processes

MDPI

Article

Modeling and Analysis of Distributed Control Systems: Proposal of a Methodology

Milan Tkáčik [1], Ján Jadlovský [1], Slávka Jadlovská [2,*], Anna Jadlovská [1,*] and Tomáš Tkáčik [1]

[1] Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, Slovakia; milan.tkacik@tuke.sk (M.T.); jan.jadlovsky@tuke.sk (J.J.); tomas.tkacik@tuke.sk (T.T.)

[2] Department of Industrial Engineering and Informatics, Faculty of Manufacturing Technologies with the Seat in Prešov, Technical University of Košice, Bayerova 1, 080 01 Prešov, Slovakia

* Correspondence: slavka.jadlovska@tuke.sk (S.J.); anna.jadlovska@tuke.sk (A.J.)

Abstract: A Distributed Control System is a concept of Network Control Systems whose applications range from industrial control systems to the control of large physical experiments such as the ALICE experiment at CERN. The design phase of the Distributed Control Systems implementation brings several challenges, such as predicting the throughput and response of the system in terms of data-flow. These parameters have a significant impact on the operation of the Distributed Control System, and it is necessary to consider them when determining the distribution of software/hardware resources within the system. This distribution is often determined experimentally, which may be a difficult, iterative process. This paper proposes a methodology for modeling Distributed Control Systems using a combination of Finite-State Automata and Petri nets, where the resulting model can be used to determine the system's throughput and response before its final implementation. The proposed methodology is demonstrated and verified on two scenarios concerning the respective areas of ALICE detector control system and mobile robotics, using the MATLAB/Simulink implementation of created models. The methodology makes it possible to validate various distributions of resources without the need for changes to the physical system, and therefore to determine the appropriate structure of the Distributed Control System.

Keywords: Cyber-Physical System; Hybrid System; Finite-State Automata; Petri net; Distributed Control System; Detector Control System

Citation: Tkáčik, M.; Jadlovský, J.; Jadlovská, S.; Jadlovská, A.; Tkáčik, T. Modeling and Analysis of Distributed Control Systems: Proposal of a Methodology. *Processes* **2024**, *12*, 5. https://doi.org/10.3390/pr12010005

Academic Editor: Hsin-Jang Shieh

Received: 9 October 2023
Revised: 8 December 2023
Accepted: 11 December 2023
Published: 19 December 2023

1. Introduction

A Distributed Control System (DCS) is a concept of Network Control Systems [1,2] often used in industrial applications where the distribution of resources throughout the system brings significant advantages. The DCS is characterized by a multi-level architecture, where individual control levels are connected by different types of communication networks, as described in the IEC 61499 [3,4]. In the case of the need to control and capture data from the controlled process in real-time, it is necessary to consider the limitations resulting from the communication and computing processes in terms of throughput and response: these system parameters can have a significant impact on the quality and stability of implemented control [5]. The estimation of these parameters may not be a trivial task in the case of more complex Distributed Control Systems with a variable size of transmitted data [6]. To estimate the throughput and response of a DCS, this system can be considered a Cyber-Physical System (CPS) that can be modeled and analyzed using the concept of Hybrid Systems [7,8]. Computing processes and communication networks within the DCS can be considered as systems with discrete events, which are convenient to model using a number of approaches including Finite-State Automata and Petri nets [9–11].

A prominent example of the implementation of Distributed Control Systems architecture is the Detector Control System (ALICE-DCS) of the ALICE experiment (A Large Ion

Collider Experiment) at the Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN). The ALICE experiment helps understand the formation of the early universe by studying the quark–gluon plasma, i.e., the fifth state of matter that is presumed to have filled the universe shortly after the Big Bang, when quarks and gluons could move freely. In the LHC, the quark–gluon plasma is created by heavy ion collisions, which are captured by a complex of ALICE experiment detectors [12]. The ALICE-DCS ensures stable and safe operation of the detectors while ensuring the tasks of control, monitoring, and data acquisition from the detector electronics [13]. The ALICE experiment and its detectors underwent modernization from 2018 to 2022, which also increased the monitoring frequency of the detector electronics and caused an order-of-magnitude increase in the amount of data that needs to be processed and distributed in real time [14]. Such an increase in real-time data processing requirements would result in an unbearable load on the supervisory control and data acquisition (SCADA) systems within the ALICE-DCS. For this reason, a new software layer, ALICE Low-Level Front End Device (ALFRED), co-developed by the authors of this paper as part of the *ALICE experiment at the CERN LHC: The study of strongly interacting matter under extreme conditions* project, was included in the control system of the detectors of the ALICE experiment. ALFRED ensures data processing before being forwarded to the SCADA system while also creating an abstraction layer for detector electronics from the point of view of the SCADA system [15]. The distribution of software and hardware resources within the ALICE-DCS is mostly determined experimentally, which is a lengthy and iterative process, which results in the need for modeling the ALICE-DCS to determine the appropriate distribution of the used resources [16]. As similar conclusions were drawn regarding the application of mobile robotics in the context of the DCS infrastructure, which had been developed beforehand at the Center of Modern Control Techniques and Industrial Informatics (CMCT&II) at the Department of Cybernetics and Artificial Intelligence (DCAI) FEEI TU of Košice [17,18], an idea for a unified methodology for modeling and analysis of Distributed Control Systems was conceived.

This paper is structured as follows. The first part presents the DCSs in the context of CPSs, including the possibilities for their modeling. The following part describes the infrastructures of DCSs, which are to be considered in scenarios to illustrate the methodology. The last part of the paper is devoted to the step-by-step description of the proposed methodology for modeling and analysis of Distributed Control Systems. The methodology is subsequently applied to the control system of the detectors of the ALICE experiment and to the application of mobile robotics in the context of the DCS at the CMCT&II.

2. Modeling of Distributed Control Systems—Fundamental Concepts

As is the case with a variety of processes, the numerous modeling methods for Distributed Control Systems can be applied both in DCS design and subsequent analyses [19,20]. Based on the idea that CPSs integrate the computational and physical part of the system through the hierarchical layers of the DCS automation pyramid, we will hereafter show how Distributed Control Systems can be represented as Cyber-Physical Systems and subsequently modeled as systems with discrete events. Basic related concepts, including Hybrid Systems formalisms and their pertinent applications, are introduced.

2.1. Cyber-Physical Systems and Hybrid Systems

Cyber-Physical Systems, a prominent technology of Industry 4.0 [21], are computer systems whose physical processes are monitored, controlled, and coordinated by computing and communication resources. From the conceptual point of view, CPSs can be defined as an aggregate consisting of interconnected computational and physical processes [22]. An essential element of CPSs are networks that ensure communication between individual computing processes, actuators, and sensors. A conceptual diagram of the structure of the CPS can be seen in Figure 1.

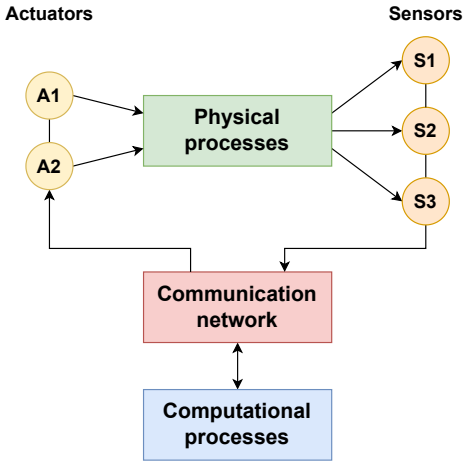

Figure 1. Schematic representation of the concept of a Cyber-Physical System.

Given the definition of the CPS, the Distributed Control System can also be considered a Cyber-Physical System. The DCS is a computer control system for complex processes in which individual control and coordination processes are distributed within the system without a central control node [23]. Unlike a Centralized Control System, individual control processes are located closer to the controlled processes, which makes the control system more reliable and characterized by lower initial costs for implementation. At the same time, supervisory systems monitor and supervise individual subsystems, thereby obtaining a comprehensive overview of the state of controlled processes. The architecture of the DCS is standardly classified into several control/functional levels, as specified by IEC 62264 (ANSI/ISA-95) [24], with some variations in level nomenclature and numbering in actual implementations [25,26]. The hierarchical structure and nomenclature employed in the DCS architecture at the CMCT&II [27] is illustrated in Figure 2.

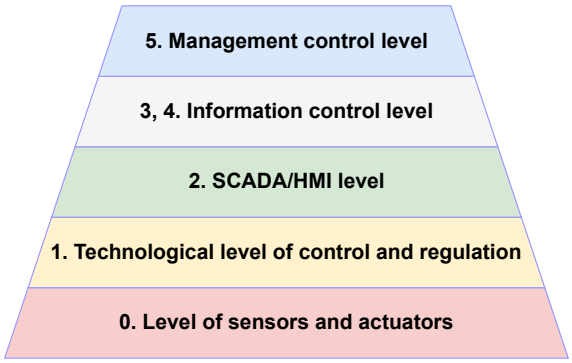

Figure 2. Considered multi-level architecture of Distributed Control Systems.

In terms of modeling, a Cyber-Physical System (and therefore the DCS it represents) can be described using the concept of Hybrid Systems [7]. Hybrid Systems are characterized by the fact that their behavior exhibits both continuous and discrete dynamics. A Hybrid System can have one or more continuous dynamics, which can be described by differential equations considering a continuous input $u(t)$ and a continuous output $y(t)$; thus, it is possible to express the behavior of physical processes within the CPS. From the point of view of the discrete dynamics of Hybrid Systems, it is possible to consider a discrete input $\sigma(t)$ and a discrete output $w(t)$ of the system, which makes it possible to represent the behavior of computing processes and communication networks of the CPS [28].

