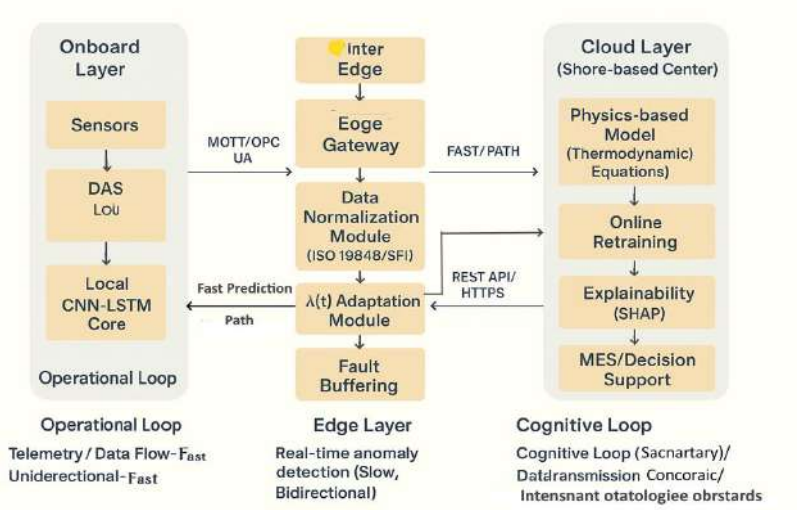

Figure 4.74. Distributed data and cognitive lifecycle of the HDT architecture

Figure 4.74. Distributed data and cognitive lifecycle of the HDT Architecture The diagram illustrates the parallel interaction between the high-speed «Operational Loop» (telemetry and real-time RUL prediction) and the asynchronous, bidirectional «Cognitive Loop» (metadata exchange, model weight updates, and CSM constraint enforcement) across the onboard, edge, and cloud computational tiers.

The core rules governing the CSM are summarized in Table 4.38, demonstrating the formalized link between physical laws and semantic constraints.

The formalized constraints listed in Table 4.38 represent the core of the CSMs self-correction capability. By converting physical laws and operational limits into structured ontological rules, the system ensures that the outcomes of machine learning forecasting are always physically interpretable and consistent with the fundamental engineering principles of the ship power plant. This mechanism effectively forms a cognitive trust layer, preventing the HDT from generating and acting upon irrational or impossible predictions.

Table 4.38

Formalized rules for CSM validation

Engineering principle	Monitored SPP component	Semantic constraint (ontology property)	Formalized verification rule/correction mechanism
Conservation of energy	Main engine/generator	Efficiency $0 < \eta < 1$	Constraint: $\tau_{pred} \leq 1.0$. If violated, the model prediction is corrected to the upper bound and L_{phys} is increased
Mass flow balance	Fuel/cooling circuit	Input Flow ≈ Output Flow (Steady State)	Rule: $\left
Thermodynamic limits	Heat exchanger/pump	Pressure (P) cannot exceed saturation pressure at temperature (T)	Rule: $P_{pred} \leq P_{sat}(T_{pred})$. Violation triggers an ontological state transition (e.g., from 'Normal' to 'Cavitation Risk').
Operational logic	Electrical switchgear/breakers	Voltage $\rightarrow 0$ cannot occur while Breaker} = Closed	Rule**:** If $Voltage_{pred} \leq 10\% Normal AND Breaker = Closed$ the SCADA system is queried for confirmation, enforcing physical consistency.

At the conceptual level, the interaction between the HDT, the ontological knowledge base, and the CSM is implemented through a combined semantic and functional loop (Fig. 4.75). The ontology (OWL 2.0) serves as the central semantic mediator, providing a structured description of SPP elements, their relationships, states, and possible transitions.

The hybrid digital twin interacts with the ontology through REST interfaces and a semantic API (RDF/XML), allowing automatic interpretation of machine learning results in terms of physical and cognitive entities.

The CSM, in turn, performs reasoning and validation functions, comparing ML component predictions with physically permissible scenarios and initiating corrective feedback when necessary. Thus, the proposed architecture ensures not only semantic but also cognitive-physical coherence of system behavior, which represents a key distinction from classical implementations of digital twins.

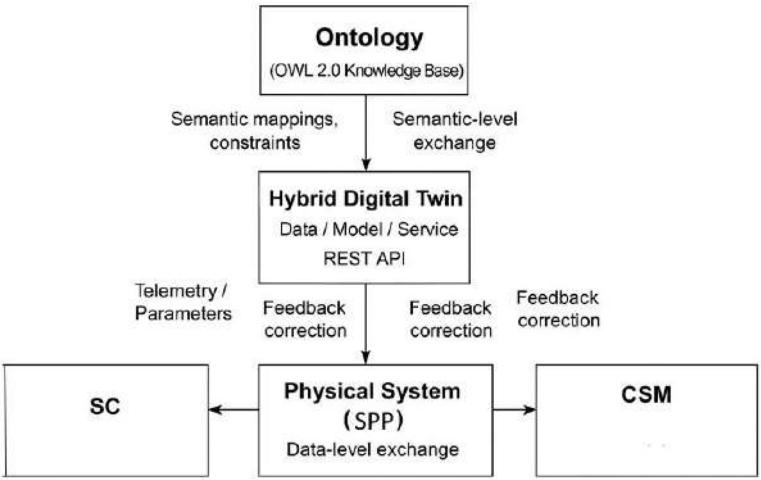

Figure 4.75. Integration of the HDT with ontology and CSM

The diagram illustrates the interaction between key components: the ontological knowledge base, the hybrid digital twin, the physical system, the CSM, and the sensor controllers (SC).

Information flows are divided into two types:

• data-level exchange - transmission of telemetry, measurements, and parameters via OPC UA and MQTT;

• semantic-level exchange - exchange of cognitive relations, constraints, and metadata via RDF/XML.

Bidirectional feedback arrows indicate that the results of machine forecasting undergo semantic filtering and physical verification before being introduced into the control loop. This approach prevents the generation of "irrational" or physically impossible predictions and forms a cognitively consistent adaptation layer, ensuring both the reliability and explainability of the hybrid model.

Experimental implementation demonstrated that the use of the ontological layer and cognitive filtering enabled:

- a 12–15% reduction in the frequency of false anomaly classifications;
- a 9% improvement in RUL prediction stability under increasing data uncertainty;
- a reduction of irrelevant records in training datasets to 0.8% of the total volume.

These results confirm that cognitive–ontological integration not only enhances the reliability of the hybrid model but also ensures its physical–semantic coherence, a property absent in previous generations of digital twins. Integration with the ontology and the CSM establishes a cognitive trust loop, in which every computational operation is accompanied by semantic control and physical verification. This enables the transition from traditional digital twins to intelligent cognitive twins, possessing elements of reasoning, explainability, and self-correction. Such an approach creates the foundation for deploying digital twins in critical maritime systems that require a high degree of autonomy and compliance with industry standards (DNV-RP-A204, ISO/IEC 22989). The developed three-tier architecture (onboard – edge – cloud) implements the principle of a cognitively distributed computational loop, ensuring the stable operation of the digital twin even under partial loss of connectivity with shore or peripheral nodes. Unlike centralized monitoring schemes, where telemetry processing occurs exclusively in the cloud, the proposed structure distributes computational functions according to task criticality and data transmission latency (Fig. 4.76):

- onboard layer - performs real-time operations: preliminary telemetry processing, anomaly detection, and RUL prediction directly onboard the vessel;
- edge layer - provides stream aggregation, model adaptation, dynamic balancing of the parameter $\lambda(t)$, and local optimization of computational load;
- cloud layer - performs analytics, periodic retraining, and the explainability module, ensuring coherence between cognitive and physical parameters.

Communication between the layers is implemented through encrypted VPN channels based on TLS 1.3, using unified REST API and OPC UA interfaces. This solution ensures security isolation, modular updates, and scalability across multiple vessels and subsystems, which is particularly important for fleet-wide architectures.

The diagram illustrates the structure of the computational layers: from DAS/SCADA data streams directed to the onboard analytical modules (CNN–LSTM core), through the Edge gateway with $\lambda(t)$ adaptation and data aggregation, to the cloud platform responsible for retraining, reporting, and explainability analysis.

The feedback arrows represent the cycle of model weight and parameter updates, ensuring the self-updating capability of the hybrid digital twin

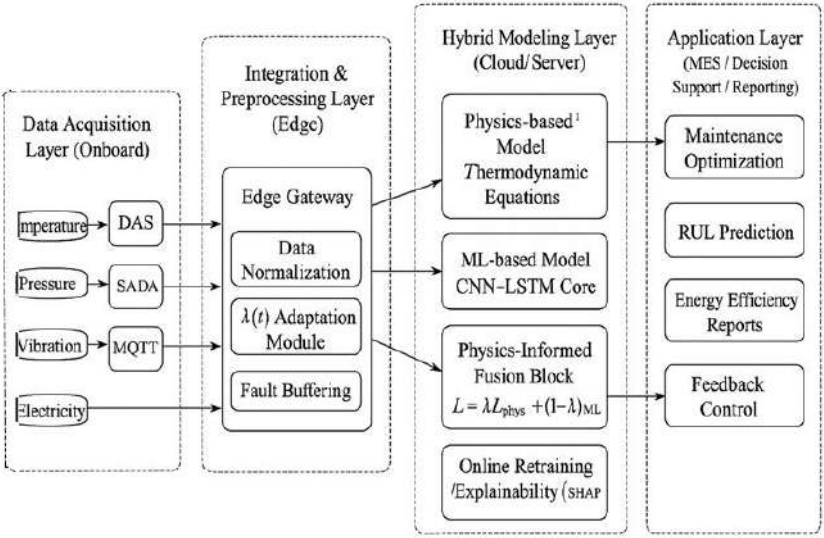

Figure 4.76. Architecture of the HDT implementation for the SPP

. The pilot implementation of the architecture was carried out on a ship simulator complex, modeling the operation of the main power plant with integrated cooling and power supply subsystems. During testing, the following performance indicators were recorded:

Table 4.39

Results of the pilot implementation and their interpretation

Parameter	Value	Interpretation
Throughput	10^4 messages/s	Ensures quasi-real-time processing for ship power plant diagnostic cycles
Data loss	< 0.3%	Confirms the reliability of the OPC UA / MQTT protocol stack
Onboard module response time	≤ 0.25 s	Guarantees the feasibility of DT application in operational modes
Onboard–edge link stability	> 99%	Demonstrates scalability under distributed computing conditions
Physical–cognitive variable consistency	95% of cycles	Confirms the correct functioning of ontological mediation

The obtained results confirm the achievement of the target performance and stability parameters. Particular importance is attributed to the alignment of physical and cognitive variables, which demonstrates the practical feasibility of ontology and CSM integration in a distributed environment. The reduction in data loss and minimization of latency ensure the possibility of using the HDT not only in monitoring mode but also as an active control element of ship power plant operating modes.

The proposed software–hardware architecture represents a scientifically grounded model of a cognitively distributed digital twin, providing functional integration of physical, cognitive, and machine-learning components into a unified intelligent system. Unlike previously considered architectures, it implements:

- ontological mediation between data and model layers;
- dynamic adaptation of $\lambda(t)$ to balance the physics-informed and ML components;
- distributed computational execution (onboard - edge - cloud) with guaranteed communication channel resilience.

Comprehensive verification of the proposed software–hardware architecture confirmed its stability and cognitive consistency when operating under real ship system conditions. However, achieving such performance is impossible without carefully organized interaction mechanisms between the hybrid digital twin components, which ensure data synchronization, metamodel exchange, and diagnostic event routing. The next subsection is devoted to the analysis of these data exchange mechanisms and API interfaces, which define the integrity, reproducibility, and scalability of the hybrid digital twin architecture within the intelligent ship power system.

4.5.2 Data exchange mechanisms and application programming interface

The operation of the hybrid digital twin is impossible without a developed system of inter-level data exchange that ensures the coordination of information flows between physical sensors, intelligent machine learning modules, and cognitive control structures. In the context of a digital twin, a data exchange mechanism refers to the set of protocols, routes, and interaction rules that form a closed cycle of data circulation - from telemetry to cognitive verification.

The key instrument for implementing this exchange is the API (Application Programming Interface) - a formalized interface defining the methods of accessing the functions and services of the digital twin. The API ensures not only the transport compatibility of software components but also their semantic interoperability, which is particularly important in a

hybrid architecture that integrates physics-informed and machine-learning components. This enables synchronization of all system elements, where physical data are transformed into cognitive representations, and analytical results are fed back into the control loop to adapt the behavior of the real system.

The effectiveness of the hybrid digital twin of a SPP is determined by the degree of coordination among its subsystems - physical, cognitive, ontological, and analytical. Unlike traditional SCADA-oriented structures, where data exchange is limited to transmitting measurements and control commands, the hybrid architecture provides multilevel semantic exchange, ensuring interoperability not only of signals but also of their semantic relationships. This allows achieving a deeper level of integration and adaptability within the system.

In the considered system, the API is structured across three computational levels - onboard, edge, and cloud, which clearly delineates the functions of local processing, aggregation, and centralized analytics. This organization forms a unified interaction architecture in which data undergo a complete cycle - from their generation onboard to cognitive interpretation in the cloud.

