mathematics

MDPI

Article

Adaptive Fuzzy Predictive Approach in Control

Anton A. Romanov *, Aleksey A. Filippov and Nadezhda G. Yarushkina

Department of Information Systems, Ulyanovsk State Technical University, 32 Severny Venetz Street, 432027 Ulyanovsk, Russia
* Correspondence: romanov73@gmail.com

Abstract: This article studies the approach to solving the problem of controlling the complex organizational and technical systems based on hybrid models. We propose a new component of intelligent decision support that is integrated with control systems. The proposed component is based on fuzzy logic and knowledge engineering. We present a model of ontology to form the context of data analysis and time series modeling. The ontological context allows us to represent trends of the analyzed object indicators. An expert can add a set of fuzzy rules to the ontology for systems control based on the fuzzy inference. The proposed approach allows reducing the time of analysis and interpretation of the results. Experimental results confirm the correctness and effectiveness of the approach proposed in this article.

Keywords: time series; context; tendencies; predictive analytic; ontology

MSC: 90B50

1. Introduction

The management of complex technical systems requires the analysis of a large volume of data. The precision of the decisions depends on various factors, such as the control object complexity, the volume of data for analysis, the control level, and the urgency of decision making. The described factors require choosing a suitable class of analytical models for decision support systems. We need to define properties of the analyzed objects to determine the acceptable approaches and methods for solving control problems.

The proposed approach is based on the following principles:

1. The principle of the control object decomposition: The management of a complex object is based on analysis of a large volume and the various types of data. It is necessary to ensure management models have sustainable functioning when choosing and adapting them. One solution is the formulation of restrictions on input data and its types and on operation modes of management models. First, this task must be solved successfully at the lowest level. Then, decision makers can build models for higher levels.
2. The principle of dynamic indicators: We must analyze the attributes of objects over the time (in dynamic states) to track tendencies in their values for successful analysis. Such an analysis allows us to consider a greater number of dependencies.
3. The principle of abstraction: It is necessary to conceptualize abstractions from the features of the analyzed objects in the modeling process.
4. The principle adaptability: The context of analyzed objects allows us to adapt models to features and restrictions of the current problem area.

These principles are based on the general theory of control [1] and also require the creation of new classes of models. We propose to use context modeling for solving forecasting problems in complex organizational and technical systems control. The novelty of the proposed approach is the creation of a new intelligent decision support component based on fuzzy logic and knowledge engineering for building hybrid control systems.

Citation: Romanov, A.A.; Filippov, A.A.; Yarushkina, N.G. Adaptive Fuzzy Predictive Approach in Control. *Mathematics* **2023**, *11*, 875. https://doi.org/10.3390/math11040875

Academic Editor: Hongyu Liu

Received: 30 December 2022
Revised: 2 February 2023
Accepted: 7 February 2023
Published: 9 February 2023

2. Related Works

As you can see in Figure 1, the scheme of the control process contains the following vectors:

- The input of the control object is the vector $X = \{x_1, \ldots, x_n\}$.
- The control object is also regulated by the vector $W = \{w_1, \ldots, w_r\}$. The values of the vector W cannot be determined.
- The output of the control object is the vector $V = \{v_1, \ldots, v_n\}$.
- The feedback vector from the control system $U = \{u_1, \ldots, u_k\}$ also comes to the input of the control object.

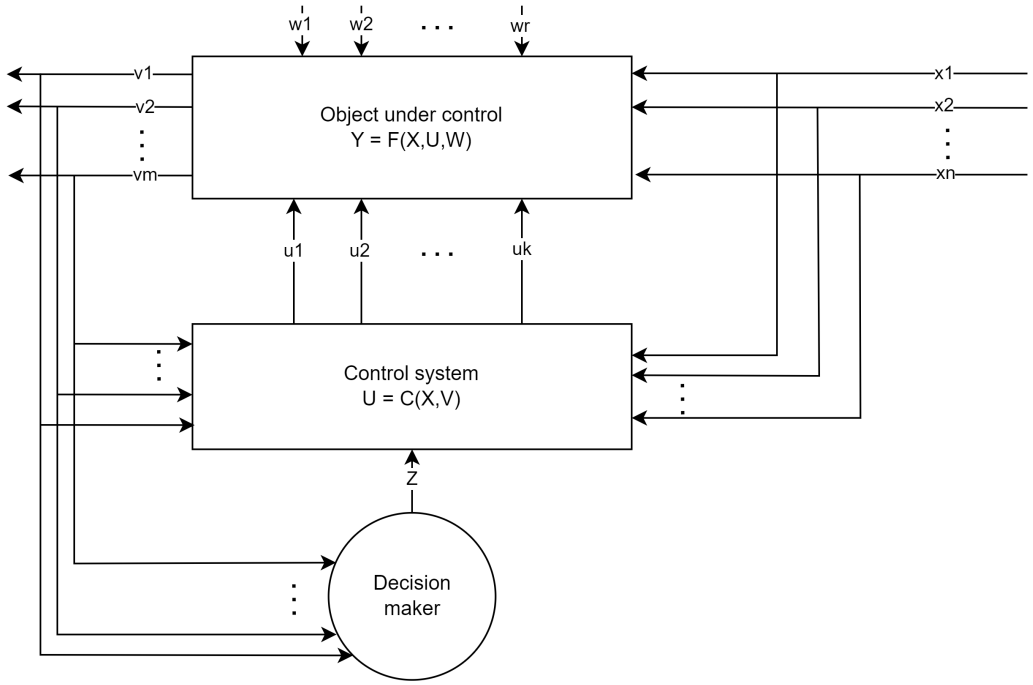

Figure 1. Scheme of the control process.

There are various approaches to control complex technical systems that provide control stability. For example, deterministic models as mathematical dependencies can be used for this task. Intelligent components and components based on machine learning can be used for systems with non-linearity and/or uncertainty in the behavior of a control object [2]. The use of forecasting methods allows us to reduce the response time of the system to emerging deviations. Predictive models have the disadvantage of needing to choose the appropriate volume of data to build a forecast and evaluate the quality of this data.

Neural networks can be a solution for a control task. A neural network is a black box model that can accurately estimate complex non-linear relationships. Fuzzy modeling also can be used for solving the control task. Fuzzy modeling is a modeling method that uses a fuzzy inference system based on fuzzy rules [3]. Hybrid solutions that overcome the limitations of existing individual models allow us to obtain new results from both. One such model is ANFIS. However, the development of methods and algorithms based on artificial intelligence requires several experiments to produce the required configuration and accuracy of the models. In addition, if the data contain incompleteness, noise, or gaps, the results of the work of intelligent models will be of poor quality [4,5].