The continuous and discrete dynamics of the Hybrid System interact with each other, as can be seen in Figure 3, where a continuous–discrete interface (*event generator*) transforms a continuous signal into a discrete event that can result in a state transition subsystem of discrete dynamics. At the same time, the discrete–continuous interface (*injector*) assigns a

continuous value to the discrete signal, which acts as the input of the continuous dynamics subsystem [29]. In order to model the discrete dynamics of a Hybrid System, it is possible to use various formalisms [20], including the Finite-State Automata and Petri nets, which will be subsequently characterized.

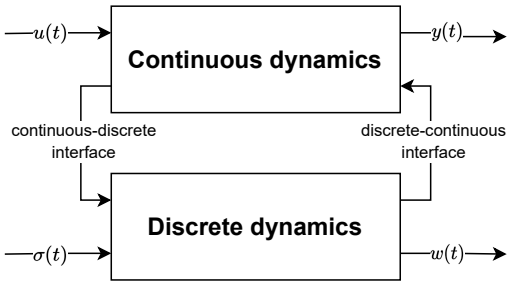

Figure 3. Interconnection of the continuous and discrete dynamics of Hybrid Systems.

2.2. Modeling of Systems with Discrete Events

Finite-State Automaton can be considered as a subset of Hybrid Automata, when we do not take into account the continuous dynamics of the described system and only express the discrete dynamics of the modeled system by the automata. Finite-State Automaton is an abstract mathematical model describing a system that is in exactly one discrete state from a finite set of states at a specific time. Supplying inputs to the automaton results in a transition between individual states based on a defined transition function [30].

A Finite-State Automaton can be expressed as a tuple:

$$M = (Q, \Sigma, Init, R, F) \tag{1}$$

where the individual elements have the following meaning:

- Q is a finite set of discrete states of the system taking the values $\{q_1, q_2, \ldots, q_m\}$;
- Σ is a finite set of discrete system inputs taking on values $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$;
- $Init \in Q$ is the initial state of the Finite-State Automaton;
- $R : Q \times \Sigma \rightarrow Q$ determines the transition function, which determines the new state q_s based on the ordered pair of the previous state q_i and the input σ_j;
- $F \subseteq Q$ is the set of final states of the automaton (can be empty).

Figure 4 depicts an example of a graphical representation of a finite-state automaton with three discrete states $\{q_1, q_2, q_3\}$, two discrete inputs $\{\sigma_1, \sigma_2\}$, and the transition function $\{R(q_i, \sigma_j)\}$ as defined for the individual ordered pairs of discrete states and inputs.

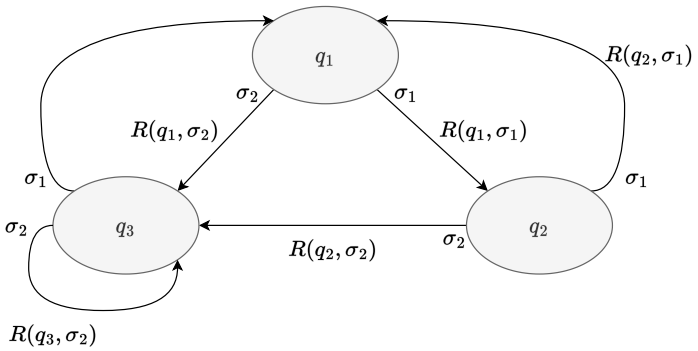

Figure 4. An example of a graphical representation of a Finite-State Automaton.

A Petri net is a mathematical model for describing systems with discrete events, making Petri nets a suitable candidate for modeling DCSs from a data-flow perspective [11]. A Petri net can be described as a bipartite directed graph containing two types of nodes—places and transitions. Individual places and transitions are connected by oriented

edges, where two places or two transitions cannot be directly connected. Thus, there must be a second type of node between two identical types of nodes. It is possible to place a non-negative amount of tokens (a token representing, for example, transmitted data) on each place, while tokens can be moved between places based on defined rules through activated or fired transitions [31].

The formal definition of Petri nets can be expressed as a tuple

$$PN = (S, T, E, V, C, M_0) \tag{2}$$

where individual elements have the following meaning:

- S is a finite set of places taking on the values $\{s_1, s_2, \ldots, s_m\}$;
- T is a finite set of transitions taking the values $\{t_1, t_2, \ldots, t_n\}$, where $S \cap T = \emptyset$;
- $E \subseteq (S \times T) \cup (T \times S)$ is the set of edges (arcs), i.e., the union of sets of edges oriented from places to transitions and from transitions to places;
- $V : E \to \mathbb{N}^*$ is a function for evaluating network edges with positive weights;
- $C : S \to \mathbb{N}^* \cup \infty$ is a function determining the maximum capacity of tokens in individual places;
- $M_0 : S \to \mathbb{N}^0 \cup \infty$ is the initial distribution of tokens in the network respecting the constraint $M_0(p) \leq C(p)$ for $\forall p \in S$.

To expand the application possibilities of Petri nets, several extensions have been developed, such as Colored Petri nets, where it is possible to assign transmitted data to individual tokens in form of a set of values (parameters). An example of a graphical representation of a colored Petri net with four places $\{s_1, s_2, s_3, s_4\}$, two transitions $\{t_1, t_2\}$, and a sequence of parameters associated with the token and the edges can be seen in Figure 5. Another extension of Petri nets are the Timed Petri nets, where time intervals can be associated with individual transitions; this makes it possible to model the duration of executed processes [32]. With the help of these extensions, it is possible to reliably model the behavior of Distributed Control Systems in terms of data-flow through communication interfaces and networks [33–35].

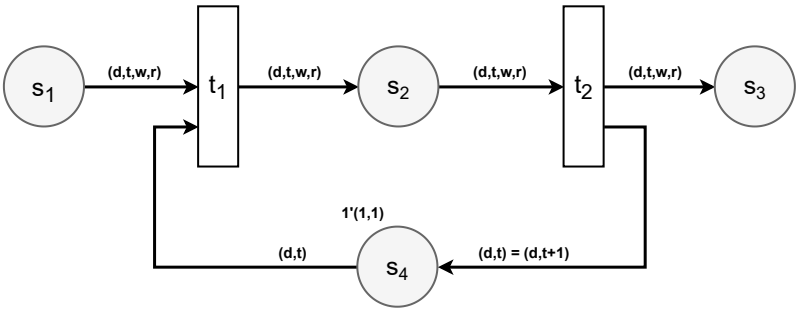

Figure 5. An example of a graphical representation of a Colored Petri net.

3. Considered Distributed Control Systems

The idea of developing a unified modeling methodology for Distributed Control Systems was inspired by the mutual similarities at various levels of abstraction between two different infrastructures of DCSs: the control system of the detectors of the ALICE experiment at CERN, and a mobile robotics application as integrated in the DCS at the CMCT&II at DCAI FEEI TU in Košice. We now present both original infrastructures. The application of the proposed methodology on considered systems will be demonstrated in two scenarios in Sections 4.1 and 4.2.

3.1. The Infrastructure of the Detector Control System of the ALICE Experiment

The ALICE experiment is a complex of 18 detectors at the Large Hadron Collider *LHC* at CERN, which is focused on the study of ultrarelativistic collisions of heavy ions [36]. The particle collision site in the ALICE experiment is surrounded by multiple layers of

particle detectors located inside an L3 magnet capable of developing a magnetic field with an induction of 0.5 T. The resulting magnetic field ensures the curvature of trajectories of new particles created during the collisions of accelerated beams.

Control and monitoring of the detectors of the ALICE experiment is handled by the ALICE Detector Control System. Although the architectures of the control systems of the individual detectors use the same concept, their structures are slightly different due to the different detector electronics of the individual detectors [37]. As an example of the architecture of the ALICE-DCS, the architecture of the Inner Tracking System (ITS) Detector Control System is presented, which can be seen in Figure 6. The system structure can be divided into four basic subsystems: the ALFRED Distributed System (Frontend System), the Power System, the Detector Safety System, and the Cooling System [15].

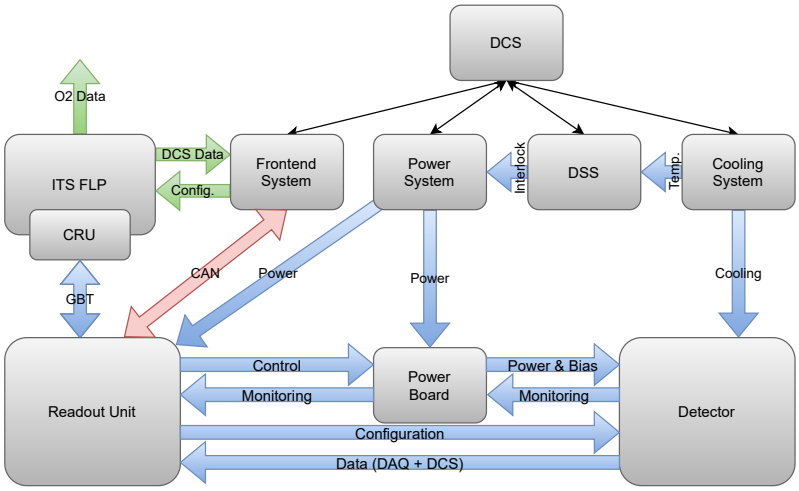

Figure 6. Architecture of the ITS Detector Control System [15].