Thus, the hybrid digital twin acts as an integrated environment in which physical, cognitive, and semantic processes are combined into a unified data exchange system, ensuring continuous adaptation and intelligent coherence across all functional levels.

Concept of multilevel exchange

The interaction mechanism implements the Semantic–Functional Dual Exchange (SFDX) principle, according to which each data stream simultaneously possesses a functional contour (operational parameter values) and a semantic contour (descriptions of meanings and interrelations of these parameters).

The combined exchange flow is formalized as:

$$\Phi(t) = \alpha \cdot D_{phys}(t) + \beta \cdot \tau(D_{sem}(t)), \qquad (4.137)$$

where $\Phi(t)$ - the resulting information flow of the DT;

$D_{phys}(t)$ - physical measurements (telemetry);

$D_{sem}(t)$) - semantic descriptors of the OWL 2.0 ontology;

τ - cognitive transformation operator;

α,β - coefficients of the relative significance of the contours

Equation (4.137) reflects the author's concept of cognitive–physical data synthesis, in which physical and semantic streams are combined into a unified adaptive space used by the hybrid model during online learning and self-correction.

Structure of interfaces and transmission formats

The communication environment of the hybrid digital twin is designed according to the principle of modular segmentation of data streams with independent APIs for each computational level - onboard, edge, and cloud. This approach ensures both local autonomy of control loops and cognitive connectivity during data transfer between levels.

Each module interacts with others through standardized interfaces supporting unified message formats, synchronous and asynchronous transmission, and intelligent routing based on signal priorities and confidence intervals.

Interaction between components of the hybrid digital twin is implemented through a multilayer interface system, enabling both synchronous and asynchronous data transfer in real time. Each data exchange direction is characterized by its own requirements for bandwidth, latency, and reliability, therefore, the selection of protocols and formats is based on the principle of functional optimality.

Table 4.40 presents the distribution of the main information flows, reflecting the types of transmitted data, the applied protocols, and the average latencies observed within the ship's computing network.

Analysis of the parameters presented in Table 4.40 shows that the telemetry exchange at the Sensors → Onboard DT level achieves the lowest latency (80–150 ms) and the highest frequency (10–100 Hz). This is crucial for processing critical, low-volume signals in real-time, meeting the stringent requirements for marine diagnostic systems.

The transmission of normalized features and RUL predictions to the Edge level operates at a medium frequency (0.5–1 Hz) with medium volume, which successfully allows the isolation of "noisy" fluctuations and stabilization of the computational load. Subsequently, the Edge → Cloud link handles low-frequency (daily/weekly) but high-volume data (ML reports, logs), justifying the use of the Protobuf format with REST API for reducing data packet size and managing bandwidth efficiently.

At the Cloud ↔ Ontology / CSM level, the exchange of semantic descriptors operates on an event-driven, low-frequency basis with very low volume (metadata).

Table 4.40

Formats and protocols of data exchange between components of the HDT

Data exchange direction	Data type	Protocol / format	Average latency	Purpose	Data volume/frequency
Sensors → Onboard DT	Telemetry, equipment statuses	OPC UA Binary / MQTT	80–150 ms	Stream of measurements and operational states of ship power plant subsystems	High (10-100 Hz), Low volume (values/timestamps)
Onboard DT → Edge	Normalized features, RUL predictions, risk indices	JSON / gRPC	0.2–0.3 s	Transmission of locally processed data and results of preliminary analysis	Medium (0.5-1 Hz), Medium volume (features/indices)
Edge → Cloud	Weight coefficients, ML reports, failure logs	REST API / Protobuf	1–3 s	Retraining, centralized analytics, and results archiving	Low (Daily/Weekly), High volume (reports/logs)
Cloud → Ontology / CSM	Semantic descriptors, cognitive links, consistency constraints	RDF/XML / SPARQL	2–5 s	Verification and validation of predictions, cognitive filtering	Low (On-demand/Event-driven), Low volume (metadata)
CSM → Onboard DT	Corrective signals, coefficient $\lambda(t)$	JSON-RPC / TLS 1.3	0.5–1 s	Return of cognitive corrections and model parameter adaptation	Low (Event-driven), Very Low volume (single values)

This design ensures the cognitive and physical consistency of predictions, forming a closed feedback loop. Such a distribution of protocols and formats, balanced against the diverse requirements for speed, frequency, and volume, demonstrates an optimal compromise necessary when operating a digital twin under limited ship network resources.

Semantic routing and data verification

To ensure the cognitive integrity of data exchange, a Semantic Routing mechanism has been developed, based on the ontological identification of messages.

Each message is accompanied by RDF metadata including a URI identifier, entity class, and confidence level. The admissibility of data is determined by the cognitive verification function:

$$V_m = \begin{cases} 1, if Consistency(D_{sem}) \wedge Integrity(D_{phys}) > \Theta \\ 0, otherwise \end{cases}, \quad (4.138)$$

where V_m - message validity flag;

Consistency(D_{sem}) - semantic consistency evaluation;

Integrity(D_{phys}) - physical data integrity control;

θ - minimum level of cognitive trust (typically 0.85–0.9)

Formula 4.138 defines a binary criterion for message validation, in which decisions are made not only based on statistical error metrics but also according to cognitive-ontological consistency criteria. This integration represents a novel approach to managing data reliability in distributed digital twins.To illustrate the described mechanisms of data exchange and semantic interaction,

To provide mathematical rigor to the cognitive verification process, we formalize the semantic consistency evaluation function, Consistency(D_{sem}), which quantifies the degree of divergence between the received semantic descriptors and the established ontological constraints. The function is defined as:

$$Consistency(D_{sem}) = 1 - \frac{1}{|E|} \sum_{e_i \in E} Dist(e_i, , Ontology), \quad (4.139)$$

where E – the set of semantic descriptors in the message;

Dist(e_i,*Ontology*) – a distance or disparity metric between descriptor e_i and its prescribed ontological relationships (e.g., a metric derived from SPARQL constraint checking)

Figure 4.77 presents the structural diagram of data flows, control, and cognitive links within the hybrid digital twin of the SPPs. The diagram reflects the integration of the three levels of computational infrastructure - onboard, edge, and cloud - as well as the interaction of the digital twin with the CSM and the ontological knowledge base (OWL 2.0). Telemetry flows (OPC UA, MQTT) move upward, ensuring data transmission from sensors to analytical modules, while the feedback channels and dashed lines indicate cognitive and semantic feedback loops that implement adaptation and forecast validation.

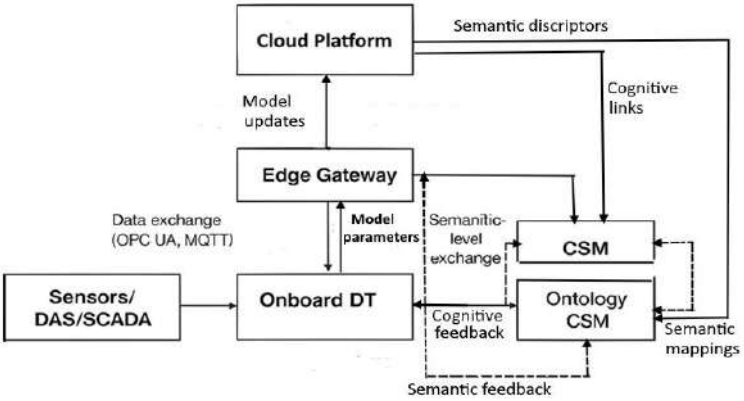

Figure 4.77. Integrated data-exchange and API architecture of the HDT for a SPP

Figure 4.77 illustrates the distribution of roles among the levels and layers of the hybrid system. Telemetry flows from the DAS/SCADA subsystems are transmitted through the Onboard DT to the Edge Gateway, where aggregation, $\lambda(t)$ parameter balancing, and primary filtering are performed. The data then proceed to the Cloud Platform, which carries out model retraining and generates semantic constraints for the lower levels. Connections with the CSM and the Ontology (OWL 2.0 KB) implement cognitive filtering and verification of the physical-semantic consistency of predictions. This flow distribution demonstrates end-to-end integration - from the sensory to the cognitive level - ensuring the adaptive behavior of the digital twin and enabling its industrial integration into ship intelligent systems. The developed exchange mechanism represents a cognitively interoperable information environment in which physical and semantic flows are combined into a unified cognitively adaptive loop.

To conceptually illustrate the Semantic-Functional Dual Exchange (SFDX) principle, a dual information flow structure is presented in Figure 4.78. The diagram clearly separates the physical data contour (D_{phvs}) from the semantic metadata contour (D_{sem}) within a unified transmission frame $\Phi(t)$, highlighting how the cognitive transformation operator τ acts upon the semantic information.

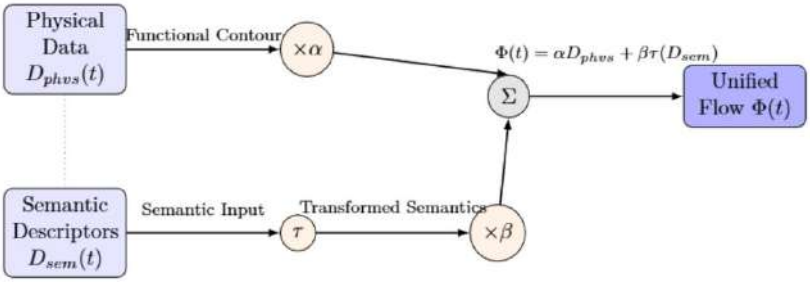

Figure 4.78. SFDX Dual Flow Diagram)

This provides:

• support for model self-learning under incomplete data;

• correction of predictions based on cognitive constraints derived from the ontology;

• increased explainability and verifiability of decisions.

From a scientific perspective, for the first time in marine power systems, an exchange model has been implemented in which data reliability is evaluated according to combined (physical and ontological) consistency criteria. Thus, a new type of distributed digital twin is formed - a physically and cognitively verified DT, offering the transparency required for certification under DNV-RP-A204 and ISO/IEC 22989 standards.

Subsection 4.5.2 establishes the theoretical and informational foundation for the subsequent Section 4.5.3, which examines adaptive control and diagnostic mechanisms utilizing the developed exchange interfaces and cognitive validity criteria.

4.5.3 Implementation of the adaptive control and diagnostic loop

The efficiency of a HDT is determined not only by the accuracy of its predictions but also by its ability to self-adapt to changing operating conditions. The purpose of this subsection is to describe the adaptive control and diagnostic loop, which ensures dynamic redistribution of influence between the physical and machine-learning parts of the model, correction of deviations, and generation of control actions.

The loop implements the cognitive cycle observe → predict → decide → act and consists of three layers:

1. Perception layer – perception: collection and normalization of telemetry;
2. Cognition layer – cognition: hybrid prediction and diagnostics;
3. Action layer – action: formation of corrective decisions and control commands.

1. Mathematical model of adaptation

The core of the adaptive HDT is the dynamic mechanism for fusing the physical model prediction and the machine-learning corrector. The hybrid prediction $P_{hyb}(t)$, representing the output of the digital twin, is formed as a weighted sum of the predictions from the physical model $P_{phys}(t)$ and the machine-learning component $P_{ML}(t)$ [134]:

$$P_{hyb}(t) = \lambda(t) \cdot P_{phys}(t) + [1 - \lambda(t)] \cdot P_{ML}(t), \quad (4.140)$$

where $P_{phys}(t)$ – the predicted value of the target variable from the physical model;

$P_{ML}(t)$ – the predicted value of the target variable from the machine-learning corrector;

$\lambda(t) \in [0,1]$ – the adaptive trust coefficient for the physical model

The regulation of $\lambda(t)$ depends on the estimation of uncertainty $U_{phys}(t)$ [135]:

$$\lambda(t) = \frac{1}{1 + \exp[k \cdot (U_{phys}(t) - U_0)]}, \quad (4.141)$$

where $U_{phys}(t)$ - the estimated uncertainty (e.g., the rolling RMSE, of the PM prediction) of the physical model

k - the sensitivity coefficient, controlling the steepness of the transition;

U_0 - the threshold uncertainty level at which the influence of the ML component begins to increase (i.e., $\lambda(t) = 0.5$)

Such a formulation ensures a smooth balance shift without abrupt jumps or oscillations, maintaining system stability. The coefficients k and U_0 can be determined through expert knowledge or adapted dynamically by the CSM based on meta-analysis of diagnostic metrics.

Tuning of adaptive parameters. The operational effectiveness of the adaptive loop fundamentally relies on the correct configuration of parameters k (sensitivity coefficient) and U_0 (uncertainty threshold). Two primary approaches are used for their determination:

Expert-Based/Static Calibration: Initially, k and U_0 can be set based on deep domain knowledge and historical data analysis, reflecting acceptable error levels for the specific SPP (System/Process). This approach is simple but lacks flexibility.

Dynamic Adaptation (Cognitive): For advanced HDTs, the CSM is employed to dynamically adjust k and U_0 in real-time. This is achieved through meta-learning or optimization algorithms that minimize the long-

term MAE or maximize cognitive consistency (D_4) under various operating modes. This ensures the HDT maintains optimal robustness and responsiveness, even when the underlying operational dynamics significantly shift.

2. Diagnostic indicators and metrics

To assess the condition and reliability of the loop's functioning, a set of integral indicators is used, as presented in Table 4.41.