There are several control systems types. The first class includes automatic control systems. The role of the decision maker is minimized in these systems [6,7]. Automatic control systems are often based on PID controllers with various types of models. Another class of systems is the decision support system. This class of systems is actively used by decision makers in various management tasks. A hierarchical approach to the decomposition of data analysis and decision-making tasks is possible in decision support systems [8].

The directions for the development of intellectual methods are [9]:

- The expansion of the set of analyzed data;
- The complication of the applied models;
- The increase in the interpretability of the results;
- The explainability of the result.

Authors in the article [10] noted that the use of additional information can improve the quality of modeling.

Researchers often use time series models to identify the patterns of changes in indicators. Time series models are based on methods of trend analysis and various methods for correlations. Correlation methods allow us to identify relationships between indicators of a single object or between indicators of different objects. These methods are based on an analysis of object attributes in dynamics. Statistical, fuzzy, and neural network models of time series allow solving these tasks, depending on the type of data [11].

3. The Problem Statement

In our study, we use a hybrid approach. The novelty of the proposed approach is the use of contextual information about the analysed object, such as dependencies, restrictions, and relationships. We use a priori information specified in the context of objects of the organizational and technical system instead of searching for dependencies through the analysis of time series.

It is necessary to change the scheme of the common control process (Figure 1) to apply the proposed approach. The new block allows for modeling dynamic parameters and using context data. The new block is not part of the control system because it provides information to the decision maker and/or to the automatic control system. This solution is not a decision support system in the classic sense, but it allows increased flexibility of the control process. We plan to continue improving the contextual analysis of the dynamics of object attributes in the future. We also plan to apply the approach to unmanned vehicles in agriculture for automatic planning and tracking the movement trajectory. The modified scheme is shown in Figure 2.

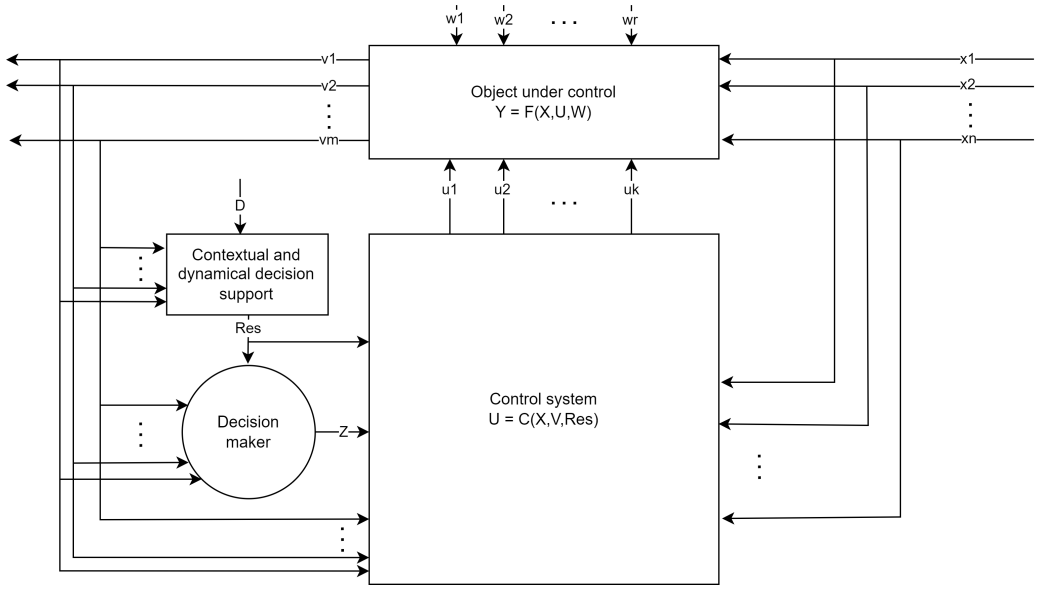

Figure 2. Modified scheme of the control process.

Let us consider the new block in more detail. The contextual and dynamic decision support block has the following properties:

- Input to obtain external information about the control context D of a control object;
- Input to accumulate the dynamics of a control object attributes (vector V);
- Management recommendations Res as the output (recommendations for decision maker and/or numerical values to regulate the behavior of a control object).

Software development based on agile methodologies is the example of a problem area used for the proposed approach demonstration. Timely response to emerging events is especially important in this domain because it allows implementing aspects such as inspection and adaptation in conditions of extreme development parameters [12].

The indicators of quality in a model software system include functionality (measured by the number of implemented tasks with new functionality), performance (measured by the time it takes to execute functions), and reliability (measured by the number of unhandled exceptions) [13].

The indicators of a quality model during the development process include mainly internal project quality indicators: efficiency (measured by the test execution time, project build time, the ratio of the number of implemented functionality to the number of developers), reliability (measured by the number of errors during static testing of the code, test coverage), and fulfillment of requirements (measured by the duration of the development tasks).

Note that this selection of parameters does not pretend to be original. These parameters are just used to demonstrate the proposed approach in the project management task. The main aim is to reduce the time required for obtaining information about the critical levels of quality models.

4. Control Models

Let us present the necessary models for the block of contextual analysis of the dynamics of indicators of the proposed scheme as shown in Figure 2. A significant difference from the traditional control scheme is the availability of information about related objects.

Building a model for managing a complex technical system must begin with the selection of a set of analyzed objects and their properties.

Let $O = \{o_1, o_2, \ldots, o_n\}$ be a set of such objects, where $n \in \mathbb{N}$. For each object, we define a set of attributes that characterizes the state of the object: $o_j = \{v_i^{o_j}\}$, where $v_i^{o_j}$ is the numerical value of the indicator i for the analyzed object o_j. In the scheme in Figure 2, such a set of attributes for a control object are represented by the vector $V = \{v_1, \ldots, v_m\}$. These indicators are the basis for the time series of indicators. They help implement the principle of dynamics of indicators, see (1)

$$TS(v_i^{o_j}, t) \rightarrow ts_i^{o_j}, \tag{1}$$

where TS is the function that creates the time series of the indicator i of the object o_j, t is the time of the fixation the value of the indicator, and $ts_i^{o_j}$ is the time series of the indicator i object o_j.

The relations between the objects of the problem area must be considered in order to analyze the mutual influence of their states and track dependencies.

Take the case of $R_{kl} = r_{o_k o_l}$, where $r_{o_k o_l}$ characterizes the object connection degree between o_k and object o_l. The relation degree is a numerical characteristic involved in the analysis of the dynamics of indicators. It shows how the selected indicator positively or negatively affects the state of the associated object. When solving the control problem, such a relation helps to achieve the principle of decomposition. The limited set of related objects makes it possible to determine the criteria for achieving a stable state of the control system based on the proposed models. Instead of the relationship degree of one object to another, we can use the object attributes relationship degree for more flexibility.