The Detector Control System of the ALICE experiment can be generalized to the multi-level DCS architecture presented in Section 2.1, while the individual components of the ALICE-DCS can be classified into the appropriate control levels according to the presented architecture, as can be seen in Figure 7. The lowest level includes the detector electronics, while the next level ensures low-level control using control computers and automata. The SCADA/HMI level ensures the coordination of control processes based on data obtained from the configuration databases. The acquired ALICE-DCS and physics data are stored in the archival databases, which are subsequently accessible using the Worldwide LHC Computing Grid (WLCG) [38].

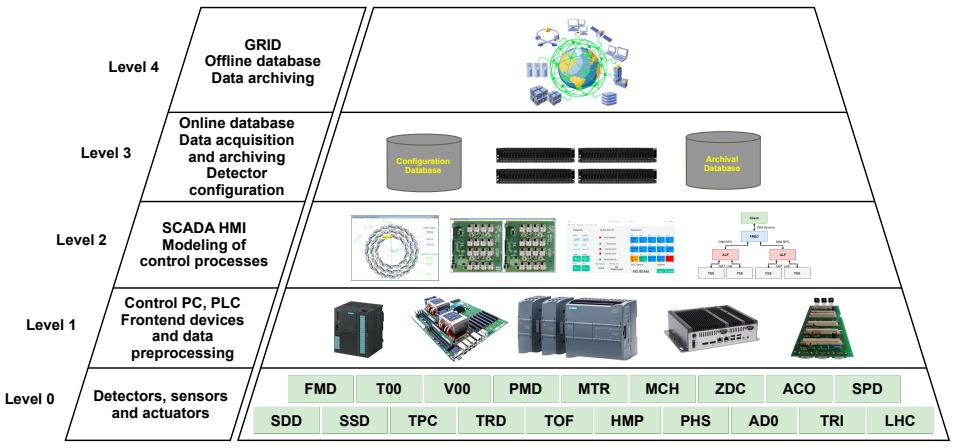

Figure 7. The inclusion of the ALICE Detector Control System components in the concept of Distributed Control Systems.

The introduced ALFRED System is also characterized by a distributed architecture [15], where individual components can be classified into individual levels of the architecture of the Distributed Control System, as can be seen in Figure 8. The lowest level of the ALFRED System consists of the detector electronics, in the case of the ITS detector specifically the Readout Unit and Power Board units. The Readout Unit provides data collection and control of the detector itself, while they communicate with the higher level of ALICE Low-Level Front End (ALF) and CANbus ALICE Low-Level Front End (CANALF) applications via a gigabit optical line—GigaBit Transceiver (GBT), or by using the CAN bus interface [39]. The ALF and CANALF applications ensure the translation of messages into the Distributed Information Management System (DIM) protocol format, through which communication with the software layer of the Front End Device (FRED) [40] applications is carried out.

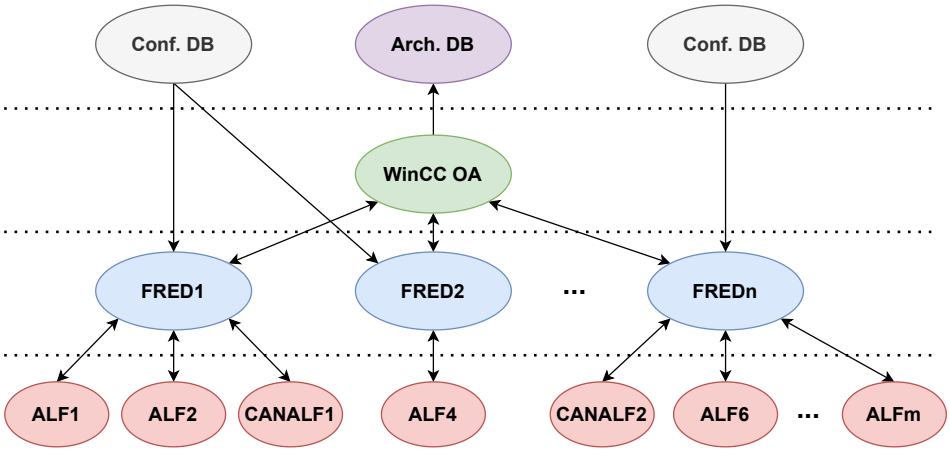

Figure 8. Distributed architecture of the ALFRED system.

The FRED application creates an abstract view of the detector electronics for the SCADA/HMI system from the point of view of unifying communication protocols and message formats. At the same time, it relieves the SCADA/HMI system of computationally intensive initial data processing and ensures the control and monitoring of detectors at the lowest level [41]. The SCADA/HMI system WinCC OA ensures supervisory control of detectors, which is implemented using Finite-State Machines (FSM) [42,43], and at the same time provides the possibility of controlling detectors through operator panels [44]. The highest level of the ALFRED System is a layer of configuration and archiving databases, which provide parameters for the configuration of detector electronics and archive physical and technical data obtained during the course of the experiment [37].

3.2. Applications of Mobile Robotics within the DCS Infrastructure at the CMCT&II

A number of implementations involving mobile robots are characterized by a distributed architecture, where the components of a mobile robotics application can be classified into appropriate levels of a DCS according to the concept presented in Section 2.1. These are often applications based on multi-agent systems, where several mobile robots are used in one application [45]. In this case, each mobile robot is an independent functional unit comprising lower levels of a DCS, including monitoring its surroundings and controlling its own movement. Higher levels of control are usually implemented outside of the mobile robots themselves, using external computers that provide supervisory control, tactical planning, or data acquisition and archiving [46,47].

As a part of the DCS at the CMCT&II at DCAI FEEI TU in Košice, the application of robotic soccer of the MiroSot category is considered [48]. The application uses differential-drive two-wheeled mobile robots MiroSot and a supervisory computer ensuring the localization of the robots and tactical planning of their movement in order to implement the robot soccer game itself [49]. The robotic soccer application can be considered a Distributed

Control System, and the individual components of the application can be integrated in the DCS architecture (Section 2.1), according to Figure 9.

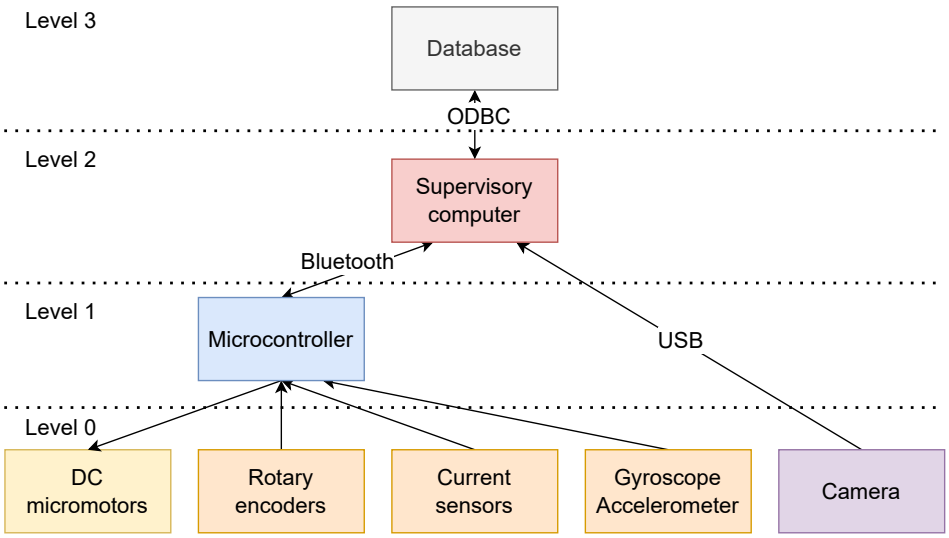

Figure 9. Inclusion of the robotic soccer application into the Distributed Control System concept at the CMCT&II.

The Distributed Control System at the CMCT&II has also been considered in the design and realization of the modular robotic platform ModBot, which is characterized by high modularity in terms of configuring the robotic platform for the needs of specific applications. Unlike MiroSot mobile robots, the ModBot platform can be easily expanded with additional sensors and actuators using multiple slots for add-on modules, thanks to which the ModBot platform can be used for a wide range of distributed applications [50].

4. Methodology for Modeling and Analysis of Distributed Control Systems

Design and implementation of Distributed Control Systems involve many challenging issues including network-induced delays, time-varying topology or throughput, or increasing complexity. The need to address these issues has stimulated the development of methodologies for modeling, analysis and synthesis of DCSs [5], which standardly rely on computational tools based on Java, C++ or MATLAB/Simulink [2,51,52] to perform simulations based on resulting models. Such approaches have enabled model-based analyses of DCS throughput/response time [53,54].

In this section, we present the proposed methodology for modeling and analysis of Distributed Control Systems. The methodology represents a procedure for creating a complex model of a DCS and subsequent analysis of the properties of the modeled system, based on the considerations in Section 2. The models created using this methodology can be easily implemented in MATLAB/Simulink using the Stateflow tool, which makes it possible to perform simulations to determine the properties of the DCS, such as throughput and response time, based on the supplied inputs (see illustrative scenarios). The intended features of the methodology include universality, employment of the state-of-the-art software tools, and various application areas. The methodology consists of three modules, each of which is divided into several submodules, as shown in Figure 10.