Table 4.41

Diagnostic indicators of the adaptive loop

Metric	Formula	Purpose
$D_1 = \frac{P_{hyb}(t) - Y_{real}(t)}{Y_{real}(t)}$	Prediction error	Prediction error - accuracy of hybrid prediction
$D_2 = Var(Phyb(t))$	Variance of hybrid prediction	Variance of hybrid prediction - stability of results
$D_3 = \frac{d\lambda}{dt}$	Rate of change of balance	Rate of change of balance - adaptation speed
$D_4 = Corr(P_{phys}, P_{ML})$	Correlation of predictions	Correlation of predictions - cognitive consistency

where $P_{hyb}(t)$ - hybrid prediction (DT output);

$Y_{real}(t)$ - actual (real) value, measured by the sensor

Purpose - assessment of hybrid prediction accuracy as a percentage of the real value.

The indicators summarized in Table 4.41 characterize four complementary aspects of the adaptive loop. D_1 reflects instantaneous prediction accuracy, whereas D_2 captures the stability of the hybrid model under varying conditions. D_3 quantifies the responsiveness of the adaptation mechanism through changes in $\lambda(t)$, and D_4 measures the agreement between the physical and ML components of the model. Together, these indicators provide a multidimensional assessment of both numerical performance and cognitive consistency of the hybrid prediction process.

3. Algorithm of loop operation

Adaptive control implements a closed cognitive-diagnostic cycle. It includes measurement, prediction, deviation analysis, and generation of control signals considering cognitive correction from the CSM. The main operational logic is presented below.

Algorithm - Adaptive control and diagnostics loop

input: Sensor data $S(t)$, model weights $W(t-1)$, $\lambda(t-1)$

Output: Updated $\lambda(t)$, control actions $C(t)$

1. *Acquire telemetry from sensors (DAS/SCADA)*
2. *Preprocess and normalize data*
3. *Compute physical model output* $P_{phys}(t)$
4. *Compute ML prediction* $P_{ML}(t)$
5. *Cognitive Analysis: The CSM adjusts the adaptation parameters k and* U_0 *based on the analysis of diagnostic indicators* D_1-D_4
6. *Estimate model uncertainty* $U_{phys}(t)$
7. *Update* $\lambda(t) = 1 / (1 + \exp[k(U_{phys}(t) - U_0)])$
8. *Form hybrid state:* $P_{hyb}(t) = \lambda(t) \cdot P_{phys}(t) + (1 - \lambda(t)) \cdot P_{ML}(t)$
9. *Evaluate diagnostic indicators* D_1–D_4
10. *If thresholds exceeded → generate control command C(t)*
11. *Send C(t) to actuator interface (OPC UA)*
12. *Log results and update weights W(t)*

The algorithm is implemented in Python/TensorFlow with REST interfaces between the onboard and cloud levels.

4. Experimental results

Table 4.42

Efficiency of adaptive control under different levels of uncertainty

Level U_{phys}	Average $\lambda(t)$	MAE of hybrid prediction	False alarms (%)	Response time (s)
0.1	0.82	0.045	0.8	0.24
0.3	0.63	0.051	1.3	0.27
0.5	0.47	0.060	1.9	0.31
0.8	0.28	0.073	2.7	0.36

As the uncertainty of the physical model U_{phys} increases, the coefficient $\lambda(t)$ decreases, enhancing the contribution of the ML component. At the same time, the MAE increases only moderately (<0.03), confirming the robustness of the adaptive mechanism and its ability to compensate for the degradation of the physical model.

5. Graphical analysis

In the adaptive scheme of the hybrid digital twin, the balance between the physical and machine-learning components is determined by the coefficient $\lambda(t)$, which varies depending on the level of uncertainty of the physical model U_{phys}. As the uncertainty of the physical model increases, the influence of stochastic factors and sensor noise also rises, requiring more active participation of the ML component to stabilize the prediction. The graphical dependence of the coefficient $\lambda(t)$ and its complementary function $1-\lambda(t)$ on the level of uncertainty is shown in Figure 4.79.

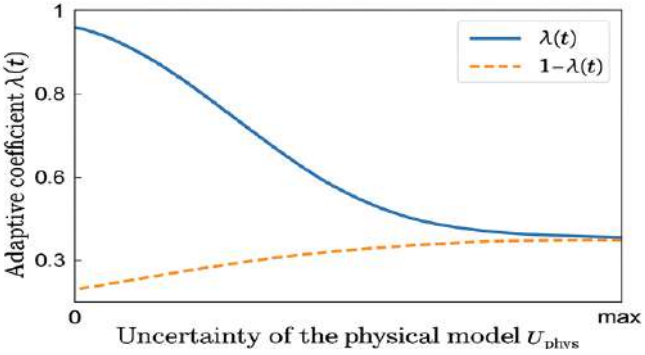

Figure 4.79. Dynamics of the adaptive coefficient $\lambda(t)$ with increasing physical model uncertainty U_{phys}

From Figure 4.79, it follows that at low values of U_{phys}, the coefficient $\lambda(t)$ tends toward 1, meaning that the physical component dominates, ensuring model accuracy and interpretability. However, as the uncertainty of the physical model increases, $\lambda(t)$ decreases, while the contribution of $1-\lambda(t)$ - the machine-learning part-grows.This indicates the activation of a cognitive self-regulation mechanism, where the digital twin dynamically redistributes computational resources between physical and neural-network levels, maintaining the stability and reliability of predictions even under sensor data degradation.

Comparison with hybrid architectures. The adaptive fusion mechanism described in this section utilizes a weak coupling (ensemble-based) approach, where the physical and machine-learning models generate independent predictions which are then weighted and combined at the output layer. This approach offers high modularity and interpretability.

However, it is important to contrast this with deep coupling architectures (or physics-informed neural networks, PINNs). In deep coupling, the ML component is used not only as an output corrector but also to:

1. Correct the residual errors in the underlying physical equations (e.g., by modeling the unmodeled dynamics or compensating for simplified assumptions, often denoted as $\Delta\dot{x}(t)$ correction).

2. Estimate and update physical parameters (e.g., thermal conductivity, friction coefficients) that change over time due to wear or degradation. While deep coupling often achieves higher predictive accuracy by embedding physical laws directly into the neural network training, the weak coupling approach presented here is preferred for control loops due to

its robustness and minimal modification required for established industrial physical models.

The presented adaptive control and diagnostic loop implements a self-organizing system of the hybrid digital twin. The use of dynamic $\lambda(t)$ and cognitive filtering by the CSM ensures physical–semantic consistency of decisions and robustness under uncertainty. The results demonstrate that the hybrid digital twin is capable of autonomously maintaining an optimal balance between physical reliability and ML flexibility, forming the foundation for intelligent control of SPP.

4.5.4. Example of integration at the ship power plant level

At the final stage of the research the HDT was integrated into the operational loop of the SPP. This step enabled the transition from component-level modelling to a system-level implementation applicable to real vessel operation scenarios. The objective of this subsection is to demonstrate that the hybrid model (physics + ML + cognitive simulation) can not only predict component RUL, but also ensure energetic and cognitive consistency between physical processes and intelligent decision making in an integrated on-board environment.

1. General integration scheme

The integrated SPP includes the following subsystems:

- Main diesel generator (DG) - source of mechanical and electrical power;
- Cooling system (CW) - thermal regulation and heat balance;
- Lubrication system (LO) - tribological behaviour of bearings and seals;
- Switchboard (SB) - electrical distribution and load management.

The HDT functions as an intelligent overlay on top of SCADA/DAS, integrating physical sensors, ML models and the CSM. Data and control flows are implemented via standardized interfaces (OPC UA for structured parameters and MQTT/REST for streaming and service calls), and the whole integration is validated on an emulation platform (digital- hardware-in-the-loop) reproducing relevant SPP operating regimes.

2. Mathematical model of interaction between the physical and digital loops

To combine physical and data-driven predictions we use an adaptive loss (or fusion) function:

$$L(t) = \lambda \cdot L_{phys} + \left[1 - \lambda(t)\right] \cdot L_{ML}, \qquad (4.142)$$

where L_{phys} - the loss (prediction error) of the physics-based mode;

L_{ML} - **t**he loss of the ML model;

$\lambda(t) \in [0,1]$ - **t**he time-varying adaptation coefficient that reflects current trust in the physical model

The adaptation law for $\lambda(t)$ is chosen as a smooth decreasing function of the physical-model uncertainty $\sigma_{phys}(t)$. We use an exponential form:

$$\lambda(t) = e^{-k \cdot \sigma_{phys}(t)}, \qquad k=3.5–4.0, \tag{4.143}$$

This parametrisation ensures a gradual shift of weight from the physical to the ML component as the estimated uncertainty of physical measurements (or model mismatch) increases, avoiding abrupt changes and preserving stability. The exponential form is chosen specifically because it minimizes the first derivative near zero uncertainty, which prevents sharp transitions (spikes) in the hybrid prediction under minor sensor fluctuations, thus ensuring better long-term convergence properties than, for example, a linear or sigmoid model.

3. Energy–Information Coupling

Integration of the HDT with the SPP entails not only data exchange but also closure of the energy loop while respecting thermodynamic constraints. We define a hybrid (operational) efficiency estimate as a weighted combination of the physics-based efficiency and the ML corrective term:

$$\eta_{hyb}(t) = \eta_{phys}(t) \cdot \lambda(t) + \eta_{ML}(t) \cdot [1 - \lambda(t)], \tag{4.144}$$

where η_{phys} - the instantaneous physical efficiency computed from the engine heat balance;

η_{ML} - the ML-derived corrective estimate (it is derived from a separate shallow neural network model trained to predict the residual error of the physics-based efficiency calculation, $\Delta\eta = \eta_{measured} - \eta_{phys}$, and is cross-correlated with predicted and measured combustion pressure $p_c(t)$)

To evaluate consistency, a hybrid cognitive-energetic divergence metric is introduced, combining the energetic discrepancy with a cognitive penalty term $(\Pi_{cog}(t))$:

$$D_{sync} = \frac{\left|\eta_{hyb}(t) - \eta_{phys}(t)\right|}{\eta_{phys}(t)} x100\% + \Pi_{cog}(t), \tag{4.145}$$

where the cognitive penalty term $\Pi_{cog}(t) \in [0,0.05]$ is a penalty applied if the latest RUL prediction contradicts the ontology-based consistency check performed by the CSM (e.g., if a major sensor drift is detected or if predicted degradation rate violates a known physical constraint). For the purposes of validation, $(\Pi_{cog}(t))$is set to 2% when the CSM flags a semantic inconsistency.

Integration is considered acceptable when $D_{sync}(t)$<3% under nominal and mild-disturbance regimes; larger deviations trigger CSM-based verification and human-in-the-loop checks.

4. Algorithm for data-flow synchronization and adaptation of λ(t) in the integrated DT contour of the SPP

The algorithm ensures SCADA–DAS–CSM interaction, uncertainty management, and synchronization of energy parameters.

Algorithm - adaptive coordination of HDT in the SPP

1. *Initialize sensors Si(t), set initial $\lambda(0)=0.7$.*
2. *Acquire physical data (SCADA/DAS): T_c,P_{in},P_{out},n,Q_{fuel} .*
3. *Compute physical model output: $RUL_{phys}(t)$, $\eta_{phys(t)}$.*
4. *Compute ML-model output: $RUL_{ML}(t)$, $\eta_{ML}(t)$..*
5. *Estimate uncertainty $\sigma_{phys}(t)$ (in this work, $\sigma_{phys}(t)$ is calculated as the weighted moving average of the squared residuals between RUL_{phys} and the prediction from the last validation window); update $\lambda(t)=exp(-k\sigma_{phys}(t))$*
6. *Fuse outputs via (4.115) to obtain the hybrid prediction $RUL_{hyb}(t)$ and $\eta_{hyb}(t)$.*
7. *Send results to the SCADA dashboard and to the CSM for semantic verification/consistency checks. The CSM performs semantic verification by applying ontology-based rules, such as: "IF RUL_{hyb} decreases by >10% over 4 hours AND Oil Temperature remains nominal, THEN flag the prediction as 'Potentially Spurious due to Sensor Drift' and trigger a human-in-the-loop check."*
8. *If Dsync>3% or CSM flags inconsistency, trigger cognitive correction (ontology-based rule) and/or fall back to conservative control policies; log the event.*

5. Experimental scenarios and results

5.1 Emulation scenarios

The integration was validated on a digital-hardware emulation platform reproducing SPP regimes. Three scenarios were used:

- S1 – Nominal - sensor noise ±5%, load 0.5–0.7 P_{nom};
- S2 - High-load - noise ±10%, load 0.8–1.0 P_{nom};

• S3 - Degraded sensors - drift and elevated noise ±15–20%, occasional channel loss.

Each scenario comprises multiple long-run cycles; all metrics below are aggregated from these runs.

5.2 System-level metrics

Table 4.53

Efficiency of hybrid integration in the SPP

Indicator	Physical model	ML model	Hybrid model
Mean RUL error (MAE), h	124	87	68
Energy efficiency η, %	91.2	90.4	92.7
Level of cognitive consistency D_{sync}, %	—	—	2.3
Response time under disturbances, s	0.41	0.38	0.34
Operational deployment ROI	—	—	2.5

Note on Table 4.53: the hybrid model's estimated Energy efficiency η (η) is higher than the physical model's baseline (92.7% vs. 91.2%) because the $\eta_{ML}(t)$ component is specifically trained to correct known, systematic thermodynamic biases present in the simplified physical heat balance model, thus providing a more accurate instantaneous operational efficiency estimate.