Calculating the dependency degree of the attributes of one object on the parameters of another is expressed as a function, see (2).

$$f(R_{kl}) = \sum ts_{t+1}^{o_k} * r_{o_k o_l} \tag{2}$$

Such functionality allows us to calculate the state or adjust the control process in the following scenarios:

1. The state and reliable trends of changes in the values of attributes of one object are known. It is possible to predict the value of the attribute associated with the source

object by the presented functional dependence: $v_i'^{o_j} = v_i'^{o_k} * r_{o_k o_l}$, where $v_i'^{o_j}$ is the predicted value of i-th attribute j-th object, or in the form of a time series forecast: $ts_{t+1}^{o_j} = ts_{t+1}^{o_k} * r_{o_k o_l}$

2. We do not know the trends in the values of related objects, but there are predictive values. Using these values, it is possible to diagnose the values of the attributes $|v_i'^{o_j} - v_i^{o_k}|$.

Control actions on the control object are defined by the control task after analyzing the deviations of the output indicators of the control object from the reference ones, see (3).

$$|v_i^{o_j} - e^{o_j}| \to 0, \forall i, j, \tag{3}$$

where $v_i^{o_j}$ is the value of the i-th attribute of the j-th object, and $e_i^{o_j}$ is the reference value.

An exact match of the values is not achieved, and it is required to set some intervals boundaries of acceptable values. We can express a vector for each object, see Equation (4).

$$\begin{gathered} \Delta^{o_j} = \{\delta_i^{o_j}\} \\ |v_i^{o_j} - e^{o_j}| \in \delta_i^{o_j}, \forall i, j. \end{gathered} \tag{4}$$

To implement the principle of adaptability, we specify intervals of acceptable values as fuzzy labels $\tilde{\Delta}^{o_j}$.

As a result, we denote the function implemented by the control model of the function (5).

$$\begin{gathered} D =< O, \tilde{\Delta}^O, R >, \\ Control(D, TSModels) \to Res, \end{gathered} \tag{5}$$

where D is an ontology that describes the links of the objects of analysis O by setting the weight of the link R and includes restrictions on the values of the attributes of the control object $\tilde{\Delta}^O$ based on the ontology and a set of models, time series $TSModels$, recommended for managing Res as consequential rules for responding to deviations.

5. Analysis Algorithm

According to the scheme shown in the Figure 2, it is necessary to determine the order of operation of the block for contextual analysis of the dynamics of indicators. The algorithm given below concerns only the operation of this block and is not a description of the entire control task. The complex of models in solving the control problem is a set of prognostic models of time series $TSModels$, a knowledge base model for the context of modeling time series D, and a control model based on fuzzy logical inference during the operation of the system of rules $Control(D, TSModels) \to Res$.

Let us describe the algorithm used for the decision-making procedure in solving control problems.

1. Form an ontology D that describes the relationships between objects and attributes.
2. Generate time series of values of indicators of the studied objects $v_i^{o_j}$. Instead of a single output vector of the control object, it is necessary to present the history of each indicator $TS(v_i^{o_j}, t) \to ts_i^{o_j}$.
3. Describe the context in the ontology D as rules for performing fuzzy inference when working with predictive $TSModels$, their parameters, and types.
4. Select and train fuzzy predictive $TSModels$ to adapt to the features of the selected objects. The results are presented in more detail in [14].
5. Make a forecast based on incoming data and context.
6. Evaluate the result on test data $|v_i^{o_j} - e^{o_j}|$.
7. If the quality is satisfactory, make a forecast for the next management period and adjust the management parameters Res based on the forecasts; otherwise, choose other predictive models.

The comparison procedure with reference values during testing works on fuzzy values and rules defined in the ontology. This is not able to set the exact values of the error thresholds, but it can be used as fuzzy labels $\tilde{\Delta}^{o_j}$.

6. Quality Assessment

Evaluation of the quality of the models is a multi-level process. There are some metrics that include a metric for assessing the quality of time series modeling and a metric for assessing the quality of a management model.

6.1. Time Series Quality Assessment

One of the metrics for assessing the quality of time series forecasting is the SMAPE [15] criterion. The expression (6) represents the calculations of the metric

$$SMAPE = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}. \quad (6)$$

where F_t is t-th forecasted value, A_t is t-th real value, and n is the forecast horizon.

The value of the forecast error by the SMAPE criterion affects the choice of the time series model when the algorithm is running.

6.2. Total Quality Assessment

The quality coefficient (the lower the better) of the management model is to measure the discrepancy between the output of the model and the referenced value. Let us express this as functional (7).

$$Q_{total} = \sum_{i,j} \frac{|v_i'^{o_j} - e_i^{o_j}|}{\delta_i^{o_j}} \rightarrow min, \quad (7)$$

where $v_i'^{o_j}$ is the predicted value of the i-th attribute of the j-th object when testing the model, $e_i^{o_j}$ is the reference value when testing the model, and $\delta_i^{o_j}$ is the maximum deviation from the reference value of the i-th attribute of the j-th object.

The error rate and the difference $|v_i'^{o_j} - e_i^{o_j}|$ are used to make recommendations for management.

7. Ontology for Representing the Context of a Control Problem

The solution to the control problem involves considering the features of both the control object itself and the domain [16,17]. In addition, the control object functions under the limitations of the restrictions of some domain. We define these features and limitations as the context of the control problem (context).

We use ontologies to describe the context of a control problem.

7.1. Description Logic

Ontologies are an effective instrument of describing the context for solving various automation problems [18]. Ontologies combine tools for describing the features of a domain, components for the formation and execution of a set of production rules, and a set of inference engines. The process of inference forms new knowledge based on ontology constraints and/or a set of production rules in the SWRL language [19]. In addition, the inference engine controls the consistency of ontology axioms.

Ontologies are based on various description logic languages. Description logic makes it possible to guarantee the decidability of ontology axioms at low computational costs. Currently, the OWL 2 standard is used to represent ontologies [20–22].

Table 1 contains description logic operators and axioms for representing the elements of the proposed ontology.

Table 1. DL operators and axioms.

Description	DL	OWL
top (a special class with every individual as an instance)	$\top$	owl:Thing
bottom (an empty class)	$\bot$	owl:Nothing
class inclusion axiom	$A \sqsubseteq B$	A owl:SubClassOf B
disjoint classes axiom	$A \sqcap B \sqsubseteq \bot$	[A, B] owl:DisjointClasses
equivalence classes axiom (or defining classes with necessary and sufficient conditions)	$A \equiv B$	[A, B] owl:equivalentClasses
intersection or conjunction of classes	$A \sqcap B$	A and B
universal restriction axiom	$\forall R.A$	R only A
existential restriction axiom	$\exists R.A$	R some A
cardinality restrictions axiom	$\leq nR.A$	R exactly n A
concept assertion axiom (a is an instance of class A)	$a: A$	a: A
role assertion axiom	$(a, b): R$	a R b

7.2. Model of the Proposed Ontology

We propose an OWL 2 ontology of the following structure to describe the context of a control problem:

$$D = \langle O^D, F^D, E^D, R^D \rangle, \tag{8}$$

where
O^D is an ontological representation of objects of analysis O;
F^D is an ontological representation of a set of modeling methods of time series;
E^D is an ontological representation of expert knowledge to implement management functions;
R^D is a set of relationships between ontology components.

