CYBPHYS

MODEL-ORIENTED CONTROL IN INTELLIGENT MANUFACTURING SYSTEMS

CHERNIHIV POLYTECHNIC NATIONAL UNIVERSITY

CYBPHYS

Co-funded by the Erasmus+ Programme of the European Union

Model-Oriented Control in Intelligent Manufacturing Systems

RTU Press

Riga 2022

Model-Oriented Control in Intelligent Manufacturing Systems. Riga RTU Press, 2022. – 260 p.

The purpose of this e-book is to serve as a textbook for graduate and postgraduate students in the field of computer engineering during the study of disciplines related to software control, modeling of production conditions and their environment. It may also be of interest to specialists in planning and assessing the quality of the production process, as well as to specialists and engineers in the field of industrial robot manufacturing.

Language Editor Oksana Ivanova

Layout Design Ģirts Semevics

Published by RTU Press
Phone: +37167089123
e-mail: izdevnieciba@rtu.lv

ISBN 978-9934-22-674-8 (pdf)

Acknowledgements

The textbook has been developed by the financial support of the European Union. The authors from Chernihiv Polytechnic National University, V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine and Riga Technical University are responsible for the content of this document. This publication reflects the views only of the authors, and it cannot be regarded as the European Union’s official position.

The textbook has been developed in a frame of the project “ERASMUS+ Capacity-building in the Field of Higher Education 2019.

It has been co-funded by the project “Development of Practically-Oriented Student-Centered Education in the Field of Modeling of Cyber-Physical Systems”, CybPhys (609557-EPP-1-2019-1-LV-EPPKA2-CBHE-JP – ERASMUS+ CBHE).

The textbook is intended for students of computer engineering and industrial automation, as well as electrical engineering specialties. It can also be useful for students and professionals focusing on innovation management issues.

Key Action: KA2 – Cooperation for innovation and the exchange of good practices
Action: Capacity Building in Higher Education
Action Type: Joint Projects
Deliverable: 2.3. E-book of five electronic courses by the declared directions

Co-funded by the Erasmus+ Programme of the European Union

Annotation

The use of computer models in Intelligent Manufacturing Systems (IMSs) is the main distinguishing feature that ensures their development as Cyber-Physical Systems (CPSs). The purpose of this e-book is to serve as a textbook for graduate and postgraduate students in the field of computer engineering during the study of disciplines related to software control, modeling of production conditions and their environment. The e-book may also be of interest to specialists in planning and assessing the quality of the production process, as well as to specialists and engineers in the field of industrial robot manufacturing.

The e-book includes seven chapters, starting with the analysis of the features of the CPSs. Further chapters consider the theoretical issues related to the construction of models for the implementation of control algorithms based on the modification of Petri nets, forecasting models using one of the varieties of temporal logic and recovery models that provide the creation of virtual images and images of the control object and its environment. The tools for creating the considered models and their integration into the control loop are also described. Examples of the practical application of the Model-Oriented Control (MOC) methods in the creation of IMSs and planning of their activities are given.

Anotācija

Datormodeļu izmantošana viedās ražošanas sistēmās (*IMS*) ir noteicoša pazīme, kas nodrošina to kā kiberfizisko sistēmu (CPS) attīstību. Šī elektroniskā grāmata ir domāta kā mācību līdzeklis datortehnikas nozares maģistrantiem, apgūstot priekšmetus, kas saistīti ar programmatūras vadību, ražošanas apstākļu un to vides modelēšanu. E-grāmata var būt noderīga arī speciālistiem, kas iesaistīti ražošanas procesu kvalitātes plānošanā un novērtēšanā, kā arī inženieriem, kuri darbojas industriālo robotu ražošanas jomā.

E-grāmatā ir septiņas nodaļas. Grāmatas sākumā ir analizētas CPS iezīmes. Turpmākajās nodaļās aplūkoti teorētiskie jautājumi, kas saistīti ar modeļu konstruēšanu vadības algoritmu ieviešanai, pamatojoties uz Petri tīklu modifikācijām, prognozēšanas modeļiem, izmantojot kādu no laika loģikas veidiem un atkopšanas modeļiem, kas nodrošina kontroles objektu un to vides virtuālo attēlu izveidi. Aprakstīti arī rīki aplūkoto modeļu izveidei un to integrācijai vadības sistēmās. Tiek sniegti piemēri modeļorientētās kontroles (MOC) metožu praktiskai lietošanai *IMS* izveidē un to darbības plānošanā.