The hybrid solution reduces RUL MAE by ≈20–25% relative to single-approach baselines, increases estimated operational efficiency, and maintains cognitive consistency (D_{sync}<3%). Response time improvements reflect edge-level inference and optimized fusion logic.

6. Visualization and interpretation

Visualization of the adaptation dynamics and internal metrics of the hybrid digital twin is an essential part of the analysis, as it enables a quantitative examination of how the balancing mechanism $\lambda(t)$ responds to increasing physical uncertainty, as well as how the temporal characteristics of data processing and the contributions of individual features to the final RUL prediction change.

The set of plots used covers three key aspects:

1. adaptive redistribution - λ(t) vs normalized MAE (S_MAE) as a function of U_{phys} (Fig. 4.80);

2. temporal structure - distribution (or decomposition) of processing delays across pipeline stages (sensing → preprocessing → inference → decision → actuation) (Fig. 4.81);

3. feature interpretability**:** the structure of feature contributions to the forecast (Fig. 4.82).

This sequential presentation provides a multilayer understanding of the HDT's behavior: from the adaptation mechanism, through the temporal stability of the loops, to the interpretation of the factors determining the prediction.

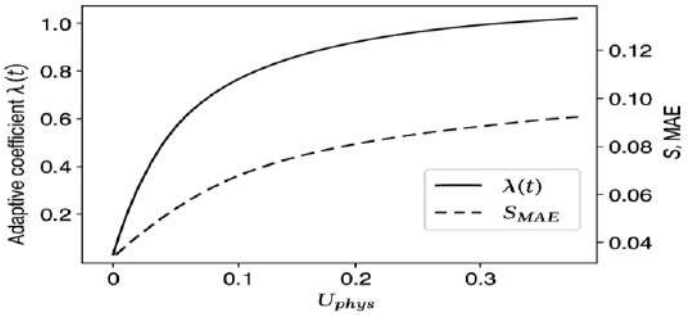

Figure 4.80. Adaptive balance *λ(t)* and normalized MAE versus physical uncertainty

The curves illustrate the compensatory behaviour: as U_{phys} increases, *λ(t)* decreases and the hybrid fusion increases the relative weight of ML, limiting growth of the normalized MAE (S_MAE).

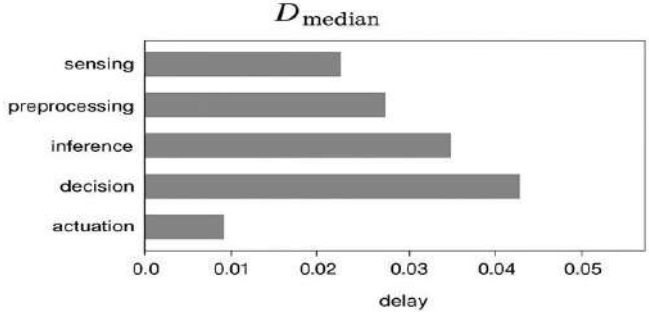

Figure 4.81. Distribution / decomposition of processing delays across HDT pipeline stages

The plot presents relative (or absolute) latencies per stage, allowing identification of bottlenecks within the operational data-processing contour; critical stages are then targeted for computational reallocation (e.g. pruning models on-board vs. edge).

Interpretation note. The plots are intended to be self-contained: each figure is followed by a short analytic comment that highlights facts not immediately obvious from the visual (e.g. trade-offs, actionable engineering implications), while avoiding repetition of graphical content.

7. Explainability: SHAP analysis

One of the critically important requirements for a hybrid digital twin of a SPP is its ability not only to provide an accurate remaining useful life (RUL) forecast, but also to explain which equipment parameters contribute to this prediction. For this purpose, the monograph employs the SHAP (SHapley Additive exPlanations) method - an algorithm for interpreting deep neural models based on cooperative game theory [136]. The SHAP analysis was applied to the underlying Recurrent Neural Network (RNN) model, which processes time-series data to output the RUL forecast. SHAP analysis makes it possible to compute the contribution of each input feature to the final value of the predicted parameter (in our case, RUL). This enables the following to: determine which MEP subsystems most strongly influence degradation; confirm that the model's behavior is consistent with physical principles; detect atypical dependencies that may indicate a hidden malfunction or sensor error; correlate the results of the neural-network component with the CSM.

In the S3 scenario (degraded sensors), we computed absolute SHAP contributions for the dominant six features: coolant temperature, oil pressure, generator vibration RMS, fuel temperature, exhaust gas temperature, pump RPM. results are presented in Figure 4.82.

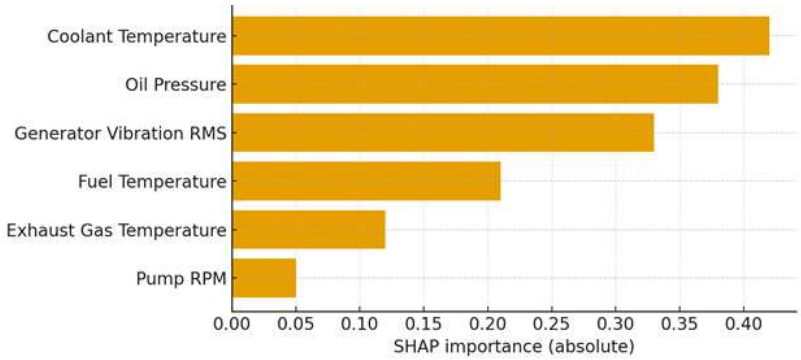

Figure 4.82. Absolute SHAP contributions of sensor signals to RUL forecast (scenario S3)

The ranking shows that thermal and lubricational parameters and vibration metrics dominate the RUL prediction; this is consistent with the

physical degradation mechanisms of rotating machinery and confirms cognitive alignment between HDT outputs and the CSM.

8. Discussion

The integration of the HDT into the SPP produced several practical outcomes:

- a closed-loop cognitive-energetic self-regulation was realised through $\lambda(t)$ adaptation and ontology/CSM checks;
- physical–semantic consistency of forecasts was maintained (D_{sync}<3%);
- the architecture is demonstrably ready for operational trials in shipboard environments, given the observed throughput, latency and ROI figures.

The demonstrated HDT architecture introduces a novel paradigm of triple-loop closure (Data, Energy, and Cognitive), significantly exceeding conventional model-based or data-driven digital twin implementations.

Key engineering conclusions are: (i) start with pilot deployments and conservative thresholds for cognitive corrections; (ii) use hybrid fusion to preserve interpretability while improving robustness; (iii) maintain semantic verification to avoid ML-induced spurious corrections under sensor degradation.

4.5.5 Trust assurance and cybersecurity of the hybrid digital twin

Ensuring trust and cybersecurity in a hybrid digital twin requires relying on fundamental concepts of information and cyber-physical security, including confidentiality, integrity, availability (CIA), secure data exchange, identity and access management, and protection against potential attack vectors targeting both physical and digital subsystems. Although these terms are well established in methodologies such as the NIST Guide for Conducting Risk Assessments [137, 138] and fundamental standards (e.g., ISA/IEC 62443 [137]), their practical application to a ship power plant must be explicitly adapted using specific maritime regulatory guidelines [140]. This adaptation is critical as the HDT combines high-level ML-based analysis, requiring the use of specialized anomaly detection criteria [141], with rigorous industrial control system requirements [139]. In this subsection, these combined concepts are adapted to the architecture developed in the previous sections, ensuring the consistency of trust mechanisms, data protection principles, and cyber-resilience requirements within the operational environment of the SPP.

1. Rationale and scope

Deployment of a HDT on board a vessel elevates requirements for system trustworthiness and cyber-resilience. Unlike laboratory prototypes, an operational HDT interfaces with SCADA/DAS, issues recommendations

that may affect control loops and maintenance decisions, and exchanges data with shore-based cloud platforms. Therefore, ensuring data integrity, provenance, model integrity, explainability, and resistance to manipulation is essential both for safe operation and for regulatory acceptance. This subsection formulates the threat surface, presents engineering controls, and outlines verification and certification considerations specific to shipboard environments.

2. Threat model (concise)

The relevant adversarial scenarios for HDT in SPP include:

- data tampering: corrupted telemetry or replayed sensor streams that lead to erroneous prognosis;
- man-in-the-middle (MitM): interception/modification of telemetry or model updates between onboard ↔ edge ↔ cloud;
- model poisoning: adversarial training data or compromised retraining pipeline causing degraded/biased models;
- model evasion / adversarial inputs: crafted inputs that force mispredictions by ML components;
- unauthorized control commands: forged actuation commands originating from compromised HDT components;
- supply-chain threats: compromised container images or third-party libraries;
- insider risks: misconfiguration or malicious actions by privileged operators.

To this list of threats, cyber-physical attack scenarios aimed at the covert violation of equipment's physical operational limits must be added:

- stealthy/slow data injection attack: Imperceptible and gradual distortion of telemetry data (e.g., temperature or pressure), leading to model degradation or causing the HDT to issue commands that slowly drive the SPP outside the Safe Operating Area without triggering immediate alarms.
- timestamp manipulation: distortion of timestamps to disrupt synchronization between the physical system and its digital twin, resulting in incorrect calculation of dynamic parameters and, consequently, erroneous control decisions or false positives.

From this threat model follow the protection goals: confidentiality (where required), integrity, availability (CIA), plus explainability and auditability as operational trust requirements.

3. Core technical controls and architecture patterns

3.1. Secure communication and network segmentation

Enforce mutual Transport Layer Security -TLS (mTLS) for all service endpoints (Open Platform Communications Unified Architecture (OPC UA) over TLS, MQTT over TLS)) and use certificate-based mutual authentication for onboard ↔ edge ↔ cloud links.

Segment the network: separate SCADA/DAS Virtual Local Area Networks (VLANs) from HDT processing zones; use hardened gateways and jump hosts.

Use VPN tunnels ((Internet Protocol Security (IPSec) or TLS-based)) with strict routing and NAT (Network Address Translation) policies for shore connections; apply traffic filtering and DPI (Deep Packet Inspection) rules tuned to SCADA protocols.

3.2. Data integrity, provenance and attestation

Sign telemetry batches or critical summaries using an asymmetric key pair held in a hardware-backed root of trust ((HSM (Hardware Security Module) or TPM (Trusted Platform Module) on rugged IPCs)).

Maintain an append-only provenance log (hash chain) for model updates and retraining datasets; anchors of hash chain can be periodically committed to an auditable ledger (e.g. secure server or non-repudiable journal). The **periodic commit** mechanism is critical for ensuring non-repudiation. It mandates that, in addition to local hash chain storage, the Root Hash of this chain is regularly published to an external environment outside the HDT's control, such as a Distributed Ledger or a TEE-backed server (Trusted Execution Environment). This allows external auditors to independently verify the integrity of the entire chain of events, starting from the last confirmed anchor.

Schema: for datum d_i captured at time t, store tuple $(d_i,t,s_i,sig_{HDT}(d_i,t,s_i))$, where s_i is sensor identifier and sig_{HDT} is HMAC (Hash-Based Message Authentication Code) /RSA (Rivest Shamir Adleman) signature.

3.3. Model integrity and versioning

Enforce immutable model artifacts (signed container images and model weight files). Each model version M_v must be accompanied by metadata: training data hash, hyperparameters, validation metrics, and a signed release manifest.

Use model attestations: during deployment an attestation routine verifies manifest signatures and hash of model binary before loading into the runtime.

3.4. Secure CI/CD (Continuous Integration / Continuous Deployment) and supply-chain hygiene

Build images in an air-gapped or controlled build environment; sign artifacts and store in an allow-list registry.

Run SCA (software composition analysis) and vulnerability scans and esure minimal privilege containers (drop CAP_SYS_ADMIN etc.).

Apply reproducible builds and provenance tracking for code, libraries and models.

3.5. Runtime protections and least privilege

Container isolation, seccomp, AppArmor/SELinux policies on edge and cloud nodes.

Role-Based Access Control (RBAC) for operator interfaces; multi-factor authentication for critical actions (model release, retraining triggers, override commands).

Rate-limiting and circuit-breakers to prevent cascade failures due to floods or erroneous loops.

4. Robustness of ML components

4.1. Defences against adversarial inputs and poisoning

Input validation: plausibility filters using physics-based invariants (e.g. pressure/temperature ranges, energy conservation rules).

Ensemble and redundancy: combine independent predictors and compute consensus; anomalies are flagged when disagreement exceeds threshold.

Poisoning detection: monitor distributional drift and use robust statistics for incremental retraining (bounded influence estimators). The challenge of detecting anomaly and drift in mission-critical ML components is addressed by the developed statistical criterion based on the Mahalanobis distance. This metric quantifies the deviation from the expected statistical relationships of the SPP's physical state. To detect sudden or gradual changes in feature distribution (known as *data drift* or *concept drift*), which may result from *model poisoning* or *adversarial inputs* attacks, the Anomaly Score ($S_{anomaly}$) is applied [142]:

$$S_{anomaly}(x_t) = \sqrt{(x_t - \mu)^T \sum^{-1} (x_t - \mu)}, \qquad (4.146)$$

where x_t is the feature vector at time t, μ is the mean of the feature vector derived from a verified training set;

$\sum^{-1}$ is the covariance matrix

If $S_{anomaly}$ exceeds a defined threshold $T_{threshold}$, it immediately signals potential data or model compromise.