We present the ontology D of the context of the control problem using a dialect of $\mathcal{ALCHONF}(\mathcal{D})$ description logic:

$$D = \langle D_{TBox}, O^D_{TBox}, O^D_{ABox}, F^D_{TBox}, F^D_{ABox}, E^D_{TBox}, E^D_{ABox} \rangle,$$

where
$TBox$ is a set of terminological axioms;
$ABox$ is a set of axioms.

7.3. Terminology of the Proposed Ontology (TBox)

The common terminology of the D ontology is:

$$\top \sqsubseteq \exists hasName.String \sqcap \forall hasName.String \sqcap = 1 hasName.String,$$

where *hasName* is a functional role common to all ontology classes.

The terminology for representing analysis objects (O^D_{TBox}) can be described as the following axioms:

- The terminology O^D_{TBox} contains a set of classes for describing:
  - Analyzed objects—*Object*;
  - Properties (attributes) of an object—*Attribute*;
  - The time series of values of an object attribute—*TS*;
  - Dependencies between attributes of objects—*Dependency*;
  - Types of dependencies: positive, negative—*DependencyType*:

$$Object \sqsubseteq \top \quad Attribute \sqsubseteq \top \quad TS \sqsubseteq \top$$
$$Dependency \sqsubseteq \top \quad DependencyType \equiv \{positive, negative\};$$

- The classes *Object*, *Attribute*, *TS*, *Dependency*, and *DependencyType* are declared as disjoint:

$$Object \sqcap Attribute \sqcap TS \sqcap Dependency \sqcap DependencyType \sqsubseteq \bot;$$

- The *Object* class has the *hasAttribute* role to specify a tie between some object and a set of its attributes:

$$Object \sqsubseteq \forall hasAttribute.Attribute;$$

- The *Attribute* class has:
  - Functional roles *hasDeltaMin* and *hasDeltaMax* to define the minimum and maximum value of an attribute;
  - The *hasDependency* role to specify a tie between attributes of analyzed objects;
  - The functional role *hasTS* to define a tie between an attribute and time series that contains a dynamic of an attribute:

$$\begin{aligned} Attribute \sqsubseteq\ & \forall hasDeltaMin.Double \sqcap = 1 hasDeltaMin.Double \sqcap \\ & \sqcap \forall hasDeltaMax.Double \sqcap = 1 hasDeltaMax.Double \sqcap \\ & \sqcap \forall hasDependency.Dependency \\ & \sqcap \exists hasTS.TS \sqcap \forall hasTS.TS \sqcap = 1 hasTS.TS; \end{aligned}$$

- The *TS* class has the *hasData* functional role to define a source of time series data:

$$TS \sqsubseteq \exists hasData.String \sqcap \forall hasData.String \sqcap = 1 hasData.String;$$

- The *Dependency* class has:
  - The functional role *dependsOn* to organize dependencies between object attributes;
  - The functional role *hasType* to define the dependency between attributes;
  - The functional role *hasWeight* to determine a weight of dependency between attributes:

$$\begin{aligned} Dependency \sqsubseteq\ & \exists dependsOn.Attribute \sqcap \forall dependsOn.Attribute \sqcap \\ & \sqcap = 1 dependsOn.Attribute \sqcap \\ & \sqcap \exists hasType.DependencyType \sqcap \forall hasType.DependencyType \sqcap \\ & \sqcap = 1 hasType.DependencyType \sqcap \\ & \sqcap \exists hasWeight.Double \sqcap \forall hasWeight.Double \sqcap = 1 hasWeight.Double. \end{aligned}$$

The terminology of representation of modeling methods of time series (F^D_{TBox}) contains the following axioms:

- The terminology F^D_{TBox} contains a set of classes for describing:
  - Methods for time series modeling—*Method*;
  - The features of time series that a method supports: fuzziness, periodicity, smoothing, trends—*MethodFeature*:

$$\begin{aligned} & Method \sqsubseteq \top \\ & MethodFeature \equiv \{fuzzy, periodic, smooth, tendency\}; \end{aligned}$$

- The classes *Method* and *MethodFeature* are declared as disjoint:

$$Method \sqcap MethodFeature \sqsubseteq \bot;$$

- The *Method* class has:
  - The *hasFeature* role to define a tie between the method and its features;
  - Functional roles *hasLengthMin* and *hasLengthMax* to determine the minimum and maximum lengths of time series that the method supports:

$$Method \sqsubseteq \exists hasFeature.MethodFeature \sqcap \forall hasFeature.MethodFeature \sqcap$$
$$\sqcap >= 1 hasFeature.MethodFeature \sqcap$$
$$\sqcap \forall hasLengthMin.Integer \sqcap = 1 hasLengthMin.Integer \sqcap$$
$$\sqcap \forall hasLengthMax.Integer \sqcap = 1 hasLengthMax.Integer;$$

- In addition, the additional *needFeature* role is defined for the TS class in the terminology O^D_{TBox}. The *needFeature* role allows declaring a tie between time series and its features:

$$TS \sqsubseteq \exists needFeature.MethodFeature \sqcap \forall needFeature.MethodFeature \sqcap$$
$$\sqcap >= 1 needFeature.MethodFeature.$$

The terminology for representing expert knowledge (E^D_{TBox}) can be described as the following axioms:

- The terminology E^D_{TBox} contains a set of classes for describing:
  - Expert knowledge—$Expert$;
  - A value of some attribute of the object at a certain point of a time series—*Point*;
  - States in which an object is located—*State*. A set of states of an object is based on a value of some object attributes. The *State* class extends the $Expert$ class;
  - Actions to correct a value of some object attribute—*Result*. The *Result* class extends the $Expert$ class:

$$Expert \sqsubseteq \top \quad Point \sqsubseteq \top$$
$$State \sqsubseteq Expert \quad Result \sqsubseteq Expert;$$

- The classes *State*, *Result*, and *Point* are declared as disjoint:

$$State \sqcap Result \sqcap Point \sqsubseteq \bot;$$

- The $Expert$ class has the functional role *hasDescription* common to all classes of expert knowledge:

$$Expert \sqsubseteq \exists hasDescription.String \sqcap \forall hasDescription.String \sqcap = 1 hasDescription.String;$$