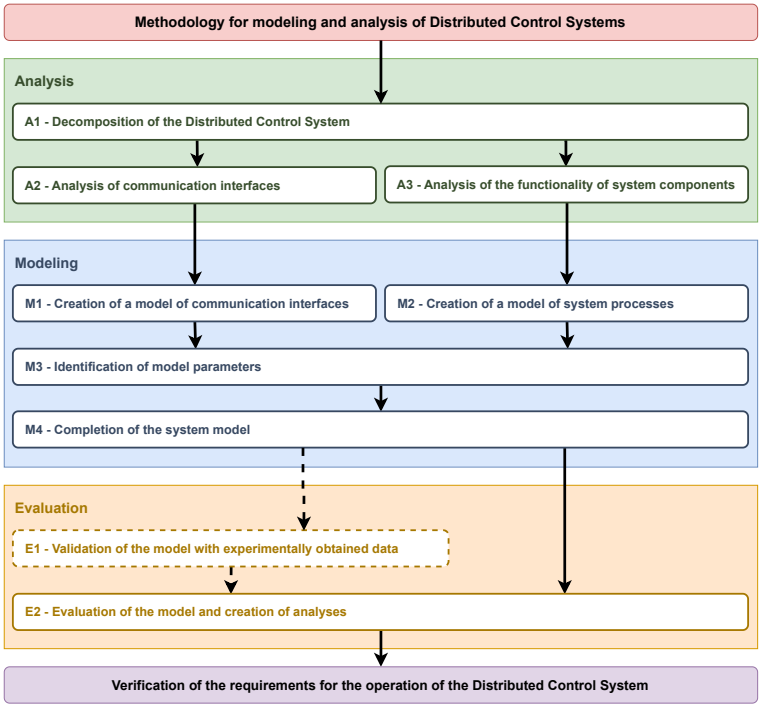

Figure 10. Methodology for modeling and analysis of Distributed Control Systems.

The process of obtaining the model of a DCS is preceded by an analysis of the system in terms of structure, communication interfaces and the functionality of computational and technical processes. This procedure is described by the **Analysis** module of the presented methodology, which is divided into three submodules.

A1—Decomposition of the Distributed Control System—in this submodule, the DCS is broken down into elementary communication interfaces and computational processes that can be modeled independently.
A2—Analysis of communication interfaces—in this submodule, the functionality of communication interfaces is analyzed in terms of data-flow and the principle of their operation.
A3—Analysis of the functionality of system components—in this submodule, the functionality of computational and technical processes is analyzed in terms of data processing complexity.

The second module of the methodology, **Modeling**, is composed of four submodules and describes the creation of models for the system components analyzed in the first module. Models of communication networks are created using Petri nets, and models of computational and technical processes are created in form of Finite-State Automata.

M1—Creation of a model of communication interfaces—in this submodule, models of communication networks are created in form of Colored Timed Petri nets, based on the analysis of functionality performed in the **A2** submodule.
M2—Creation of a model of system processes—in this submodule, models of computational and technical processes are created in form of Finite-State Automata, based on the analysis of functionality performed in the **A3** submodule.
M3—Identification of model parameters—in this submodule, parameters are determined for the created models in terms of data transfer duration or subprocesses execution, based on the analysis performed in submodules **A2** and **A3**, and on the experimentally obtained data.
M4—Completion of the system model—in this submodule, a complex model of the DCS, composed of models of communication interfaces and computational processes, is created and the interconnections of individual models are defined based on the analysis performed in **A1** submodule.

The third and final module of the methodology, **Evaluation**, involves validation of the created model and analysis of the properties of the Distributed Control System. The module is composed of two submodules.

E1—Validation of the model with experimentally obtained data—in this submodule, the resulting DCS model is validated against the experimentally obtained data. This submodule can be applied if the DCS has already been implemented, at least in part.
E2—Evaluation of the model and creation of analyses—in this submodule, the resulting model is used to perform analyses of the DCS, such as determining the throughput and response of the system with respect to various supplied inputs.

The output of the proposed methodology is the analysis of the behavior of the DCS at various inputs, which can be used to optimize the structure of the DCS, determine its limits or modify the designed algorithms to achieve better results. The methodology is next demonstrated in two scenarios based on the infrastructures described in Section 3.

4.1. Scenario 1: ALFRED System Throughput Modeling and Analysis

Scenario 1 deals with the modeling and analysis of the ALFRED System throughput. In this scenario, four units of detector electronics (FEE), two instances of the ALF application, one instance of the FRED server application, and a test client ensuring the generation of sequences of commands for the detector electronics were considered. When creating the model, parallel processing of data in the FRED application, as well as sequential data transmission via the DIM interface are considered [41].

The block diagram of the modeled system can be seen in Figure 11, which defines the decomposition of the modeled system based on the **A1** submodule of the methodology. At the same time, the system model is designed in such a way that it can be expanded horizontally based on the number of parallel branches of the modeled system.

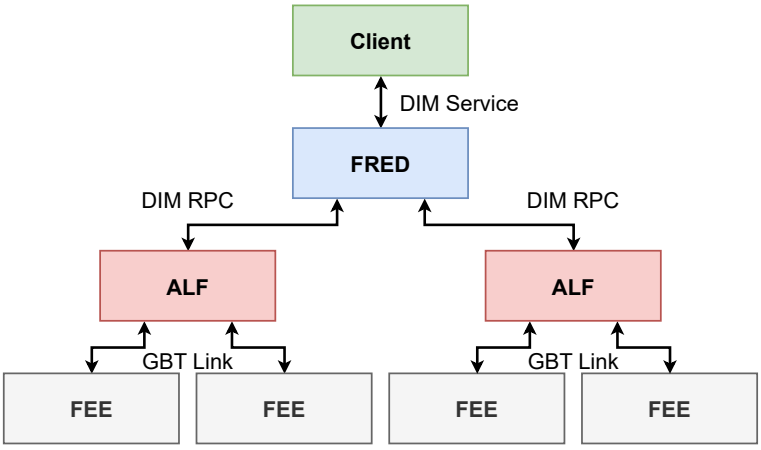

Figure 11. Block diagram of the modeled ALFRED System.

Figure 12 shows an example of a communication interface model created in Scenario 1, which has resulted from the successive application of the **A2** and **M1** submodules of the presented methodology, using the formalism presented in (2) in Section 2.2. The model in the form of a Colored Timed Petri net with the places and transitions denoted as $\{s_1, s_2, \ldots, s_{29}\}$ and $\{t_1, t_2, \ldots, t_{21}\}$, respectively, represents the functionality of the DIM RPC communication interface connecting the FRED application with two ALF applications. The routing of tokens to the respective ALF applications is implemented based on the identification number of the target application stored within the token (parameter d) and the restriction of edge traversability based on the colored properties of the network. The remaining parameters of the token data structure include t as the sequence identification number and r/w as the number of read/write commands. The sequential sending of data over the network between the client and the server is ensured through a loopback in the individual branches

of the model. Transitions representing data transmission over the network (denoted as *) also have an assigned duration which depends on the size of transmitted packets and is determined according to the **M3** submodule of the presented methodology, where the size of the transmitted packet is stored within the token data structure.

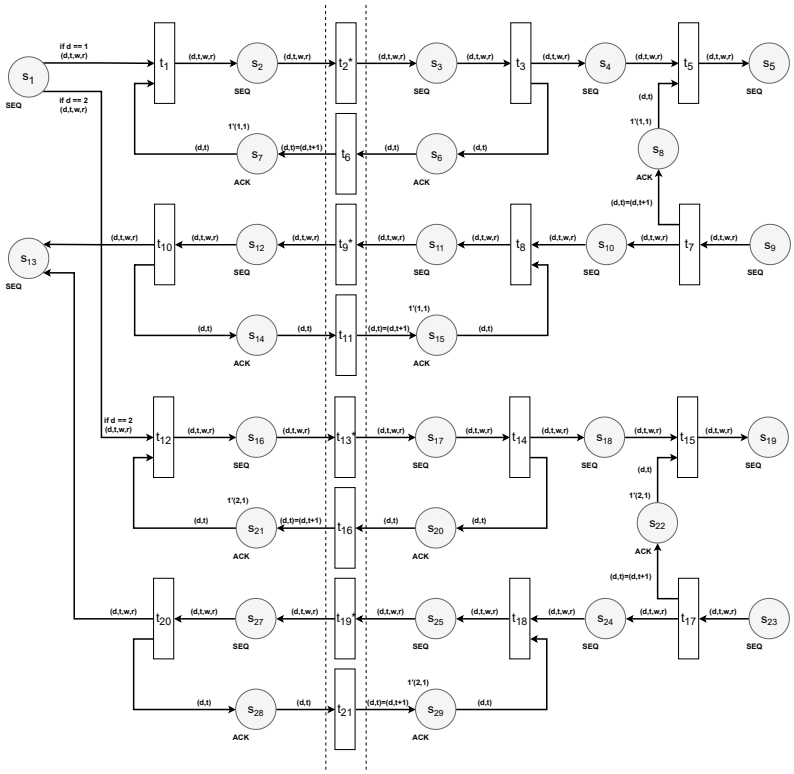

Figure 12. The model of data transfer through the DIM RPC interface between the client and two servers in the form of a Petri net (based on the formalism declared in (2) in Section 2.2).