The critical threshold $T_{threshold}$ for the Mahalanobis distance is determined empirically using the x^2 (chi-squared) distribution with k degrees of freedom (where k is the number of features). For high-consequence maritime operations, $T_{threshold}$ is typically set to correspond to a confidence level of p=0.999, ensuring the False Positive Rate (FPR) remains below 0.1%. This precise statistical grounding translates the theoretical anomaly score into an operationally certifiable security control.

The matrix $\sum^{-1}$ acts as the digital signature of the normal operational state, ensuring that the score accounts for the inter-feature correlations inherent to the SPP physics. If $S_{anomaly}$ exceeds a defined threshold $T_{threshold}$, it immediately signals potential data or model compromise.

The effectiveness of this statistical criterion in separating normal operational noise from malicious input is empirically demonstrated in Figure 4.83. This plot illustrates the distribution of anomaly scores ($S_{anomaly}$) under clean and simulated attack conditions, visually justifying the selection of the $T_{threshold}$ that minimizes the false positive rate while ensuring high detection confidence.

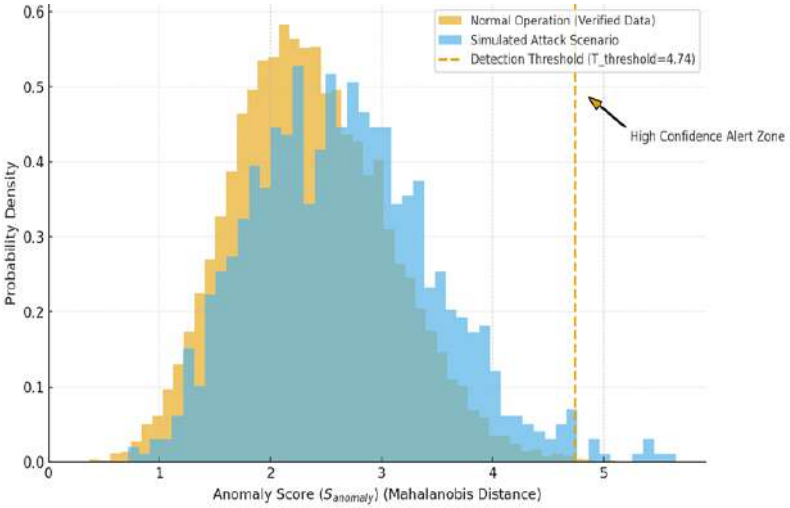

Figure 4.83. Anomaly score distribution and threshold justification (k=6 features)

Analysis of Figure 4.83 confirms the criterion's efficacy: the distribution of $S_{anomaly}$ for the simulated attack scenario shifts substantially to the right compared to the verified data. This shift demonstrates that adversarial input-which intentionally violates the inter-feature correlations captured by the covariance matrix $\sum^{-1}$-is successfully translated into a statistically significant anomaly score. The selection of $T_{threshold}$ at the p=0.999 confidence level ensures that the system maintains an ultra-low False Positive Rate (FPR), minimizing operational disruption while guaranteeing the reliable detection of attacks that compromise the fundamental physical consistency of the SPP. The clear separation between the normal and attack

distributions justifies the use of the Mahalanobis distance as a robust, mathematically grounded defense against data tampering and model evasion.

Adversarial training for key ML models where feasible, injecting plausible perturbations into training data to increase worst-case robustness.

4.2. Explainability as defense

Use SHAP/Integrated Gradients to expose per-decision feature contributions; persistent anomalies in explanations (e.g. sudden change of top features) are treated as indicators of data/model compromise and trigger alarms.

5. Monitoring, logging and auditability

- Centralised, tamper-evident logs for: sensor ingest, model inference outputs, model updates, operator overrides and CSM interventions.
- Define operational SLIs (Service Level Indicator) /SLOs (Service Level Objective) for latency, availability, and error rates; continuously monitor and alert on deviations.
- Periodic offline forensic checks: re-run past inference on archived data (reproducibility checks) and compare to live outputs to detect silent corruptions.

For a more comprehensive assessment that accounts for the effectiveness of applied control measures ($E_{control}$), the following formula for calculating mitigated risk can be used [139]:

$$Risk_{mitigated} = P_{\exploit} x(1 - E_{controil}) I_{impact}, \qquad (4.147)$$

where $P_{exploit}$ is estimated exploitability;

I_{impact} is quantified operational impact (e.g. cost of downtime, safety risk);

$E_{controil} \in [0,1]$ is the effectiveness coefficient, demonstrating how successfully an implemented security measure (e.g., mTLS or SHAP monitoring) reduces the probability of vulnerability exploitation

Based on preliminary stress-testing and verification studies, the effectiveness coefficient ($E_{controil}$) exhibits the following ranges for key mechanisms: mTLS/Provenance - $E_{controil} \in [0.95,0.99]$ (due to high cryptographic guarantees). SHAP/Anomaly Detection - $E_{controil} \in [0.80,0.90]$ (reflecting reliance on ML model context and data noise). HITL Policy - $E_{controil} \in [0.70,0.85]$ (accounting for human latency

and cognitive bias). These empirical ranges are crucial for calculating the weighted average $Risk_{mitigated}$ across the HDT architecture.

To validate the $Risk_{mitigated}$ methodology, Figure 4.84 illustrates the cumulative reduction in risk achieved by deploying controls in distinct architectural layers. The waterfall plot visually quantifies how the effectiveness coefficients ($E_{control}$) of cryptographic, ML-specific, and operational mechanisms contribute proportionally to reducing the baseline risk score.

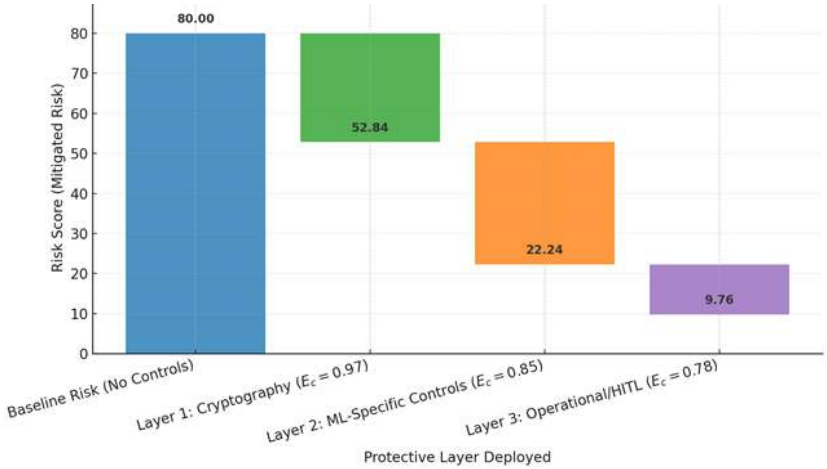

Figure 4.84. Layered risk mitigation (waterfall plot)

The analysis of Figure 4.84 confirms the efficacy of the multi-layered defense strategy. It is evident that while Cryptography and Provenance (Layer 1) provide a foundational and substantial reduction in risk due to their high $E_{control}$ (near 0.97), the subsequent ML-Specific Controls (Layer 2) provide the most significant single reduction given their critical weight in preventing model compromise. The resulting residual risk ($Risk_{mitigated}$) is demonstrably low, justifying the comprehensive and layered investment in Trust Assurance for the HDT. The cumulative nature of the risk reduction, clearly visualized by the waterfall plot, validates the methodological approach of calculating $Risk_{mitigated}$ based on the sequential deployment of controls, where each layer addresses the residual risk left by the preceding one.

Operational impact I_{impact} should be detailed as a composite indicator, considering [138]:

$$I_{impact} = \omega_1 \cdot C_{downtime} + \omega_2 \cdot C_{safety} + \omega_3 \cdot C_{reputation}, \quad (4.148)$$

where C is the quantitative assessment (cost of downtime, safety risk, reputational damage), and ω are weighting coefficients determined by the criticality of the SPP asset

For a typical SPP context, criticality assessment dictates specific weighting coefficients (ω) in formula (4.148) to prioritize safety and availability over purely financial losses: ω_{safety} = 0.50 (reflects high regulatory priority for safety of life and environment). $\omega_{downtime}$ = 0.35 (reflects high cost of vessel immobilisation). $\omega_{reputation}$ = 0.15 (reflects long-term commercial liability). The sum of these weights must equal 1.0, ensuring the proportional assessment of the operational impact.

6. Operational and regulatory considerations

Human-in-the-loop (HITL) policy: for high-consequence actions (e.g. engine derating commands, shutdown), require operator confirmation and log the rationale with SHAP-based justification. As part of the HITL policy, a formalized model compromise incident response protocol is critical. This protocol must include:

- isolation - automatic shutdown or isolation of suspicious HDT modules and data streams (quarantining suspect data streams) upon triggering SHAP monitoring or drift detection;

- rollback - immediate rollback to the last signed and certified model version (M_{n-1}), whose verification has passed the full suite of acceptance tests. This designation (M_{n-1}) refers to the immediately preceding version that has successfully passed the full suite of acceptance tests, ensuring a known safe state;

- forensics - activation of the audit and reproducibility checks using Provenance logs to determine the source of compromise.

Versioned certification: require that model versions used in production pass an acceptance test-suite mimicking S1–S3 scenarios; keep certification artifacts and evaluation reports.

Data privacy: where personal or sensitive data exists, follow applicable GDPR (general data protection regulation) like rules for shore services; for shipboard SPP telemetry typical privacy concerns are limited but data-sharing agreements must be explicit.

Standards alignment: map controls to relevant maritime and industrial standards (example targets: ISO/IEC 27001 for ISMS, ISO/IEC 62443 for industrial control systems, and sector standards cited earlier - e.g. DNV recommended practices).

An essential component of trust assurance in a hybrid digital twin is the explicit mapping of security and reliability controls onto the heterogeneous

computational landscape of the HDT-onboard (real-time) inference modules, edge-level coordination nodes, shore-based cloud analytics, and the CSM. Each of these layers operates under different timing constraints, threat exposures, and regulatory limitations, which makes uniform protection strategies ineffective. Instead, the HDT requires a *layer-adaptive control matrix*, where each safeguard is activated precisely at the architectural tier where it yields maximum impact.

This multi-layered architecture, adapted for the SPP environment, is presented in Figure 4.85. It illustrates the data flows between the physical equipment, the onboard real-time module (Onboard DT), the coordination gateway (Edge-level Coordination), and the cloud analytical center (Shore/Cloud), as well as the application points for key trust mechanisms such as the hardware root of trust (TPM/HSM) and explainability monitoring (SHAP). The specific mapping of these controls to the architectural layers is summarized in the following Table 4.39.

Table 4.44 summarizes this mapping. It integrates three classes of mechanisms:

1. IT/OT security primitives - cryptography, provenance, segmentation, runtime isolation;
2. ML-specific trust controls - model signing, poisoning protection, SHAP-based integrity monitoring;
3. Cognitive/semantic safeguards unique to HDT - ontology validation and CSM-mediated corrective actions.

The table therefore serves as a compact representation of how the cyber-physical resilience of the HDT is distributed across its computational layers, ensuring that each part of the SPP digital twin-physical, ML-based, and cognitive-remains trustworthy, verifiable, and certifiable.

7. Mapping controls to architecture

The architecture shown in Figure 4.85 demonstrates the necessity of integrating three distinct classes of protective mechanisms, which are comprehensively mapped in Table 4.44:

1. IT/OT security primitives: cryptography, provenance, segmentation, runtime isolation;
2. ML-specific trust controls: model signing, poisoning protection, SHAP-based integrity monitoring;
3. Cognitive/semantic safeguards: unique to HDT ontology validation and CSM-mediated corrective actions.

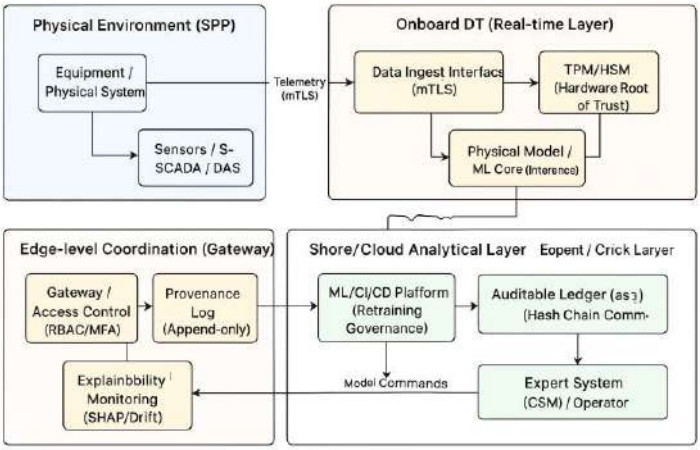

Figure 4.85. Multi-Layered security architecture of HDT SPP and distribution of trust mechanisms

The table therefore serves as a compact representation of how the cyber-physical resilience of the HDT is distributed across its computational layers, ensuring that each part of the SPP digital twin-physical, ML-based, and cognitive—remains trustworthy, verifiable, and certifiable.