- The *Point* class has:
  - The functional role *hasIndex* to associate a value of some attribute with a specific point of a time series,;
  - The functional role *hasResult* to define actions to correct a value of some object attribute;
  - The functional role *hasValue* to define a value of an attribute at some point in a time series;
  - The functional role $forAttribute$ to associate some point of a time series with an object attribute:

$$Point \sqsubseteq \exists hasIndex.Integer \sqcap \forall hasIndex.Integer \sqcap = 1 hasIndex.Integer \sqcap$$
$$\sqcap \exists hasState.State \sqcap \forall hasState.State \sqcap = 1 hasState.State \sqcap$$
$$\sqcap \exists hasResult.Result \sqcap \forall hasResult.Result \sqcap = 1 hasResult.Result \sqcap$$
$$\sqcap \exists hasValue.Double \sqcap \forall hasValue.Double \sqcap = 1 hasValue.Double \sqcap$$
$$\sqcap \exists forAttribute.Attribute \sqcap \forall forAttribute.Attribute \sqcap = 1 forAttribute.Attribute.$$

7.4. Axioms of the Proposed Ontology (ABox)

A set of facts $ABox$ of the proposed ontology is formed automatically using special screen forms to automate the work of an expert.

The following steps are used to form the axioms (O^{D}_{TBox}) for the control problem:

1. Specification of objects:

$$object1 : Object$$
$$(object1,' Some\ object') : hasName.$$

2. Specification of the attributes for each object. Each attribute specifies a data source and a valid range of values:

$$ts1 : TS$$
$$(ts1,' Some\ time\ series') : hasName$$
$$(ts1,' ../data/ts1.csv') : hasData$$
$$attribute1 : Attribute$$
$$(attribute1,' Some\ attribute') : hasName$$
$$(attribute1, 0.5) : hasDeltaMin$$
$$(attribute1, 11.6) : hasDeltaMax$$
$$(attribute1, ts1) : hasTS$$
$$(object1, attribute1) : hasAttribute.$$

Currently, only one-dimensional time series in CSV format are supported.

3. Specification of dependencies between object attributes:

$$dependency1 : Dependency$$
$$(dependency1, attribute2) : dependsOn$$
$$(dependency1, positive) : hasType$$
$$(dependency1, 0.85) : hasWeight$$
$$(attribute1, dependency1) : hasDependency.$$

The following query in the SQWRL language is used to obtain a list of dependencies for the selected attribute of some object:

$$hasDependency(attribute1, ?d) \wedge hasType(?d, positive) \wedge$$
$$\wedge dependsOn(?d, ?adep) \wedge hasWeight(?d, ?w) \rightarrow$$
$$\rightarrow sqwrl : select(attribute1, ?adep, ?w)$$

It is also necessary to add to the ontology the information about the modeling methods of time series and specify their features:

$$method1 : Method \quad (method1,' F - transform') : hasName$$
$$(method1, smooth) : hasFeature$$
$$(method1, fuzzy) : hasFeature$$
$$(method1, periodic) : hasFeature$$
$$(ts1, periodic) : needFeature$$
$$(ts1, fuzzy) : needFeature.$$

The following SQWRL query can be used to obtain suitable time series modeling methods:

$$Method(?m) \wedge hasFeature(?m, ?mf) \circ sqwrl : makeSet(?mfset, ?mf) \wedge$$
$$\wedge sqwrl : groupBy(?mfset, ?m) \wedge$$
$$\wedge hasTS(attr1, ?ts) \wedge needFeature(?ts, ?af) \circ sqwrl : makeSet(?afset, ?af) \circ$$
$$\circ sqwrl : contains(?mfset, ?afset) \rightarrow sqwrl : select(?m).$$

An expert needs to execute the following steps to form axioms of expert knowledge representation (E^{D}_{TBox}):

1. Specification of a set of states for some attribute and a membership function to make fuzzification of attribute values [23]:

$$low: State \quad (low,' Low\ state') : hasDescription$$
$$middle: State \quad (middle,' Middle\ state') : hasDescription$$
$$high: State \quad (high,' High\ state') : hasDescription.$$

A more detailed description of the fuzzy inference module is presented in the Section 7.5.

2. Specification of a set of actions to correct a value of some object attribute:

$$result1: Result \quad (result1,' Some\ result') : hasDescription$$
$$result2: Result \quad (result2,' Another\ result') : hasDescription.$$

3. Specification of a set of SWRL-rules for transition from states to results:

$$forAttribute(?p1, attr1) \wedge hasState(?p1, low) \wedge$$
$$forAttribute(?p2, attr2) \wedge hasState(?p2, middle) \wedge$$
$$forAttribute(?p3, attr3) \wedge hasState(?p3, low) \rightarrow$$
$$\rightarrow hasResult(p1, result1)$$
$$forAttribute(?p1, attr1) \wedge hasState(?p1, middle) \wedge$$
$$forAttribute(?p2, attr2) \wedge hasState(?p2, middle) \wedge$$
$$forAttribute(?p3, attr3) \wedge hasState(?p3, low) \rightarrow$$
$$\rightarrow hasResult(p1, result2).$$

The rules are based on the states of the attributes at a certain point in time (a point in a time series) through the 'forAttribute' role.

7.5. Fuzzy Inference

Conclusions about the state of some objects based on the analysis of their attribute values are formed with fuzzy inference. The following concepts of the fuzzy sets theory are used for fuzzy inference [24,25]:

- Membership functions;
- Linguistic variables;
- Fuzzy implication methods.

A membership function in fuzzy logic determines a membership degree of the elements of a universal set to some fuzzy set.

The fuzzy set for a universal set U and a membership function $\mu: U \rightarrow [0, 1]$ is defined as [24,25]:

$$\tilde{A} = \{(u, \mu_A(x)) | u \in U\}.$$

Membership function $\mu_A(x)$ quantitatively grades the membership of the elements of a universal set $u \in U$ to a fuzzy set $\tilde{A}$. Membership degree zero means that an element is

not included in a fuzzy set, and membership degree one describes a fully included element. Values between zero and one represent fuzzy included elements.

A linguistic variable in the fuzzy set theory takes the semantic of terms in a natural or formal language. Terms represent fuzzy variables and are described by a fuzzy set.

A linguistic variable can be represented as an expression [25]:

$$LT = \{x, T(x), X, G, M\},$$

where x is a variable name, and $T(x)$ is a set of values of a linguistic variable x. Each value of a set is a fuzzy variable on a set X; G is a is a set of additional features for generating new x values; M is a mathematical rule for determining the type of membership function for each value formed based on a G set.

For example, for the linguistic variable 'employees number' (Figure 3):

- x is an employees number;
- X a set of integers from the range [0, 6000];
- $T(x)$ is a set of fuzzy variables: 'low', 'middle', 'high'. It is necessary to set a membership function that specifies information about what employees number should be considered 'low', 'middle', or 'high';
- G is a set of additional features: 'less than', 'more than'. Additional features are used to create new fuzzy variables. For example, 'less than middle' and 'more than low', etc.