In Scenario 1, the computational and technical processes of the ALFRED System are modeled in the form of Finite-State Automata, consecutively based on the **A3** and **M2** submodules of the methodology. The elements of the resulting models were determined based on the formalism presented in (1) in Section 2.2. As an example, we show the model of the communication queue functionality of the FRED application, whose graphic representation is shown in Figure 13. The communication queue of the FRED application ensures data processing and communication with one detector electronics unit; several communication queues run in the FRED application in parallel, depending on the number of serviced units. The Finite-State Automaton of the communication queue of the FRED application contains several states, $q_i, i = 1, \ldots, 8$, representing different stages of processing and forwarding of command sequences, from receiving a request from a supervisory system, to generating sequences, sending requests to the ALF application, to processing responses to sequences and generating a response for the supervisory system. At the same time, the transition functions $R(q_i, \sigma_j), i = 1, \ldots, 8, j = 1, \ldots, 12$, which define the movement between the states of the automaton based on the selected state-input pair, have an assigned transition duration according to the **M3** submodule based on the number of commands in the sequence, which is the emulated time required for data processing by the FRED application.

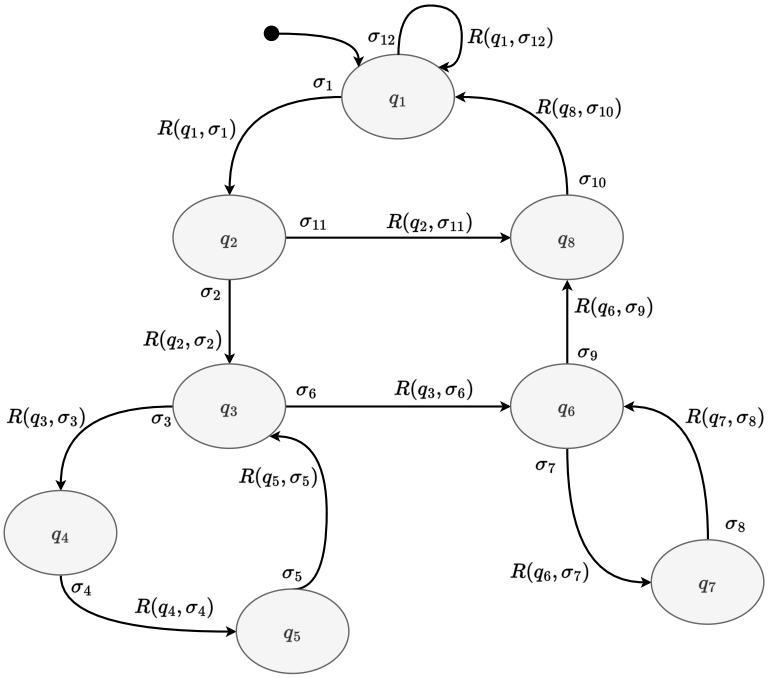

Figure 13. Model of the functionality of the communication queue of the FRED application in the form of a Finite-State Automaton (based on the formalism declared in (1) in Section 2.2).

Figure 14 depicts how the created models of communication interfaces and computational processes are interconnected into the resulting model of the ALFRED System, created in the **M4** submodule. The connection of individual models (in the form of Petri nets and Finite-State Automata) is realized by a mechanism where a token at the output point of the Petri net results in the activation of the input of the Finite-State Automaton, and vice versa, i.e., the activation of the input of the Finite-State Automaton results in the addition of a token into the entry point of the corresponding Petri net model. The resulting model was implemented in the MATLAB/Simulink environment using the Stateflow tool with the purpose of using it to perform simulations with different input data.

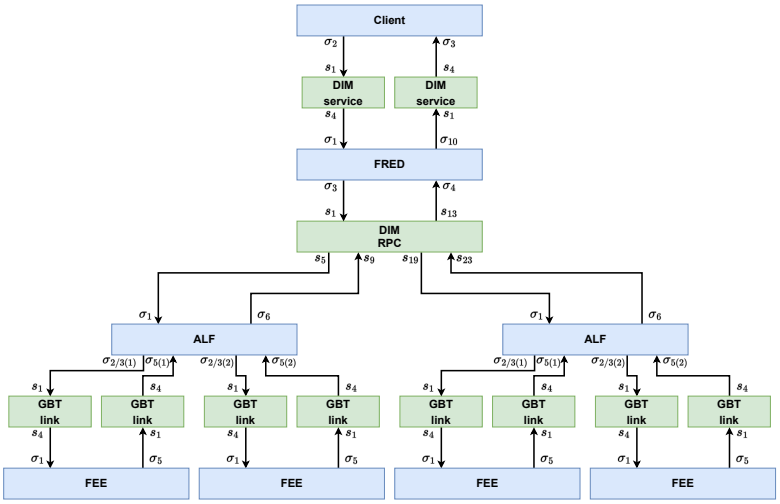

Figure 14. Linking subsystem models within the ALFRED System model.

Figure 15 shows a comparison of the duration of execution of command sequences for detector electronics on four parallel links based on a simulation model with experimentally obtained data, which was performed based on the **E1** submodule of the presented methodology. The created model shows the mean absolute percentage error $MAPE_{ALFRED} = 11.36\%$

and the coefficient of determination $R^2_{ALFRED} = 0.9997$, so it can be concluded that the model correctly mirrors the behavior of the real system.

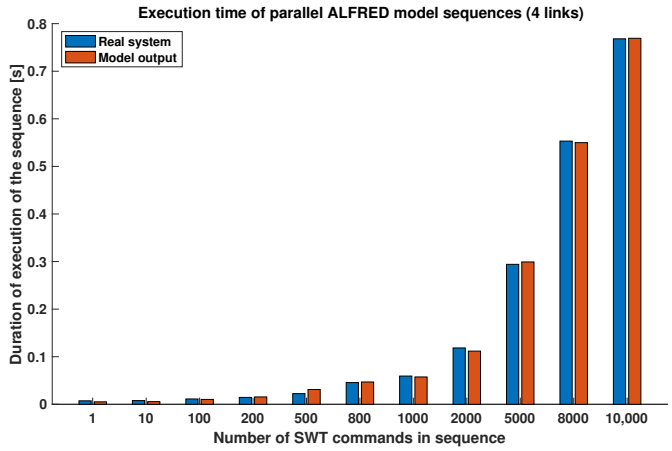

Figure 15. Comparison of the response of the parallel ALFRED System model with experimentally obtained data.

Using the created model, it was also possible to obtain a prediction of the maximum throughput of the parallel ALFRED System according to the **E2** submodule of the proposed methodology. The maximum throughput was calculated depending on the number of commands for the detector electronics within one sequence ranging from 1 to 20,000. The maximum throughputs obtained from the simulation model can be seen in Figure 16, which shows a comparison of the duration of parallel and sequential execution of sequences on four links. The highest throughput of the system is achieved with sequences of 2000 commands, worth approximately 17,900 commands per second for each link. For comparison, when executing commands sequentially, it is possible to achieve a maximum throughput of about 5500 commands per second on four links.

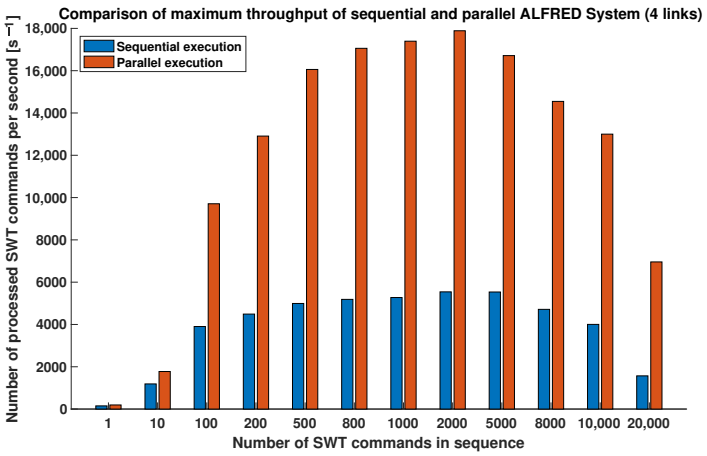

Figure 16. Comparison of throughputs of sequential and parallel ALFRED Systems based on simulations using the created models.

4.2. Scenario 2: Modeling and Analyzing the Response of a Mobile Robotics Application

Scenario 2 is dedicated to applying the presented methodology to the mobile robotics setup based on MiroSot mobile robots in the context of a DCS at the CMCT&II at DCAI FEEI TU in Košice.

Based on the **A1** submodule, the decomposition of the modeled process was performed: the modeled application is composed of four MiroSot mobile robots that communicate

with the Comm (communication) Module running on the supervisory computer through the Bluetooth SPP interface [49], as shown in Figure 17. The Comm Module ensures the translation of messages between the ROS and Bluetooth SPP interfaces, where each MiroSot mobile robot is assigned one of the parallel running communication queues [55] within the Comm Module. The Control Module can be used to generate trajectories for mobile robots, but in terms of Scenario 2, the Control Module is considered a test client that generates messages for mobile robots.