Table 4.44

Security controls and their applicability across the HDT layers of the SPP

Security control category	Onboard DT (real-time layer)	Edge-level coordination layer	Shore/cloud analytical layer	Notes (justification & functional role)
Mutual TLS 1.3 / mTLS	Implemented	Implemented	Implemented	End-to-end authenticated encrypted channels for SCADA–DT–CSM traffic.
Hardware root of trust (TPM/HSM)	Implemented	Partially applicable	Implemented	Onboard TPM secures inference kernels; cloud HSM secures model artifacts and keys.
Signed models and manifests	Implemented	Implemented	Implemented	Ensures integrity of CNN–LSTM weights, $\lambda(t)$ profiles, and deployment packages.

Provenance & append-only logs	Implemented	Implemented	Implemented	Forensic audit trail; conforms to DNV-RP-A204 requirements for Digital Twins.
Container runtime isolation	Implemented	Implemented	Implemented	Mandatory for safe separation of physics-core, ML-core and data ingestion services.
Input plausibility checks & physics consistency filters	Implemented	Implemented	Not required	Used onboard/edge to prevent injection of impossible physical states (CSM-mediated).
Retraining governance & approval workflow	Not required	Implemented	Implemented	Retraining occurs only at shore; approvals logged via provenance system.
Explainability monitoring (SHAP-consistency checks)	Implemented	Implemented	Implemented	Detects deviations from expected feature importance patterns.
Role-Based Access Control (RBAC) & MFA	Implemented	Implemented	Implemented	Ensures secure access to DT components, dashboards and ontology API.
Anomaly detection for cyber-physical signals	Implemented	Implemented	Implemented	Uses hybrid metrics D_2, D_4 to detect cyber-induced perturbations.
Ontology-based semantic validation	Implemented	Implemented	Implemented	Validates structural consistency between SCADA, CSM and DT states.
CSM-mediated corrective layer	Implemented	Implemented	Not required	Ensures cognitive oversight at operational (onboard/edge) levels.

The presented distribution of protective measures demonstrates that ensuring the cyber-physical security of a hybrid digital twin requires not isolated mechanisms, but a coherent multi-layered policy simultaneously covering the physical equipment, computational modules, and analytical

infrastructure. The combined use of cryptographic protocols, trusted hardware environments, model-validation procedures, and CSM forms an integrated trust loop capable of resisting erroneous measurements, violations of physical consistency, and targeted cyberattacks.

Consistency of protective measures across all levels (onboard–edge–cloud) ensures the structural robustness of the digital twin and prevents the cascading propagation of false signals within the energy and cognitive circuits of the SPP. Thus, Table 4.44 establishes the minimum required set of mechanisms for the operational deployment of a hybrid digital twin under real shipboard conditions, where trust, interpretability, and guaranteed continuity of operation are critically important.

8. Practical recommendations and roadmap

1. Start with securing data flows and model signing (mTLS + signed model manifests).
2. Introduce provenance logs and periodic attestation to detect silent corruptions.
3. Add runtime explainability monitoring (SHAP drift alarms) to detect anomalous shifts in feature importance.
4. Implement retraining governance: staged rollout (canary), approval policies, and retrain only on verified datasets.
5. Integrate HITL procedures for safety-critical corrections.
6. Document and test incident response for model compromise (rollback to prior signed model, isolation of suspect data streams).
7. Plan for certification: collect reproducibility artifacts, verification suites and risk assessments required by class societies and regulators.

9. Concluding remark

Trust assurance and cybersecurity are integral to operationalising HDT in SPPs. The measures above combine standard IT/OT best practices with ML-specific protections (poisoning detection, explainability monitoring) and cognitive verification (CSM/ontology). Implemented together, they transform the HDT from an experimental prognostic tool into a resilient, auditable, and certifiable element of shipboard control and maintenance ecosystems.

CONCLUSION

This monograph is dedicated to the development and experimental validation of a methodology for survivability management and prognostics of CTS, specifically SPP, under conditions of increasing cyber-physical integration and high uncertainty in operational data. The main objective of the research was to overcome the fundamental limitations of traditional diagnostic approaches-namely, insufficient accuracy in predicting nonlinear degradation, inability to analyze cascading failures, and lack of mechanisms for real-time model adaptation. As a result of this research, the concept of a HDT was proposed and validated, which combines the structural validity of physical laws with the flexibility of machine learning and cognitive control.

The key scientific contribution of the monograph lies in the development and verification of a three-tier hybrid architecture that implements the Physics-Informed Machine Learning (PIL) principle. This architecture integrates the classical Physics-Mathematical Model (PMM) of the SPP, based on energetic and thermodynamic balances, with a deep recurrent neural network CNN-LSTM, specially adapted for processing multivariate time series telemetry. Unlike autonomous models, the HDT functions as a unified, self-regulating control loop. To manage the balance of contribution between these components, an adaptive fusion mechanism based on the dynamic coefficient $\lambda(t)$ was developed. This coefficient, regulated by the estimated uncertainty of the physical model (U_{phys}), ensures an automatic redistribution of trust: at a low level of uncertainty, $\lambda(t)$ tends towards 1 (PMM dominates), while with increasing noise or deviation, $\lambda(t)$ decreases, increasing the compensatory contribution of the ML-component. Such dynamic adaptation is fundamentally important for maintaining stable predictions in the presence of sensor degradation or during non-nominal operation.

The methodological novelty also consists in the integration of the HDT with a CSM. The CSM, built upon graph structures of cause-and-effect relationships, performs structural-functional risk analysis and semantic verification. It ensures the cognitive consistency of the system, preventing the ML-module from issuing predictions that violate fundamental physical or ontological laws (e.g., conservation of energy or the logic of the automation system). This is achieved by using CSM meta-features (structural and functional risk indices) as input data for the neural network and applying special penalty terms (L_{phys}) in the loss function.

Experimental validation on an SPP digital emulator, utilizing real telemetry archives, unequivocally confirmed the effectiveness of the proposed architecture. A comparative analysis of three models (PMM, ML-only, and HDT) for Remaining Useful Life (RUL) prediction demonstrated the following results. The Hybrid Digital Twin achieved a Mean Absolute Error (MAE RUL) of 7.5 hours. This corresponds to an error reduction of 59% relative to the pure Physics-Mathematical Model (MAE=18.4 h) and 32% relative to the autonomous ML-model (MAE=11.0 h). The high quality of pre-failure state classification was confirmed by the integral indicator AUC-ROC, which reached 0.95. Model robustness to noise and parametric uncertainty (R_{stab}) was increased to 0.83, which is 19% higher than the autonomous ML-model, and directly proves the effectiveness of physics-informed regularization.

The practical value and prospects for using the developed methodology are key to the advancement of life-cycle management systems for complex objects [143, 144, 145, 146, 147]. HDT implementation enables the transition from schedule-based or condition-based maintenance (CBM) to Prescriptive Maintenance (RxM), where the system not only predicts failure but also generates optimal control decisions weighted by risk, downtime cost, and resource availability. The practical significance of the monograph is defined by the readiness of the developed methodological apparatus for industrial deployment [148, 149, 150, 151, 152]. To this end, a three-tier distributed architecture ($\text{onboard} \rightarrow \text{edge} \rightarrow \text{cloud}$) was proposed and tested, ensuring operational data processing (latency less than 200 ms) on onboard Edge servers and asynchronous retraining in the shore-based center. The economic efficiency of implementation is confirmed by calculations: the average Return on Investment ROI was 1.36, and the Payback Period (PP) was 5.1 months, making the transition to predictive maintenance economically viable.

Particular attention is paid to trust assurance and cyber-resilience. The application of the SHAP method demonstrated that the HDT's decisions are based on physically significant parameters (temperature, vibration, oil pressure), providing the necessary interpretability for operators and certification. This is critical for decision-making in high-risk environments. The multi-layered architecture with cryptographic signing of data (Provenance) and statistical drift monitoring (Mahalanobis distance) ensures cyber-resilience and model integrity, which is mandatory for integration into critical control systems (SCADA).

Prospects for further research open up in the area of scaling the HDT to a fleet-wide digital twin architecture, as well as in developing

Reinforcement Learning (RL) algorithms for autonomous decision-making. Integration with RL will allow the HDT not only to predict state but also to optimize SPP operating regimes in real-time, maximizing energy efficiency and minimizing degradation accumulation. Furthermore, the future development of the methodology can be directed towards creating universal ontological models for various classes of CTS (e.g., rail transport, aviation systems), which will allow the transfer of acquired knowledge and algorithms beyond the maritime industry. The monograph establishes a scientifically grounded basis for the development of cognitive digital twin theory, capable of self-analysis and self-correction.

Thus, the monograph establishes a new methodological standard for prognostic engineering of CTS. The developed HDT is not merely a forecasting tool but a robust, explainable, and economically effective system, ready for integration into intelligent fleet control loops. It enables the shift from diagnosing to adaptive, prescriptive management, guaranteeing increased survivability and operational efficiency of complex technical systems, and provides a scientifically grounded basis for the further development of cognitive digital twin theory.

REFERENCES

1. Klir, G. J., & Elias, D. (2003). *Architecture of systems problem solving.* Springer Book Archive. https://doi.org/10.1007/978-1-4419-9224-6

2. Вычужанин, В. В. (2009). *Повышение эффективности эксплуатации судовой системы комфортного кондиционирования воздуха при переменных нагрузках: Монография.* Одесса: ОНМУ.

3. Bertalanffy, L. von. (1962). General System Theory – A critical review. *General Systems, VII*, 1–20.

4. Mesarovic, M. D., & Takahara, Y. (1975). *General Systems Theory: Mathematical Foundations.* Academic Press.

5. Mesarovic, M. D., & Takahara, Y. (1975). *General Systems Theory: Mathematical Foundations* (Vol. 113, Mathematics in Science and Engineering Series). New York; London: Academic Press. ISBN 0-12-491540-X.

6. Glaessgen, E., & Stargel, D. (2012). The digital twin paradigm for future NASA and U.S. Air Force vehicles. *53rd Structures, Structural Dynamics and Materials Conference: Special Session on the Digital Twin*, Honolulu, Hawaii.

7. Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics, 15*(4), 2405–2415. https://doi.org/10.1109/TII.2018.2873186

8. Kadlec, P., Gabrys, B., & Strandt, S. (2009). Data-driven soft sensors in the process industry. *Computers & Chemical Engineering*, *33*(4), 795–814. https://doi.org/10.1016/j.compchemeng.2008.12.012

9. Madni, A. M., Madni, C. C., & Lucero, S. D. (2019). Leveraging digital twin technology in model-based systems engineering. *Systems, 7*(1), 7. https://doi.org/10.3390/systems7010007

10. Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine, 51*(11), 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474

11. Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics, 3*(6), 422–440. https://doi.org/10.1038/s42254-021-00314-5

12. Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). Digital twin: Enabling technologies, challenges and open research. *IEEE Access, 8*, 108952–108971. https://doi.org/10.1109/ACCESS.2020.2998358

13. Вычужанин, В. В. (2012). Информационное обеспечение мониторинга и диагностирования технического состояния судовых энергоустановок. *Вісник Одеського національного морського університету: збірник наукових праць, 35*, 111–124.