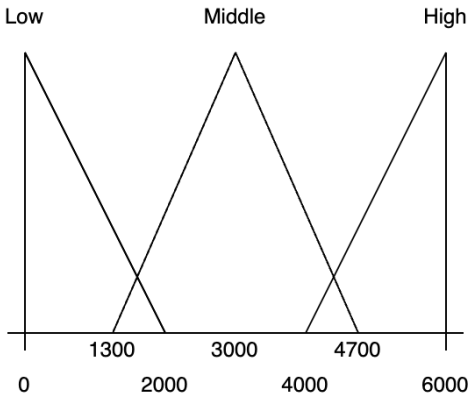

Figure 3. Example of linguistic variables and their membership functions.

Fuzzy inference is executed based on fuzzy rules. A set of fuzzy rules form a fuzzy production system in the context of some subject area. A fuzzy rule can be represented as the expression:

$$R = \langle K, E, W \rangle,$$

where K is the core of a fuzzy rule of the following form:

$$K = \langle A \Rightarrow C \rangle,$$

where $A = \{A_1, A_2, \ldots, A_i, \ldots, A_n\}$ is an antecedent of a rule consisting of many atoms A_i; and $C = \{C_1, C_2, \ldots, C_i, \ldots, C_k\}$ is a consequent of a rule consisting of atoms C_i. Atoms of a fuzzy rule must represent single and compound statements connected by binary operations AND and OR. In SWRL-rules, the $\wedge$ operator represents the AND operation, and multiple queries represent the OR operation; the E function determines a truth degree (from range (0; 1)) of a fuzzy rule result; W is a fuzzy rule weight. If rule weight is not defined, then the weight is one.

The fuzzy inference module uses the Mamdani algorithm. The fuzzy inference consists of the following steps:

1. Fuzzification. Fuzzification is used to switch from numerical indicators of object properties to linguistic variables. The values of all input variables are associated with

specific values of the membership functions of linguistic terms from antecedents of fuzzy rules at the fuzzification stage.

A set of attribute states is formed in the process of fuzzy inference based on the values of time series points ('hasValue' role). Each point of a time series is associated through the 'hasState' role with a corresponding linguistic term in the process of fuzzy inference [23]. The 'hasState' role describes a membership of some time series point to a certain fuzzy set with some degree of membership:

$$point_i : Point \quad (point_i, i) : hasIndex$$
$$(point_i, s_j \in State) : hasState.$$

2. Aggregation. A truth degree of antecedents for each rule is determined at the aggregation stage. If an antecedent of a fuzzy rule contains one atom, then a truth degree of an antecedent is a truth degree of this atom. A truth degree of an atom is calculated based on the value of a membership function of a linguistic variable term. If an antecedent of a rule contains several atoms, then a truth degree is calculated based on the truth degrees of the antecedent atoms using fuzzy logic AND (min) operator.
3. Activation. A truth degree of each consequent atom of the fuzzy rule is determined at the stage of activation. A truth degree of each consequent atom is equal to the algebraic product of a rule weight and a truth degree of a rule antecedent. If weights of production rules are not specified, then their default values are equal to one. Minimum is used to calculate truth degrees.

   A Set of results for object control is formed as a result of the inference. It is possible to obtain results for each point of a time series or for the individual points. Fuzzy inference allows obtaining a set of solutions ordered by their degree of truth. The degree of truth is calculated at the stages of aggregation and activation of the fuzzy inference [23]. For example:

Attribute	Result	Result description	Truth degree
attr1	result1	Some result	0.356
attr1	result2	Another result	0.047

4. Accumulation. A membership function is formed for each linguistic variable from the consequent fuzzy rules at the accumulation stage. Accumulation is based on the union of fuzzy sets of all consequent atoms for some linguistic variable using fuzzy logic OR (max) operator.
5. Defuzzification. The result of defuzzification is quantitative (crisp) values for each output linguistic variable based on the results of the accumulation of all output linguistic terms from the consequences of fuzzy rules using the center of gravity method.

As you can see from Figure 2, the proposed fuzzy inference mechanism can produce two types of results:

1. A set of recommendations for decision makers ordered by truth degree. The result is formed at the aggregation step, the accumulation and defuzzification steps are not required.
2. A set of numerical values to regulate the behavior of the object. All steps must be completed.

Thus, the proposed ontology model makes it possible to form a context for solving the control problem. Creation of axioms for the proposed ontology does not require knowledge of special tools from an expert. Using fuzzy inference makes it easier to form a set of SWRL-rules.

8. Results

We will consider the analysis according to the scheme in Figure 2. The object under control is the software project development process. The vector of input parameters $X = \{x_1, \ldots, x_n\}$ will be a set of requirements for developers: a list of tasks to be imple-

mented and time limits. The output data $V = \{v_1, \ldots, v_m\}$ of the control object will be artifacts and indicators extracted from the git repositories of the program code, testing systems, and source code analysis.

By themselves, the values of these parameters may not reflect the actual state of the project being implemented. Therefore, we introduce an integrated assessment, which makes it possible to track the influence of input data, control actions and random influences as a quality assessment in dynamics.

Two main sections: product quality and quality of development processes, represent software development quality models. Development management keeps track of both key metrics. Therefore, it is possible to identify development indicators that will influence the formation of quality. For each indicator, the history of the value change as a time series is retrieved.

We show the application of the proposed approach on one of the software projects to automate the activities of the university research group [26]. Junior programmers developed the project over several months.

The source data for analysis are the indicators of the git source code repository, which reflects the indicators of the development process and its results. We host the repository on the GitLab service, which allows storing and versioning the source code, to automate development processes: building and testing the project, keeping statistics, and managing project tasks.

Let s provide a description of each indicator. The indicator values are average values over a weekly time interval.

- The number of features is determined by the number of issues (issue) in the "Implemented" state.
- Feature execution time is determined by the cumulative execution time of automated tests.
- Exception count is the sum of all exceptions found during testing.
- Test time is the cumulative runtime of unit tests.
- Project build time is the average time of compilation of a project and running in automatic mode to check if it works.
- Efficiency of developers is determined by the number of completed tasks in a weekly interval.
- Bugs in the code are determined using the SonarQube service to obtain an indicator of the static code analysis system.
- Code test coverage shows the measure of checking all the implemented functionality with unit tests.
- Code writing time reflects a measure similar to developer efficiency, the time developers spend on tasks. However, unlike the first indicator, the measurement of this indicator does not include such overhead costs as task management, life cycle changes, discussion, and testing costs.

Table 2 also lists the weights of indicator links that affect quality models. That is, for example, the relationship between the problem area object "Product Quality" and "Number of Features" is ranked by weight five.