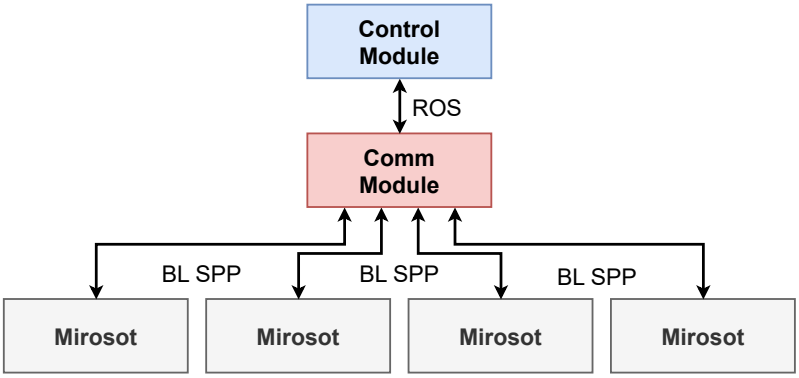

Figure 17. Block diagram of the modeled mobile robotics application.

Figure 18 shows a model of data transfer through the ROS interface in the form of a Colored Timed Petri net created based on **A2** and **M1** submodules of the proposed methodology. It is an example of a communication interface model created in Scenario 2 using the formalism presented in (2) in Section 2.2, with the places and transitions denoted as $\{s_1, s_2, \ldots, s_6\}$ and $\{t_1, t_2, \ldots, t_5\}$, respectively. In addition to the identification number of the message (parameter t), the transmitted tokens contain information about the size of the data transmitted through the interface (parameter s), as well as the identification number of the target robot (parameter d), based on which the messages are routed within the Comm Module. The loopback in the model represents the confirmation of received messages, since the ROS interface uses the TCP protocol to transfer data over the network. At the same time, transitions representing data transmission over the network have an assigned duration based on the size of the data being transmitted and on the experimentally obtained information on how the duration of the transmission depends on the size of the transmitted message, which is performed based on the **M3** submodule of the presented methodology.

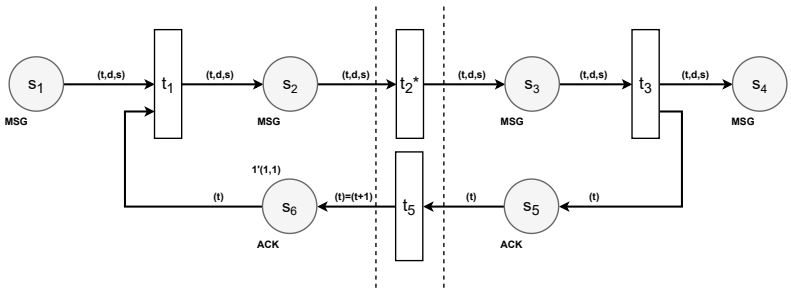

Figure 18. The model of data transfer through the ROS interface in the form of a Petri net (based on the formalism declared in (2) in Section 2.2).

As an example of a model of computational processes in Scenario 2, we show the model of the functionality of the Communication Module queue in the form of a Finite-State Automaton shown in Figure 19, which is created based on the **A3** and **M2** submodules of the presented methodology. Models of computational processes were created in the form of Finite-State Automata according to (1) in Section 2.2. The states $q_i, i = 1, \ldots, 4$ of the Finite-State Automata represent the stages of data forwarding between the Control Module and the MiroSot mobile robot, such as receiving a request from the Control Module,

translating the request, sending the request to the mobile robot, or generating a response for the supervisory system. The transition functions $R(q_i, \sigma_j), i = 1, \ldots, 4, j = 1, \ldots, 6$ between individual states, based on the selected state–input pairs, have an assigned duration based on the size of the transmitted messages, while the dependence between the time required to process the message and the size of the transmitted messages was determined based on experimentally obtained data within the **M3** submodule of the presented methodology.

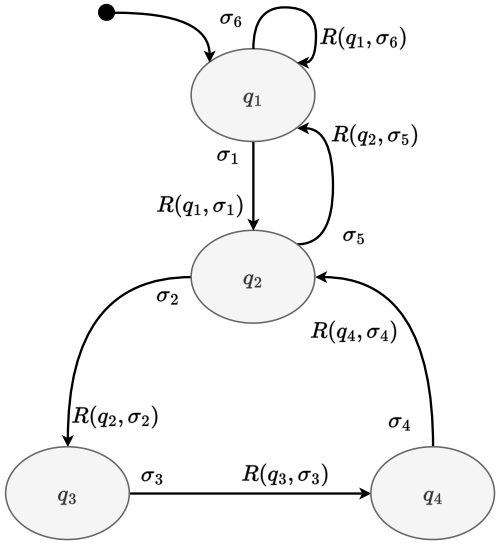

Figure 19. Model of the functionality of the queue of the Communication Module in the form of a Finite-State Automaton (based on the formalism declared in (1) in Section 2.2.

Figure 20 shows how the communication interface models (in the form of Petri nets) and computing processes (in the form of Finite-State Automata) are connected to form the complex model according to the **M4** submodule, which is realized by the same mechanism as in Scenario 1. The resulting model was implemented in the MATLAB/Simulink environment using the MATLAB Stateflow tool to determine the properties of the modeled DCS with respect to various input data.

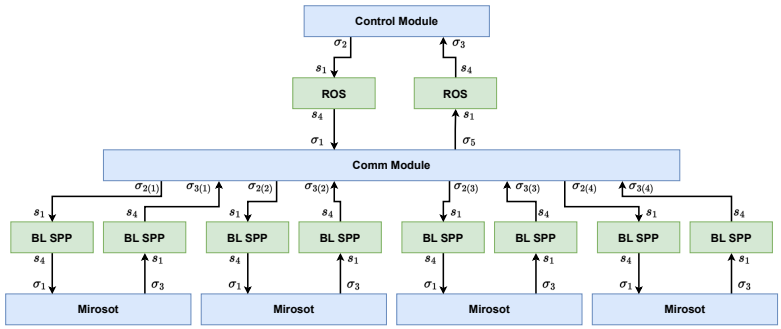

Figure 20. Connection of subsystem models within the mobile robotics application model.

In Figure 21, we can see a comparison of the response of the real system and the results of simulations using the created model for messages of different sizes forwarded between the Control Module and MiroSot mobile robots according to the **E1** submodule of the presented methodology. Validation of the model includes parallel communication with four mobile robots, where processes that are executed sequentially are also considered in the output of the model. Various message sizes from 1 to 2000 bytes were used in the experiment. From a statistical point of view, the created model shows the mean absolute percentage error $MAPE_{RAPP} = 7.45\%$ and the coefficient of determination $R^2_{RAPP} = 0.9994$, which means that the model the behavior of the real robotic application system is appropriately mirrored.

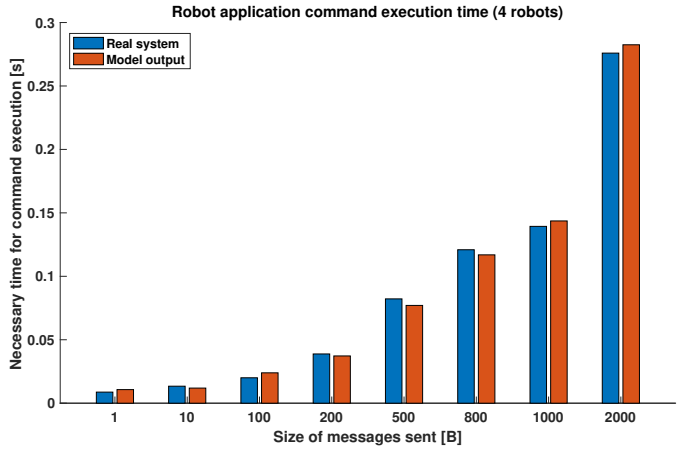

Figure 21. Comparison of the response of the robotic application model with experimentally obtained data.

Based on the **E2** submodule, the implemented model can also be used to determine the maximum throughput of the modeled system in order to determine the optimal size of the transmitted data to achieve the highest possible throughput. The sizes of sent messages from 1 to 10,000 bytes were used as model inputs, where Figure 22 shows the dependence of the maximum throughput of the system on the size of transmitted messages. As can be seen, the robotic application system using four units of MiroSot mobile robots achieves the highest throughput when sending messages of size 2000 bytes, when the throughput is approximately 7000 bytes per second.

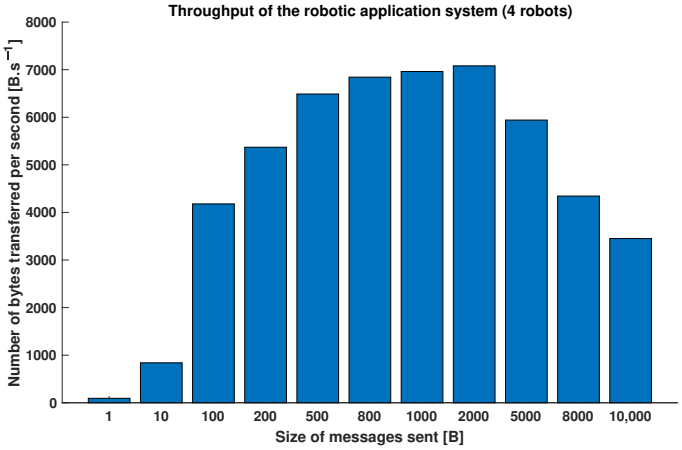

Figure 22. Throughput of the robotic application system based on simulations using the created model.

The results of these simulations can be used in the design of the composition of sent messages and control algorithms at the supervisory level to achieve the highest possible performance and reliability of the system. At the same time, it is possible to assess the suitability of the implemented control algorithm with regard to the possible minimum sampling period, which is based on the predicted response and throughput of the modeled system.