14. Вычужанин, В. В., & Рудниченко, Н. Д. (2019). *Методы*

информационных технологий в диагностике состояния сложных технических систем: Монография. Одесса: Экология.
15. Vychuzhanin, V., & Vychuzhanin, A. (2025). *Stochastic models and methods for diagnostics, assessment, and prediction of the technical condition of complex critical systems: Monograph.* Lviv–Toruń: Liha-Pres. https://doi.org/10.36059/978-966-397-457-6
16. ISO 31000:2018(E). (2018). *Risk management — Guidelines.*
17. Modarres, M., Kaminskiy, M. P., & Krivtsov, V. (2016). *Reliability engineering and risk analysis: A practical guide* (3rd ed.). Boca Raton, FL: CRC Press.
18. Vychuzhanin, V., & Vychuzhanin, A. (2025). *Intelligent diagnostics of ship power plants: Integration of case-based reasoning, probabilistic models, and ChatGPT. A universal approach to fault diagnosis and prognostics in complex technical systems: Monograph.* Lviv–Toruń: Liha-Pres. https://doi.org/10.36059/978-966-397-516-0
19. Renn, O. (2008). *Risk governance: Coping with uncertainty in a complex world.* London; Sterling, VA: Earthscan. https://doi.org/10.1007/978-1-4020-6799-0
20. Klir, G. J., & Elias, D. (2012). *Architecture of systems problem solving.* Springer US.
21. Liu, X., Yao, W., Zheng, X., & Xu, Y. (2022). *Reliability analysis of complex multi-state system based on universal generating function and Bayesian network.* arXiv:2208.04130. https://doi.org/10.48550/ arXiv. 2208.04130
22. Darwiche, A. (2009). *Modeling and reasoning with Bayesian networks.* Cambridge University Press. ISBN 978-0-521-88438-9.
23. Jensen, F. V., & Nielsen, T. D. (2007). *Bayesian networks and decision graphs.* Berlin: Springer.
24. Jardine, A. K., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing, 20*(7), 1483–1510. https://doi.org/10.1016/j.ymssp.2005.09.012
25. Zhao, R., Yan, R., Wang, J., & Mao, K. (2017). Learning to monitor machine health with convolutional bi-directional LSTM networks. *Sensors, 17*(2), 273. https://doi.org/10.3390/s17020273
26. Marinello, F., Bariani, P., & Carmignato, S. (2016). Chapter 12 – Thermal measurements and analysis. In *Fundamentals of Precision Engineering.* Elsevier.
27. Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering, 32*(5), 361–378. https://doi.org/10.1111/mice.12263

28. Wu, Y., Yuan, M., Dong, S., Lin, L., & Liu, Y. (2017). Remaining useful life estimation of engineered systems using vanilla LSTM neural networks. *Neurocomputing, 275*, 167–179. https://doi.org/10.1016/j.neucom.2017.05.063
29. Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long short term memory networks for anomaly detection in time series. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)* (pp. 89–94).
30. Siraskar, R., Kumar, S., Patil, S., Bongale, A., & Kotecha, K. (2023). Reinforcement learning for predictive maintenance: A systematic technical review. *Artificial Intelligence Review, 56*, 12885–12947. https://doi.org/10.1007/s10462-023-10468-6
31. Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological), 34*(2), 187–202.
32. Ishwaran, H., Kogalur, U. B., Blackstone, E. H., & Lauer, M. S. (2008). Random survival forests. *The Annals of Applied Statistics, 2*(3), 841–860. https://doi.org/10.1214/08-AOAS169
33. Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). DeepSurv: Personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Medical Research Methodology, 18*(1), 1–12. https://doi.org/10.1186/s12874-018-0482-1
34. Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., & Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics, 3*(6), 422–440. https://doi.org/10.1038/s42254-021-00314-5
35. Vert, M. P. J., Sharpanskykh, A., & Curran, R. (2021). Adaptive resilience of complex safety-critical sociotechnical systems: Toward a unified conceptual framework and its formalization. *Sustainability, 13*(24), 13915. https://doi.org/10.3390/su132413915
36. Czichos, H. (Ed.). (2013). *Handbook of technical diagnostics: Fundamentals and application to structures and systems.* Springer.
37. Peralta, A., Olivas, J. A., Romero, F. P., & Navarro-Illana, P. (2025). Integration of fuzzy techniques and formal representation of domain and expert knowledge in AI systems: A comprehensive review. *Contemporary Mathematics.* https://doi.org/10.37256/cm.6220256231
38. Yu, W. (2009). *Recent advances in intelligent control systems.* Springer-Verlag London. https://doi.org/10.1007/978-1-84882-548-2
39. Dalzochio, J., et al. (2020). Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges. *Computers in Industry, 123*, 103298. https://doi.org/10.1016/j.compind.2020.103298
40. Sipos, R., Fradkin, D., Moerchen, F., & Wang, Z. (2014, August). Log-

based predictive maintenance. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1867–1876). https://doi.org/10.1145/2623330.2623340
41. Lee, J., et al. (2014). A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters, 3*, 18–23. https://doi.org/10.1016/j.mfglet.2014.12.001
42. Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal, 3*(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198
43. Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access, 6*, 52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052
44. Bansal, G., et al. (2019). Beyond accuracy: The role of mental models in human-AI team performance. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* (Vol. 7, pp. 2–11). https://doi.org/10.1609/hcomp.v7i1.5285
45. Rieg, R., Vanini, U., & Gleißner, W. (2025). *Enterprise risk management: A modern approach.* Springer, Cham. https://doi.org/10.1007/978-3-031-86425-4
46. Aven, T. (2016). Risk assessment and risk management: Review of recent advances on their foundation. *European Journal of Operational Research, 253*(1), 1–13. https://doi.org/10.1016/j.ejor.2015.12.023
47. IEC 31010:2019. *Risk management — Risk assessment techniques.*
48. Rausand, M., & Haugen, S. (2020). *Risk assessment: Theory, methods, and applications.* Wiley. https://doi.org/10.1002/9781119377351
49. Koval, V., Atstāja, D., Shmygol, N., Udovychenko, V., Hrinchenko, H., & Tsimoshynska, O. (2025). Strategic management and security risk assessment of energy systems. *Urban Sci., 9*, 48. https://doi.org/10.3390/urbansci9020048
50. Вычужанин, В. В., & Рудниченко, Н. Д. (2014). Метод управления рисками судовых сложных технических систем. *Проблеми техніки, (2)*, 138–142.
51. Ekaterinovskaya, M. A., Orusova, O. V., Tshadadze, N. V., & Haustova, K. V. (2021). *The program-oriented management in the public sector: Development tools.* Springer, Cham. https://doi.org/10.1007/978-3-030-56433-9_165
52. Power, D. J. (2008). Decision support systems: A historical overview. In *Handbook on Decision Support Systems 1*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-48713-5_7
53. Madni, A. M., Madni, C. C., & Lucero, S. D. (2019). Leveraging digital twin technology in model-based systems engineering. *Systems, 7*(1), 7. https://doi.org/10.3390/systems7010007

54. Bansal, G., et al. (2019). Beyond accuracy: The role of mental models in human-AI team performance. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* (Vol. 7, pp. 2–11).
55. Siraskar, R., Kumar, S. V. C., Patil, S., Bongale, A., & Kotecha, K. (2023). Reinforcement learning for predictive maintenance: A systematic technical review. *Artificial Intelligence Review, 56*(11), 1–63. https://doi.org/10.1007/s10462-023-10468-6
56. Mitra, S., & Acharya, T. (2003). *Data mining: Multimedia, soft computing, and bioinformatics.* Hoboken, NJ: John Wiley & Sons, Inc.
57. Vychuzhanin, V., Rudnichenko, N., Boyko, V., Shibaeva, N., & Konovalov, S. (2016). Devising a method for the estimation and prediction of technical condition of ship complex systems. *Eastern-European Journal of Enterprise Technologies, 84*(6/9), 4–11.
58. Khandpur, R. P., Tang, T., & Ji, S. Y. (2020). Towards immutable logging for industrial control systems. In *2020 IEEE Conference on Communications and Network Security (CNS)* (pp. 1–2). IEEE.
59. Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). LSTM-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint* arXiv:1607.00148. https://doi.org/10.48550/arXiv.1607.00148
60. Farshidi, S., & Jansen, S. (2021). A decision support system for pattern-driven software architecture. In *International Conference on Advanced Information Systems Engineering* (pp. 3–18). Springer, Cham.
61. Вычужанин, В. В. (2018). Распределенный программный комплекс на базе фреймворка APACHE SPARK для обработки потоковых BIG DATA от сложных технических систем. *Інформатика та математичні методи в моделюванні, 8*(2), 146–154.
62. Ríos, J., Hernández, J. C., Oliva, M., & Mas, F. (2015). Product avatar as digital counterpart of a physical individual product: Literature review and implications in an aircraft. In *Advances in Transdisciplinary Engineering* (Vol. 2, pp. 657–666). IOS Press. https://doi.org/10.3233/978-1-61499-544-9-657
63. Boschert, S., & Rosen, R. (2016). Digital twin—the simulation aspect. In *Mechatronic futures: Challenges and solutions for mechatronic systems and their designers* (pp. 59–74). Springer, Cham. https://doi.org/10.1007/978-3-319-32156-1_5
64. Mathis, C. (2017). Data lakes. *Datenbank-Spektrum, 17*(3), 289–293.
65. Dehghani, Z. (2020). *Data mesh: Delivering data-driven value at scale.* O'Reilly Media.
66. Hendler, J., & Mulvehill, A. M. (2016). *Social machines: The coming collision of artificial intelligence, social networking, and humanity.* Apress.
67. Vychuzhanin, V. V., & Rudnichenko, N. D. (2014). Assessment of risks

structurally and functionally complex technical systems. *Eastern-European Journal of Enterprise Technologies*, *1*(2), 18–22.
68. Рудниченко, Н. Д., & Вычужанин, В. В. (2013). Информационная когнитивная модель технологической взаимозависимости сложных технических систем. *Информатика и математические методы в моделировании, (3)*, 240–247.
69. Бойко, В. Д., & Вычужанин, В. В. (2012). Модель оценки живучести судовых технических систем. *Вестник Миколаївського кораблебудівного університету, (3)*, 62–67.
70. Bailey, D., & Wright, E. (2003). *Practical SCADA for industry.* New York: Elsevier.
71. Zadrozny, P., & Kodali, R. (2013). *Big data analytics using Splunk.* Berkeley, CA, USA: Apress.
72. Ohlhorst, F. J. (2013). *Big data analytics: Turning big data into big money.* Hoboken, NJ, USA: Wiley.
73. Pokorny, J. (2011). NoSQL databases: A step to database scalability in web environment. In *Proceedings of the 13th International Conference on Information Integration and Web-based Application and Services* (pp. 278–283).
74. Apache Hadoop Documentation. (2014). [Electronic resource]. Retrieved from http://hadoop.apache.org/
75. White, T. (2012). *Hadoop: The definitive guide.* O'Reilly. Retrieved from http://cdn.oreillystatic.com/oreilly/booksamplers/9781449311520_sampler.pdf
76. Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop distributed file system. In *MSST '10 Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies* (pp. 1–10).
77. Вычужанин, В. В., Шибаев, Д. С., Шибаева, Н. О., & Рудниченко, Н. Д. (2017). Оптимізація відбору та аналізу інформації в різноструктурних сховищах даних. *Інформатика та математичні методи в моделюванні, 7*(4), 318–325.
78. Bhatotia, P., Wieder, A., Rodrigues, R., Acar, U. A., & Pasquin, R. (2011). Incoop: MapReduce for incremental computations. In *Proceedings of the 2nd ACM Symposium on Cloud Computing* (pp. 1–14).
79. Apache Spark Documentation. (2014). [Electronic resource]. Retrieved from https://spark.apache.org/documentation.html
80. Apache Spark Research. (2014). [Electronic resource]. Retrieved from https://spark.apache.org/research.html
81. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D. Talwalkar, A. (2016). MLlib: Machine learning in Apache Spark. *The Journal of Machine Learning Research*, *17*(1), 1235–1241.
82. Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed

messaging system for log processing. In *Proceedings of the NetDB* (pp. 1–7).
83. Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., & Tzoumas, K. (2015). Apache Flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, *36*(4).
84. Marz, N., & Warren, J. (2015). *Big data: Principles and best practices of scalable realtime data systems.* Manning Publications Co.
85. Kleppmann, M. (2017). *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems.* O'Reilly Media.
86. Богомолов, А. М., & Твердохлебов, В. А. (1974). *Диагностика сложных систем.* Киев: Наукова думка.
87. Redelinghuys, A. J. H., Basson, A. H., & Kruger, K. (2020). A six-layer architecture for the digital twin: A manufacturing case study implementation. *Journal of Intelligent Manufacturing*, *31*, 1383–1402. https://doi.org/10.1007/s10845-019-01516-6
88. Botín-Sanabria, D. M., et al. (2022). Digital twin technology: A review of the current state and future perspectives. *Applied Sciences*, *12*(17), 8561. https://doi.org/10.3390/rs14061335
89. Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, *6*, 52138–52160.
90. Dragoni, N., et al. (2017). Microservices: Yesterday, today, and tomorrow. In *Present and Ulterior Software Engineering* (pp. 195–216). Springer, Cham. https://doi.org/10.1007/978-3-319-67425-4_12
91. Zimmermann, O. (2017). Microservices tenets: Agile approach to service development and deployment. *Computer Science – Research and Development*, *32*(3–4), 301–310.
92. Dalzochio, J., et al. (2020). Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges. *Computers in Industry*, *123*, 103298. https://doi.org/10.1016/j.compind.2020.103298
93. Pimentel, J. F., et al. (2021). A framework for the design and evaluation of prescriptive maintenance systems. *Journal of Manufacturing Systems*, *60*, 87–98.
94. Kritzinger, W., et al. (2018). Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, *51*(11), 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474
95. Straka, O., & Punčochář, I. (2022). Distributed design for active fault diagnosis. *International Journal of Systems Science*, *53*(3), 562–574. https://doi.org/10.1080/00207721.2021.1963501
96. Travé-Massuyès, L., & Wotawa, F. (2024). Bridging hardware and