By the steps of the algorithm (see the Section 5), the ontology describes choosing methods for predicting time series and relationships between analyzed objects (see the Section 7.2).

Figures 4 and 5 show examples of extracted time series.

Forecast values of indicators allow making a logical inference in the ontology. These forecasted values are input as parameters according to the trends in the time series of indicators, as shown in Table 3. In this experiment, the forecast horizon was based on known real values to help assess the quality of the forecasts. The test forecast horizon covers time series points from 11 to 16. The described earlier in [27] approach forecasts by choosing a model suitable for the time series context.

Table 2. Indicators of development and theirs impact on quality models.

	Indicator Name	Relation Weight
Product quality	Features count	5
	Features execution time	−3
	Exception count	−1
	Testing time	−2
Project quality	Building time	−1
	Developers efficiency	5
	Code faults	−4
	Code coverage by tests	3
	Coding time	−5

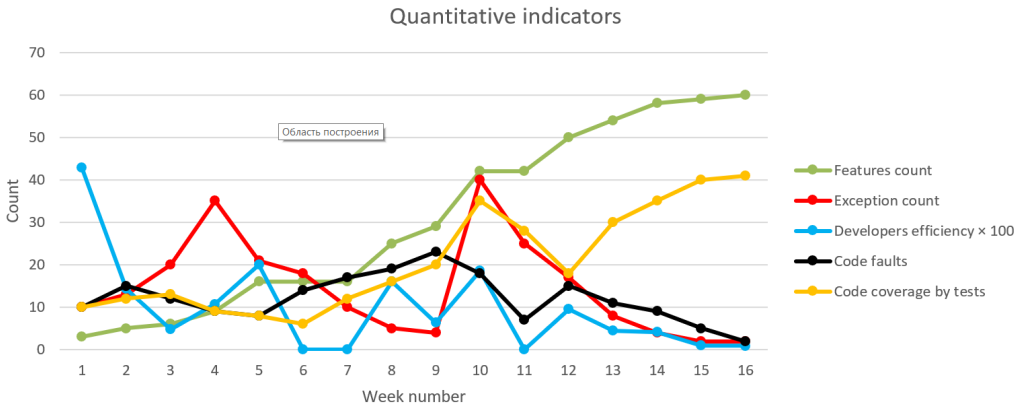

Figure 4. Time series of quantitative indicators.

Forecast values represent only a single trend value per forecast interval (for simplicity). Based on the forecast data and information from the ontology about the allowable change intervals, the following indicators have deviations: exception count, building time, and testing time parameters. The values of these indicators are inflated compared to the model's predicted values.

Previously, there were certain links between indicators and objects of management—the quality of development processes and the quality of a software product. This relationship can explain why the state of the control object is deteriorating and in what proportion.

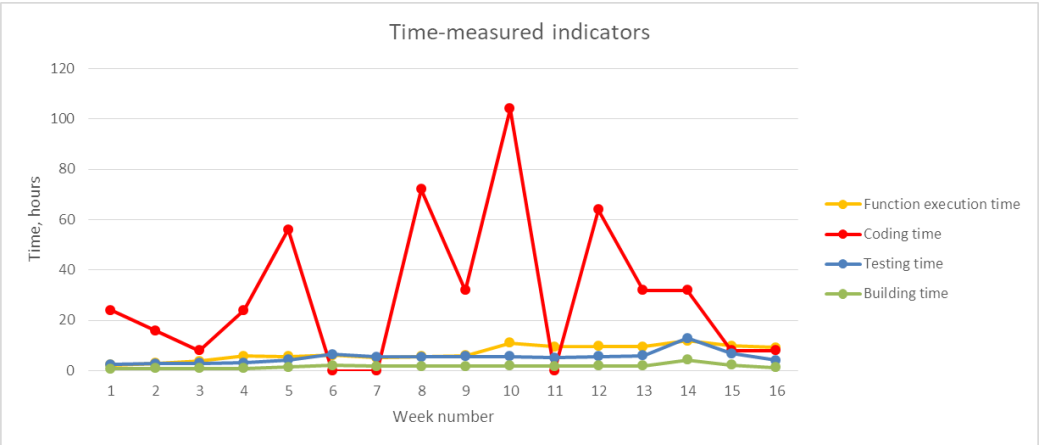

Figure 5. Time series of time-measured indicators.

Table 3. Forecasts of indicators of development.

Indicator Name	Actual Value	Forecast Tendency Value	Smape, %
Features count	60	63	1.28
Features execution time	9.1	10	4.35
Exception count	1.9	1.7	10.79
Testing time	8.81	4.4	15.83
Building time	3.55	1.42	15.65
Developers efficiency	0.14	0.72	0.01
Code faults	1.99	1.91	4.67
Code coverage by tests	40.8	45.1	3.83
Coding time	8	9	0.1

In the compiled ontology, the boundaries of the permissible values of indicators, as well as the rules for responding to exceeding the boundaries, are determined.

Let us assess the control object indicators based on the forecast values as shown in Table 4. We made the calculation considering the normalized values because the weights of the $r_{o_k o_l}$ indicators according to (2) are already normalized relative to each other, and the absolute values of the indicators can differ greatly. We normalize considering the maximum value among the current and forecast values of indicators. For example, for the "Features count" indicator, the maximum value will be 63. Further, using the formula $value * weight$, we will recalculate the values of indicators from the Table 3.

The values from Table 4 are used to calculate the integral quality index Q_{total}. The value of the software system quality model reflects a higher value of the quality of the software product. It was the primary target of the development of the software project, and the indicators of the quality model of the development processes significantly worsened its total value. Predictive models reflected this state. The final assessment of the quality is $Q_{total} = 5.94$.

During the implementation of the project, we made organizational changes. We changed the test environment (continuous integration server) and the logic for testing developers' code changed. These changes allowed us to improve the indicators "|Testing time" and "Project build time". After the changes, Q_{total} was 2.75.

Table 4. Quality models evaluation.

	Indicator Name	Weight × Normalized Value	Weight × Normalized Forecast
Product quality	Features count	4.7	5
	Features execution time	−1.72	−1.89
	Exception count	−0.03	−0.02
	Total	2.95	3.09
Project quality	Testing time	−1.11	−0.55
	Building time	−0.22	−0.08
	Developers efficiency	0.01	0.057
	Code faults	−0.12	−0.12
	Code coverage by tests	1.94	2.14
	Coding time	−2.52	−2.84
	Total	−1.02	−1.48

9. Conclusions

The results presented in this article develop a previously proposed approach for the contextual analysis of indicators of objects in the problem area, expressed as time

series. This approach helps obtain a set of models that reflect the features of the analyzed objects, reduces the burden on experts, and has complete possibilities for interpreting the simulation result.