5. Conclusions

This paper presents the proposed methodology for modeling and analysis of Distributed Control Systems. The methodology is composed of three modules, each of which is divided into several submodules. The methodology includes steps for the analysis of communication interfaces and computational processes, and presents the method of

creating models of DCS components in the form of Colored Timed Petri nets and Finite-State automata. The result of the applied methodology is the analysis of the DCS, such as predictions of throughput and response of the system with respect to various input factors. The models created by applying this methodology can also be used to optimize the structure or functionality of the investigated DCS. Various optimization methods can be used to optimize the structure of the investigated system, where evolutionary algorithms can be mentioned as an example.

The presented methodology was demonstrated and verified on two scenarios: first of these was devoted to the application of the methodology onto the ALFRED distributed system, which is a subsystem of the control system of the detectors of the ALICE experiment at CERN, and the second one dealing with the application of mobile robotics in the context of the DCS at the CMCT&II at DCAI FEEI TU in Košice. Thanks to the universality of the proposed methodology, it can be applied to Distributed Control Systems with different structures. The created models have been implemented in the MATLAB/Simulink environment using the Stateflow tool. By performing simulations using created models, it is possible to predict the behavior of the system at different inputs.

FSM models are used in the ALICE experiment at the SCADA/HMI system level for the purpose of monitoring and control of the detectors. However, there was no methodology for determining the distribution of individual processes within the system, since this distribution was only determined experimentally. The proposed methodology enables the existing models to be extended by the behavior of communication interfaces, thanks to which it is possible to determine the appropriate distribution of software and hardware resources within the system based on the system throughput requirements. The results obtained by applying the proposed methodology were used during the modification of the FRED system, which was first tested at the development workplace created as part of the project ALICE experiment at the CERN LHC and then implemented within the Detector Control System of the ALICE experiment.

Author Contributions: Conceptualization, M.T. and J.J.; methodology, M.T. and A.J.; software, M.T. and T.T.; validation, M.T., S.J. and T.T.; formal analysis, M.T. and S.J.; investigation, J.J.; resources, M.T. and T.T.; data curation, M.T.; writing—original draft preparation, M.T.; writing—review and editing, M.T., S.J. and A.J.; visualization, M.T. and T.T.; supervision, J.J., A.J. and S.J.; project administration, J.J.; funding acquisition, J.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the project ALICE experiment at the CERN LHC: The study of strongly interacting matter under extreme conditions (ALICE TUKE 0410/2022 (2022–2026)).

Data Availability Statement: Data are contained within the article.

Acknowledgments: This work was supported by the Slovak Research and Development Agency under the contract No. APVV-19-0590 and by the project KEGA 022TUKE-4/2023 granted by the Ministry of Education, Science, Research and Sport of the Slovak Republic. This work is also the result of project implementation: ALICE experiment at the CERN LHC: The study of strongly interacting matter under extreme conditions (ALICE TUKE 0410/2022 (2022–2026)).

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

Abbreviations

The following abbreviations are used in this manuscript:

$u(\cdot)$	input of the system with continuous dynamics
$\sigma(\cdot)$	input of the system with discrete dynamics
$y(\cdot)$	output of the system with continuous dynamics
$w(\cdot)$	output of the system with discrete dynamics
$x(\cdot)$	state of the system with continuous dynamics
$q(\cdot)$	state of the system with discrete dynamics

X	set of states of the system with continuous dynamics
Q	set of states of the system with discrete dynamics
Σ	set of inputs to the system with discrete dynamics
R	set of transitions of the finite state automaton
S	set of places in the Petri net
T	set of transitions in the Petri net
$s(\cdot)$	Petri net place
$t(\cdot)$	Petri net transition
ALF	ALICE Low-Level Front End
ALFRED	ALICE Low-Level Front End Device
ALICE	A Large Ion Collider Experiment
ALICE-DCS	ALICE Detector Control System
CANALF	CANbus ALICE Low-Level Front End
CERN	Conseil Européen pour la Recherche Nucléaire (European Organization for Nuclear Research)
CMCT&II	Center of Modern Control Techniques and Industrial Informatics
CPS	Cyber-Physical System
DCS	Distributed Control System
DIM	Distributed Information Management System
FRED	Front End Device
GBT	GigaBit Transceiver
HMI	Human–Machine Interface
ITS	Inner Tracking System
LHC	Large Hadron Collider
RPC	Remote Procedure Call
SCADA	Supervisory Control And Data Acquisition
SWT	Single Word Transaction (communication protocol)
WinCC OA	SCADA and HMI system from Siemens

References

1. Zhang, X.M.; Han, Q.L.; Ge, X.; Ding, D.; Ding, L.; Yue, D.; Peng, C. Networked control systems: A survey of trends and techniques. *IEEE/CAA J. Autom. Sin.* **2019**, *7*, 1–17. [CrossRef]
2. Tomura, T.; Uehiro, K.; Kanai, S.; Yamamoto, S. Developing simulation models of open distributed control system by using object-oriented structural and behavioral patterns. In Proceedings of the Fourth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, ISORC 2001, Magdeburg, Germany, 2–4 May 2001; pp. 428–437.
3. Koziorek, J. Design of Distributed Control Systems Based on New International Standards. *IFAC Proc. Vol.* **2004**, *37*, 313–318. [CrossRef]
4. Cruz, E.M.; Carrillo, L.R.G.; Salazar, L.A.C. Structuring Cyber-Physical Systems for Distributed Control with IEC 61499 Standard. *IEEE Lat. Am. Trans.* **2023**, *21*, 251–259. [CrossRef]
5. Ge, X.; Yang, F.; Han, Q.L. Distributed networked control systems: A brief overview. *Inf. Sci.* **2017**, *380*, 117–131. [CrossRef]
6. Himrane, O.; Ourghanlian, A.; Amari, S. Response time evaluation of industrial-scale distributed control systems by discrete event systems formalisms. *Int. J. Control* **2022**, *95*, 419–431. [CrossRef]
7. Yang, Y.; Zhou, X. Cyber-physical systems modeling based on extended hybrid automata. In Proceedings of the 2013 International Conference on Computational and Information Sciences, Shiyang, China, 21–23 June 2013; pp. 1871–1874.
8. Sanfelice, R.G. Analysis and design of cyber-physical systems. A hybrid control systems approach. In *Cyber-Physical Systems: From Theory to Practice*; CRC Press: Boca Raton, FL, USA, 2016; pp. 1–29.
9. Oumeziane, F.A.; Ourghanlian, A.; Amari, S. Analysis of distributed control systems using timed automata with guards and dioid algebra. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; Volume 1, pp. 1373–1376.
10. Li, J.; Wang, Z.; Sun, L.; Wang, W. Modeling and analysis of network control system based on hierarchical coloured Petri net and Markov chain. *Discret. Dyn. Nat. Soc.* **2021**, *2021*, 9948855. [CrossRef]
11. Baldellon, O.; Fabre, J.C.; Roy, M. Modeling distributed real-time systems using adaptive petri nets. *Actes 1re J.* **2011**, *10*, 7–8.
12. ALICE Collaboration. Performance of the ALICE experiment at the CERN LHC. *Int. J. Mod. Phys. A* **2014**, *29*, 1430044. [CrossRef]
13. Augustinus, A.; Chochula, P.; Jirdén, L.; Lechman, M.; Rosinský, P.; Pinazza, O.; De Cataldo, G.; Kurepin, A.; Moreno, A. Computing architecture of the ALICE detector control system. In Proceedings of the 13th International Conference on Accelerator and Large Experimental Physics Control Systems—ICALEPCS 2011, Grenoble, France, 10–14 October 2011; pp. 156–158.
14. Bernardini, M.; Foraz, K. Long shutdown 2@ lhc. *CERN Yellow Rep.* **2016**, *2*, 290.