software diagnosis: Leveraging fault signature matrix and spectrum-based fault localization similarities. In *Proceedings of the 35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024)* (pp. 5:1–5:15). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/OASIcs.DX.2024.5
97. Boychuk, I. P., Grinek, A. V., Martyushev, N. V., Klyuev, R. V., Malozyomov, B. V., Tynchenko, V. S., Kukartsev, V. A., Tynchenko, Y. A., & Kondratiev, S. I. (2023). A methodological approach to the simulation of a ship's electric power system. *Energies, 16*, 8101. https://doi.org/10.3390/en16248101
98. Shannon, R. E. (1975). *Systems simulation: The art and science*. Prentice-Hall.
99. Beetz, M., Buss, M., & Wollherr, D. (2007). Cognitive technical systems—What is the role of artificial intelligence. In *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI* (pp. 19–42).
100. Botín-Sanabria, D. M., et al. (2022). Digital twin technology: A review of the current state and future perspectives. *Applied Sciences, 12*(17), 8561.
101. Shao, G., & Helu, M. (2020). Framework for a digital twin in manufacturing: Scope and requirements. *Manufacturing Letters, 24*, 105–107.
102. Vychuzhanin, V., Rudnichenko, N., Shybaiev, D., & Gritsuk, I. et al. (2018). Cognitive model of the internal combustion engine. *SAE Technical Paper 2018-01-1738*. https://doi.org/10.4271/2018-01-1738
103. Рудниченко, Н. Д., Вычужанин, В. В., & Шибаев, Д. С. (2017). Применение кластерного анализа данных для выделения меры схожести факторов влияния на работоспособность сложных технических систем. *Информатика и математические методы в моделировании, (3)*, 214–219.
104. Шибаев, Д. С., Вычужанин, В. В., Шибаева, Н. О., & Рудниченко, Н. Д. (2018). Оптимизация методов прогнозирования, обработки и анализа информации в разноструктурных хранилищах данных. *Информатика и математические методы в моделировании, (1)*, 78–85.
105. Vychuzhanin, V. V., Shibaev, D. S., Boyko, V. D., Shibaeva, N. O., & Rudnichenko, N. D. (2017). Big data mapping in the geopositioning systems for fishing industry. *International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, 28–31.
106. Pop, F., & Neagu, G. (Eds.). (2021). *Big data platforms and applications: Case studies, methods, techniques, and performance evaluation.* Springer.
107. Li, K.-C., Jiang, H., & Zomaya, A. Y. (2017). *Big data management and processing.* CRC Press.
108. *Data analytics and machine learning: Navigating the big data*

landscape. (2024). Springer.
109. Luo, B., et al. (2023). SQL and NoSQL database software architecture performance analysis and assessments — A systematic literature review.
110. Shah, K., Shah, N., Sawant, V., & Parolia, N. (Eds.). (2023). *Practical data mining techniques and applications.* Auerbach.
111. *A review of intelligent data analysis: Machine learning approaches for addressing class imbalance in healthcare — Challenges and perspectives.* (2024). https://doi.org/10.1177/1088467X241305509
112. Greenwell, B. M. (Tree-Based Methods for Statistical Learning in R).
113. *Artificial intelligence-driven vehicle fault diagnosis to revolutionize automotive maintenance: A review.* (2024). *Computer Modeling in Engineering & Sciences.* TechScience Press.
114. Abdi, A., Ahmadinezhad, Z., & Habibi, M. (2025). A hybrid DEA and decision tree framework for classifying and ranking commercial bank branches. *International Journal of Operations Research and Artificial Intelligence*, *1*(2), 74–81. https://doi.org/10.48314/ijorai.v1i2.63
115. Wen, Z., Li, Q., He, B., & Cui, B. (2021). Challenges and opportunities of building fast GBDT systems. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21), Survey Track* (pp. 4661–4666). IJCAI Organization.
116. Vychuzhanin, V., Rudnichenko, N., Boyko, V., Shibaeva, N., & Konovalov, S. (2016). Devising a method for the estimation and prediction of technical condition of ship complex systems. *Eastern-European Journal of Enterprise Technologies, 6*(9 (84)), 4–11.*
117. Li, X., Yang, P., Gu, Y., Zhan, X., Wang, T., Xu, M., & Xu, C. (2024). Deep active learning with noise stability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, *38*(12), 13655–13663. https://doi.org/10.1609/aaai.v38i12.29270
118. Mafia, M. M. P., Ayoub, N., Trumpler, L., & de Oliveira Hansen, J. P. (2024). A digital twin design for conveyor belts predictive maintenance. In O. Niggemann, J. Beyerer, M. Krantz, & C. Kühnert (Eds.), *Machine learning for cyber-physical systems: Selected papers from the International Conference ML4CPS 2023* (Vol. 18, Technologien für die intelligente Automation, pp. 111–119). Springer, Cham. https://doi.org/10.1007/978-3-031-47062-2_11
119. Вычужанин, В. В. (2010). Диагностика, контроль при эксплуатации и ремонте систем кондиционирования воздуха на основе гибридных нейро-нечетких экспертных систем. *Вісник ОНМУ, збірник наукових праць*, (30), 100–109.
120. Bechard, S., et al. (2021). Opportunities and challenges of applying cognitive process dimensions to map-based learning and alternate assessment. *Frontiers in Education, 6.* https://doi.org/10.3389/

feduc.2021.653693
121. Davidich, N., Galkin, A., Davidich, Y., Schlosser, T., Capayova, S., Nowakowska-Grunt, J., Kush, Y., & Thompson, R. (2022). Intelligent decision support system for modeling transport and passenger flows in human-centric urban transport systems. *Energies, 15*, 2495. https://doi.org/10.3390/en15072495
122. Rahman, A., Myo Aung, K., Ihsan, S., Raja Ahsan Shah, R. M., Al Qubeissi, M., & Aljarrah, M. T. (2023). Solar energy dependent supercapacitor system with ANFIS controller for auxiliary load of electric vehicles. *Energies, 16*, 2690. https://doi.org/10.3390/en16072690
123. Papatheodorou, A., Papadimitriou, N., Stathatos, E., Benardos, P., & Vosniakos, G.-C. (2025). Recent advances in sensor fusion monitoring and control strategies in laser powder bed fusion: A review. *Machines, 13*, 820. https://doi.org/10.3390/machines13090820
124. Elahi, M., Afolaranmi, S., Lastra, J. L. M., & García, J. A. P. (2023). A comprehensive literature review of the applications of AI techniques through the lifecycle of industrial equipment. *Discover Artificial Intelligence, 3*(43). https://doi.org/10.1007/s44163-023-00089-x
125. Zhou, Q., Wang, C., Sun, Z., Li, J., Williams, H., & Xu, H. (2021). Human-knowledge-augmented Gaussian process regression for state-of-health prediction of lithium-ion batteries with charging curves. *Journal of Electrochemical Energy Conversion and Storage, 18*(3), 030907. https://doi.org/10.1115/1.405079
126. Abdelkareem, M. A., Alshathri, S. I., Masdar, M. S., & Olabi, A. G. (2023). Adaptive neuro-fuzzy inference system modeling and optimization of microbial fuel cells for wastewater treatment. *Water, 15*, 3564. https://doi.org/10.3390/w15203564
127. Cervantes, J., Yu, W., Salazar, S., & Chairez, I. (2017). Takagi–Sugeno dynamic neuro-fuzzy controller of uncertain nonlinear systems. *IEEE Transactions on Fuzzy Systems, 25*(6), 1601–1615. https://doi.org/10.1109/TFUZZ.2016.2612697
128. Yang, Y., Tu, F., Huang, S., Tu, Y., & Liu, T. (2023). Research on CNN–LSTM DC power system fault diagnosis and differential protection strategy based on reinforcement learning. *Frontiers in Energy Research, 11*, Article 1258549. https://doi.org/10.3389/fenrg.2023.1258549
129. Giesecke, T., Mozaffar, M., & El-Gharably, A. (2021). Quantification of model uncertainty in physics-informed machine learning and its application in civil engineering. *Computer-Aided Civil and Infrastructure Engineering, 36*(8), 1083–1099.
130. Jardine, A. K. S., & Tsang, A. H. C. (2013). *Maintenance, replacement, and reliability: Theory and applications* (2nd ed.). Boca Raton, FL: CRC Press (Taylor & Francis Group). ISBN 978-1466554856.

131. British Standards Institution. (2017). *BS EN 60300-3-3:2017. Dependability management. Application guide. Life cycle costing*. London: BSI. ISBN 978-0-580-91628-1.
132. Sullivan, W. G., Wicks, E. M., & Koelling, C. P. (2019). *Engineering economy* (17th ed.). New York, NY: Pearson. ISBN 978-0-13-487006-9.
133. Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)* (pp. 4768–4777).
134. Hastie, T., Tibshirani, R., & Friedman, J. (2004). *The Mathematical Intelligencer*, *27*(2), 83–85. https://doi.org/10.1007/BF02985802
135. Ioannou, P. A., & Sun, J. (2010). *Robust adaptive control*. https://doi.org/10.1201/b10384-41
136. Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 4768–4777. https://doi.org/10.48550/arXiv.1705.07874
137. National Institute of Standards and Technology. (2012). *Special Publication 800-30 Revision 1: Guide for conducting risk assessments*. NIST.
138. International Organization for Standardization. (2018). *ISO 31000:2018 — Risk management: Guidelines*. ISO.
139. American National Standards Institute. (2020). *ANSI/ISA-62443-3-2-2020: Security for industrial automation and control systems – Part 3-2: Security risk assessment for system design*. ANSI.
140. DNV GL AS. (2023). *Recommended practice DNVGL-RP-0496: Cyber security resilience management for ships and mobile offshore units*. DNV GL.
141. Liu, J.-H., Corbita, N. T., Jr., Lee, R.-M., & Wang, C.-C. (2022). Wind turbine anomaly detection using Mahalanobis distance and SCADA alarm data. *Applied Sciences, 12*, 8661. https://doi.org/10.3390/app12178661
142. Abshari, D., & Sridhar, M. (2025). A survey of anomaly detection in cyber-physical systems. *arXiv preprint* arXiv:2502.13256. https://doi.org/10.48550/arXiv.2502.13256
143. Vychuzhanin, V., & Vychuzhanin, A. (2025). Adequacy and verification of an intelligent diagnostic model for ship power plants. *Informatics and Mathematical Methods in Simulation, 15*(3), 312–326. https://doi.org/10.15276/imms.v15.no3.312
144. Vychuzhanin, V., & Vychuzhanin, A. (2025). Integrated modeling of reliability and maintenance of ship's power plant equipment considering degradation and operational conditions. *Системи контролю інформації та інтелектуальні технології. Досягнення та застосування:*

монографія (pp. 335–363). Lviv–Toruń: Liha-Pres. https://doi.org/10.36059/978-966-397-538-2-18
145.Vychuzhanin, V., & Vychuzhanin, A. (2025). Reconfigurable similarity metrics for interpretable fault diagnostics in complex ship power systems. *Інформаційні управляючі системи і технології (ІУСТ-ОДЕСА-2025): матеріали XIII Міжнародної науково-практичної конференції* (pp. 35–38). https://doi.org/10.36059/978-966-397-531-3
146. Vychuzhanin, V., & Vychuzhanin, A. (2025). Adaptive similarity assessment metric for intelligent failure diagnostics in ship power plants. *Bulletin of Cherkasy State Technological University, 30*(3), 80–92. https://doi.org/10.62660/bcstu/3.2025.80
147. Vychuzhanin, V., & Vychuzhanin, A. (2025). Development of a technical condition assessment algorithm for complex systems based on probabilistic failure estimation. *Information Technologies and Computer Engineering, 22*(2), 9–19. https://doi.org/10.63341/vitce/2.2025.09
148. Vychuzhanin, V., & Vychuzhanin, A. (2025). Model for determining similarity between cases in failure diagnosis of ship power plant equipment. *Вісник Приазовського державного технічного університету. Серія: Технічні науки, 50*, 9–17. https://doi.org/10.31498/2225-6733.50. 2025. 336233
149. Vychuzhanin, V., & Vychuzhanin, A. (2025). Integrated approach to diagnosing complex technical systems: Experimental validation and multidimensional efficiency assessment. *Вісник Східноукраїнського національного університету імені Володимира Даля, 5*(291), 5–17. https://doi.org/10.33216/1998-7927-2025-291-5-5-17
150. Vychuzhanin, V., & Vychuzhanin, A. (2025). Integrated approach to creating a case-based database for diagnosing failures in ship power plants. *Informatics and Mathematical Methods in Simulation, 15*(2), 155–165. https://doi.org/10.15276/imms.v15.no2.155
151.Vychuzhanin, V., & Vychuzhanin, A. (2025). Adaptive case-based reasoning with probabilistic integration for the diagnosis and prognosis of the technical condition of complex systems. *Вісник Східноукраїнського національного університету імені Володимира Даля, 4*(290), 5–20. https://doi.org/10.33216/1998-7927-2025-290-4-5-20
152.Vychuzhanin, V., & Vychuzhanin, A. (2025). Using ChatGPT for the intelligent diagnostics of complex technical systems. *Bulletin of Cherkasy State Technological University, 30*(1), 68–79. https://doi.org/10.62660/bcstu/1.2025.68

FOR NOTES

Наукове видання

ВИЧУЖАНІН Володимир
ВИЧУЖАНІН Алексій

Цифрові методи та моделі для керування та живучості складних технічних систем

Монографія

Англійською мовою

Підписано до друку 13.11.2025. Формат 60×84/16.
Папір офсетний. Гарнітура Times. Цифровий друк.
Умовно-друк. арк. 21,27. Тираж 300.
Замовлення № 1225-106. Ціна договірна.
Віддруковано з готового оригінал-макета.

Українсько-польське наукове видавництво «Liha-Pres»
79000, м. Львів, вул. Технічна, 1
87-100, м. Торунь, вул. Лубіцка, 44
Телефон: +38 (050) 658 08 23
E-mail: editor@liha-pres.eu
Свідоцтво суб'єкта видавничої справи
ДК № 6423 від 04.10.2018 р.