This article showed an approach to solving a control problem based on the dynamics of changes in the indicators of a set of objects in the problem area.

The advantages of this approach are the ability to use more complete data for making management decisions, as well as reducing the analysis time through the use of intelligent models. Thus, the experiment conducted on the data of the git repository of the program code showed that the proposed approach also made it possible to increase the explainability of the analysis results. We show the difference between the proposed approach as a hybrid solution between traditional control systems and a decision support system.

Further work within the framework of this study involves the creation of a control system for solving problems of automatic control based on the analysis of the context of the dynamics of changes in the parameters of objects.

Author Contributions: Conceptualization, A.A.R.; data curation, A.A.F.; formal analysis, A.A.F.; methodology, N.G.Y.; visualization, A.A.R. All authors contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Higher Education of the Russian Federation in the framework of the state task No. 075-00233-20-05 "Research of intelligent predictive multimodal analysis of big data, and the extraction of knowledge from different sources".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pospelov, D.A. *Situational Management: Theory and Practice*; Fizmatlit: Moscow, Russia, 1986. (In Russian)
2. Xu, J.; Wang, Q.; Lin, Q. Parallel robot with fuzzy neural network sliding mode control. *Adv. Mech. Eng.* **2018**, *10*. [CrossRef]
3. Nguyen, A.-T.; Taniguchi, T.; Eciolaza, L.; Campos, V.; Palhares, R.; Sugeno, M. Fuzzy Control Systems: Past, Present and Future. *IEEE Comput. Intell. Mag.* **2019**, *14*, 56–68. [CrossRef]
4. Tabbussum, R.; Dar, A.Q. Performance evaluation of artificial intelligence paradigms—Artificial neural networks, fuzzy logic, and adaptive neuro-fuzzy inference system for flood prediction. *Environ. Sci. Pollut. Res.* **2021**, *28*, 25265–25282. [CrossRef]
5. Frazzon, E.M.; Kück, M.; Freitag, M. Data-driven production control for complex and dynamic manufacturing systems. *CIRP Ann.* **2018**, *67*, 515–518.
6. Bejarano, G.; Alfaya, J.A.; Rodríguez, D.; Morilla, F.; Ortega, M.G. Benchmark for PID control of refrigeration system based on vapour compression. *IFAC PapersOnLine* **2018**, *51*, 497–502.
7. Márquez-Vera, M.A.; Ramos-Fernández, J.C.; Cerecero-Natale, L.F.; Lafont, F.; Balmat, J.F.; Esparza-Villanueva, J.I. Temperature control in a MISO greenhouse by inverting its fuzzy model. *Comput. Electron. Agric.* **2016**, *124*, 168–174. [CrossRef]
8. Kotenko, I.; Saenko, I.; Ageev, S. Hierarchical fuzzy situational networks for online decision-making: Application to telecommunication systems. *Knowl.-Based Syst.* **2019**, *185*, 104935. [CrossRef]
9. Petrushevskaya, A.A. Control model of technological operations of mounting automatic printed circuit boards based on a multiparameter fuzzy classifier. *J. Phys. Conf. Ser.* **2019**, *1333*, 042026.
10. Morales Escobar, L.; Aguilar, J.; Garcés-Jiménez, A.; Gutierrez De Mesa, J.A.; Gomez-Pulido, J.M. Advanced Fuzzy-Logic-Based Context-Driven Control for HVAC Management Systems in Buildings. *IEEE Access* **2020**, *8*, 16111–16126. [CrossRef]
11. Novák, V.; Mirshahi, S. On the Similarity and Dependence of Time Series. *Mathematics* **2021**, *9*, 550. [CrossRef]
12. What Is Scrum. Available online: https://www.scrum.org/resources/what-is-scrum (accessed on 30 December 2022).
13. ISO/IEC 25010:2011. Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models. Available online: https://www.iso.org/ru/standard/35733.html (accessed on 30 December 2022).
14. Romanov, A.; Filippov, A.; Yarushkina, N. An Approach to Contextual Time Series Analysis. In Proceedings of theArtificial Intelligence and Soft Computing: 20th International Conference, ICAISC 2021, Virtual Event, 21–23 June 2021.
15. SMAPE Criterion by Computational Intelligence in Forecasting (CIF). Available online: http://irafm.osu.cz/cif/main.php (accessed on 30 December 2022).

16. Perfilieva, I.G.; Yarushkina, N.G.; Afanasieva, T.V.; Romanov, A.A. Web-Based System for Enterprise Performance Analysis on the Basis of Time Series Data Mining. In Proceedings of the First International Scientific Conference "Intelligent Information Technologies for Industry" (IITI-16), Rostov-on-Don, Russia, 16–21 May 2016; pp. 75–86.
17. Romanov, A.; Filippov, A.; Yarushkina, N. Extraction and Forecasting Time Series of Production Processes. In Proceedings of the International Conference on Information Technologies, Paris, France, 23–26 April 2019; pp. 173–184.
18. Guarino, N.; Musen, M.A. Ten years of applied ontology. *Appl. Ontol.* **2015**, *10*, 169–170.
19. Horrocks, I.; Patel-Schneider, P.F.; Boley, H.; Tabet, S.; Grosof, B.; Dean, M. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Memb. Submiss.* **2004**, *21*, 1–31.
20. Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; Patel-Schneider, P.F. *The Description Logic Handbook: Theory, Implementation, and Applications*; Cambridge University Press: New York, NY, USA, 2003.
21. Bonatti, P.; Tettamanzi, A. Some complexity results on fuzzy description logics. In Proceedings of the WILF 2003 International Workshop on Fuzzy Logic and Applications, Naples, Italy, 9–11 October 2003; Volume 2955.
22. Grosof, B.; Horrocks, I.; Volz, R.; Decker, S. Description logic programs: Combining logic programs with description logics. In Proceedings of the WWW 2003, Budapest, Hungary, 20–24 May 2003; pp. 48–57.
23. Romanov, A.; Stroeva, J.; Filippov, A.; Yarushkina, N. An Approach to Building Decision Support Systems Based on an Ontology Service. *Mathematics* **2021**, *9*, 2946. [CrossRef]
24. Zadeh, L.A. Fuzzy logic. *Computer* **1988**, *21*, 83–93. [CrossRef]
25. Zadeh, L.A. The concept of a linguistic variable and its application to approximate reasoning—I. *Inf. Sci.* **1975**, *8*, 199–249. [CrossRef]
26. Automation of Scientific Group Activities. Available online: https://gitlab.com/romanov73/ng-tracker (accessed on 30 December 2022).
27. Romanov, A.A.; Filippov, A.A.; Voronina, V.V.; Guskov, G.; Yarushkina, N.G. Modeling the Context of the Problem Domain of Time Series with Type-2 Fuzzy Sets. *Mathematics* **2021**, *9*, 2947. [CrossRef]