15. Jadlovský, J.; Jadlovská, A.; Jadlovská, S.; Oravec, M.; Vošček, D.; Kopčík, M.; Čabala, J.; Tkáčik, M.; Chochula, P.; Pinazza, O. Communication architecture of the Detector Control System for the Inner Tracking System. In Proceedings of the 16th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2017), Barcelona, Spain, 8–13 October 2017; p. THPHA208. [CrossRef]
16. Chochula, P.; Augustinus, A.; Kurepin, A.; Lechman, M.; Pinazza, O.; Rosinský, P.; Kurepin, A.N.; Pinazza, O. Operational experience with the ALICE Detector Control System. In Proceedings of the 14th International Conference on Accelerator & Large Experimental Physics Control Systems, ICALEPCS, San Francisco, CA, USA, 6–11 October 2013.
17. Jadlovská, A.; Jadlovská, S.; Vošček, D. Cyber-physical system implementation into the distributed control system. *IFAC-PapersOnLine* **2016**, *49*, 31–36. [CrossRef]
18. Jadlovský, J.; Čopík, M.; Papcun, P. *Distributed Control Systems (In Slovak: Distribuované Systémy Riadenia)*; Elfa: Košice, Slovakia, 2013.
19. Frey, G.; Hussain, T. Modeling techniques for distributed control systems based on the IEC 61499 standard-current approaches and open problems. In Proceedings of the 2006 8th International Workshop on Discrete Event Systems, Ann Arbor, MI, USA, 10–12 July 2006; pp. 176–181.
20. Luder, A.; Hundt, L.; Biffl, S. On the suitability of modeling approaches for engineering distributed control systems. In Proceedings of the 2009 7th IEEE International Conference on Industrial Informatics, Cardiff, UK, 23–26 June 2009; pp. 440–445.
21. Jazdi, N. Cyber physical systems in the context of Industry 4.0. In Proceedings of the 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca, Romania, 22–24 May 2014; pp. 1–4.
22. Rajkumar, R.; Lee, I.; Sha, L.; Stankovic, J. Cyber-physical systems: The next computing revolution. In Proceedings of the Design Automation Conference, Anaheim, CA, USA, 13–18 June 2010; pp. 731–736.
23. Holecko, P. Overview of distributed control systems formalisms. *Adv. Electr. Electron. Eng.* **2011**, *7*, 253–256.
24. IEC 62264-1:2013 Enterprise-Control System Integration—Part 1: Models and Terminology. Available online: https://webstore.iec.ch/publication/6675 (accessed on 8 October 2023).
25. Martinez, E.M.; Ponce, P.; Macias, I.; Molina, A. Automation pyramid as constructor for a complete digital twin, case study: A didactic manufacturing system. *Sensors* **2021**, *21*, 4656. [CrossRef] [PubMed]
26. Apilioğulları, L. Digital transformation in project-based manufacturing: Developing the ISA-95 model for vertical integration. *Int. J. Prod. Econ.* **2022**, *245*, 108413. [CrossRef]
27. Jadlovský, J.; Jadlovská, A.; Jadlovská, S.; Čerkala, J.; Kopčík, M.; Čabala, J.; Oravec, M.; Varga, M.; Vošček, D. Research activities of the center of modern control techniques and industrial informatics. In Proceedings of the 2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 21–23 January 2016; pp. 279–285.
28. Van Der Schaft, A.J.; Schumacher, J.M. *An Introduction to Hybrid Dynamical Systems*; Springer: London, UK, 2000; Volume 251.
29. Lunze, J.; Lamnabhi-Lagarrigue, F. *Handbook of Hybrid Systems Control: Theory, Tools, Applications*; Cambridge University Press: Cambridge, UK, 2009.
30. Hopcroft, J.E.; Motwani, R.; Ullman, J.D. Introduction to automata theory, languages, and computation. *ACM Sigact News* **2001**, *32*, 60–65. [CrossRef]
31. Reisig, W. *Petri Nets: An Introduction*; Springer: Berlin/Heidelberg, Germany, 2012; Volume 4.
32. Jamro, M.; Rzonca, D.; Rząsa, W. Testing communication tasks in distributed control systems with SysML and Timed Colored Petri Nets model. *Comput. Ind.* **2015**, *71*, 77–87. [CrossRef]
33. Moreno, R.P.; Tardioli, D.; Salcedo, J.L.V. Distributed implementation of discrete event control systems based on Petri nets. In Proceedings of the 2008 IEEE International Symposium on Industrial Electronics, Cambridge, UK, 30 June–2 July 2008; pp. 1738–1745.
34. Ghanaim, A.; Borges, G.A.; Frey, G. Estimating delays in networked control systems using colored Petri nets and Markov chain models. In Proceedings of the 2009 IEEE Conference on Emerging Technologies & Factory Automation, Palma de Mallorca, Spain, 22–25 September 2009; pp. 1–6.
35. Louis, B.D.S.; Alain, O.; Saïd, A. Delays Evaluation of Networked Control System Switches using Timed Coloured Petri Nets and Formal Series. In Proceedings of the 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), Sinaia, Romania, 12–15 September 2023; pp. 1–8.
36. Aamodt, K.; Quintana, A.A.; Achenbach, R.; Acounis, S.; Adamová, D.; Adler, C.; Aggarwal, M.; Agnese, F.; Rinella, G.A.; Ahammed, Z.; et al. The ALICE experiment at the CERN LHC. *J. Instrum.* **2008**, *3*, S08002. [CrossRef]
37. Chochula, P.; Jirden, L.; Augustinus, A.; De Cataldo, G.; Torcato, C.; Rosinsky, P.; Wallet, L.; Boccioli, M.; Cardoso, L. The ALICE detector control system. *IEEE Trans. Nucl. Sci.* **2010**, *57*, 472–478. [CrossRef]
38. Huang, J.; Saiz, P.; Betev, L.; Carminati, F.; Grigoras, C.; Schreiner, S.; Zhu, J. Grid Architecture and implementation for ALICE experiment. In Proceedings of the 16th International Conference on Advanced Communication Technology, Pyeongchang, Republic of Korea, 16–19 February 2014; pp. 253–261.
39. Moreira, P.; Ballabriga, R.; Baron, S.; Bonacini, S.; Cobanoglu, O.; Faccio, F.; Fedorov, T.; Francisco, R.; Gui, P.; Hartin, P.; et al. The GBT Project. In Proceedings of the Topical Workshop on Electronics for Particle Physics, Paris, France, 21–25 September 2009; pp. 342–346. [CrossRef]
40. Gaspar, C.; Dönszelmann, M.; Charpentier, P. DIM, a portable, light weight package for information publishing, data transfer and inter-process communication. *Comput. Phys. Commun.* **2001**, *140*, 102–109. [CrossRef]

41. Tkáčik, M.; Jadlovský, J.; Jadlovská, S.; Koska, L.; Jadlovská, A.; Donadoni, M. FRED—Flexible Framework for Frontend Electronics Control in ALICE Experiment at CERN. *Processes* **2020**, *8*, 565. [CrossRef]
42. Gaspar, C.; Franek, B. Tools for the automation of large distributed control systems. *IEEE Trans. Nucl. Sci.* **2006**, *53*, 974–979. [CrossRef]
43. De Cataldo, G.; Augustinus, A.; Boccioli, M.; Chochula, P.; Jirdén, L.S. Finite state machines for integration and control in ALICE. In Proceedings of the ICALEPCS07, Knoxville, TN, USA, 15–19 October 2007.
44. Chochula, P.; Augustinus, A.; Bond, P.; Kurepin, A.; Lechman, M.; Lã, J.; Pinazza, O. Challenges of the ALICE Detector Control System for the LHC RUN3. In Proceedings of the 16th International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPCS, Barcelona, Spain, 8–13 October 2017.
45. Santos, J.M.; Portugal, D.; Rocha, R.P. An evaluation of 2D SLAM techniques available in robot operating system. In Proceedings of the 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Linköping, Sweden, 21–26 October 2013; pp. 1–6.
46. Sechenev, S.; Ryadchikov, I.; Gusev, A.; Lampezhev, A.; Nikulchev, E. Development of a design methodology for cloud distributed control systems of mobile robots. *J. Sens. Actuator Netw.* **2021**, *11*, 1. [CrossRef]
47. Agrawal, S.; Jain, S.K.; Ibeke, E. An orchestrator for networked control systems and its application to collision avoidance in multiple mobile robots. *Int. J. Eng. Syst. Model. Simul.* **2021**, *12*, 103–110. [CrossRef]
48. Jadlovský, J.; Kopčík, M. Distributed control system for mobile robots with differential drive. In Proceedings of the 2016 Cybernetics & Informatics (K&I), Levoca, Slovakia, 2–5 February 2016; pp. 1–5.
49. Jadlovská, A.; Jadlovský, J.; Jadlovská, S.; Čerkala, J.; Kopčík, M.; Čabala, J.; Oravec, M.; Varga, M.; Vošček, D.; Tkáčik, M.; et al. Proposal of a methodology for modeling, control, simulation and non-destructive diagnosis of mobile robots (In Slovak: Návrh metodiky pre modelovanie, riadenie, simuláciu a nedeštruktívnu diagnostiku mobilných robotov). *Strojárstvo/Strojírenství Eng. Mag.* **2017**, *XXI* , 1–9.
50. Tkáčik, M.; Březina, A.; Jadlovská, S. Design of a Prototype for a Modular Mobile Robotic Platform. *IFAC-PapersOnLine* **2019**, *52*, 192–197. [CrossRef]
51. Sahraoui, Z.; Labed, A. Methodology for fast prototyping of distributed real-time systems. In Proceedings of the 2022 5th International Symposium on Informatics and its Applications (ISIA), M'sila, Algeria, 29–30 November 2022; pp. 1–6.
52. Lora, M.; Muradore, R.; Reffato, R.; Fummi, F. Simulation alternatives for modeling networked cyber-physical systems. In Proceedings of the 2014 17th Euromicro Conference on Digital System Design, Verona, Italy, 27–29 August 2014; pp. 262–269.
53. Imama, K.G.; Ourghanlian, A.; Amari, S. Modeling Distributed Control Systems response time: From theory to measures. In Proceedings of the 2020 IEEE 16th International Conference on Control & Automation (ICCA), Singapore, 9–11 October 2020; pp. 696–701.
54. dit Sollier, L.B.; Ourghanlian, A.; Amari, S. Coloured Petri Nets for Temporal Performance Evaluation of Distributed Control Systems—Application to a FIFO Queue. *IEEE Robot. Autom. Lett.* **2022**, *7*, 11268–11274. [CrossRef]
55. Tkáčik, M. Methods and Tools for Design, Modeling and Realization of Distributed Control Systems of Large Physical Experiments (In Slovak: Metódy a Prostriedky Pre Návrh, Modelovanie a Realizáciu Distribuovaných Systémov Riadenia Veľkých Fyzikálnych Experimentov). Ph.D. Thesis, Department of Cybernetics and Artificial Inteligence, Technical University of Košice, Košice, Slovakia, 2023.