Contributors

Volodymyr Kazymyr, Professor, Information and Computer Systems Department, Chernihiv Polytechnic National University, 95 Shevchenko Str., Chernihiv, 14035, Ukraine, vvkazymyr@gmail.com

Oleg Novomlynets, Professor, Rector, Chernihiv Polytechnic National University, 95 Shevchenko Str., Chernihiv, 14035, Ukraine, oon1@ukr.net

Sergey Ivanets, Associate Professor, Institute of Electronics and Information Technologies, Chernihiv Polytechnic National University, 95 Shevchenko Str., Chernihiv, 14035, Ukraine, sergey.ivanets@gmail.com

Oleksandr Palagin, Professor**,** Academician of the National Academy of Sciences of Ukraine, V.M. Glushkov Institute of Cybernetics, 40 Academician Glushkov Avenue, Kyiv, 03187, Ukraine, incyb@incyb.kiev.ua

Volodymyr Opanasenko, Professor**,** V.M. Glushkov Institute of Cybernetics, 40 Academician Glushkov Avenue, Kyiv, 03187, Ukraine, vlopanas@ukr.net

Nadezhda Kunicina, Professor, Senior Researcher of the Division of Industrial Electronic Equipment, Institute of Industrial Electronics and Electrical Engineering, Faculty of Power and Electrical Engineering, Riga Technical University, 12/1 Azenes Str. – 503, Riga, LV1048, Latvia, nadezda.kunicina@rtu.lv

Anatolijs Zabasta, Senior Researcher of the Institute of Industrial Electronics and Electrical Engineering, Faculty of Power and Electrical Engineering, Riga Technical University, 12/1 Azenes Str. – 503, Riga, LV1048, Latvia, anatolijs.zabashta@rtu.lv

Andrejs Romanovs, Senior Researcher of the Institute of Information Technology, Faculty of Computer Science and Information Technology, Riga Technical University, 2 Daugavgrivas Str. – 429, Riga, LV1048, Latvia, andrejs.romanovs@rtu.lv

Jurijs Merkurjevs, Professor, Senior Researcher of the Institute of Information Technology, Faculty of Computer Science and Information Technology, Riga Technical University, 2 Daugavgrivas Str. – 426, Riga, LV1048, Latvia, jurijs.merkurjevs@rtu.lv

Foreword

Intelligent manufacturing systems (IMSs), which began to emerge at the end of the last century, are now becoming a defining factor in technological development. Continuing the trends in the use of automation tools, laid down by Flexible Production Systems and Computer-Integrated Manufacturing, they brought their own special features associated with the widespread use of computer models directly to control not only the technological process, but also the enterprise as a whole. This became especially evident with the beginning of the implementation of Industry 4.0 strategy, which actually boiled down to the creation of Cyber-Physical Systems (CPSs). Namely, the CPSs externalize all basic ideas of intellectual production. Through the integration of the real physical world and the virtual world created with the help of computers, IMSs provide an incredible increase of production efficiency, their orientation towards the needs of society, and time reduction for introducing new technology achievements.

Everything that CPSs bring to the production sphere is based on the use of computer models, which cease to be just an element of designing new systems, but become separate components of the manufacturing process and, above all, in the field of control. This applies to both the control of technological processes and management at the level of planning production activities and ensuring their quality. Thus, the use of computer models in the context of control tasks fully justifies the name of CPS.

It is important to note that the use of computer modeling as an approach to managing manufacturing processes does not take away all the existing achievements of traditional methods, but make them more intelligent and smart. This is important from the point of view of continuity in methods and technologies which have proven them through long-term approbation in practice.

At the same time, the use of computer models directly in the control loop poses new challenges for science and practice that go beyond the already familiar model-based approach, for example, in automatic control systems. Considering the all-encompassing nature of the application of computer modeling in CPS, which goes far beyond the scope of mathematical calculation, now we can talk about a new trend in cybernetic science that can be named Model-Oriented Control (MOC).

The concept of MOC disclosed in this e-book covers the main components of the control process, including the description of control algorithms using implementation models, forcasting the possible development of the control process based on predictive models and identifying the states of the control object using recovery models.

However, the presented material goes beyond just a theoretical presentation of the basic principles and methods of MOC, but also covers the issues of the practical application of this approach in modern IMSs. The striking example is the use of MOC in the creation of novel electron beam welding machines which, thanks to built-in models, acquired the features of intelligent industrial robots. Algorithm implementation models, which are actually executable programs, prediction of situations using modifications of temporal logics and virtual reality models, all of which in aggregate, solve the still impossible task of automatically welding highly complex spatial trajectories for high-tech products in the field of aviation and astronautics. A unique robot-welder that uses three electron guns simultaneously solves its problems based on the principles of multi-agent control, which is implemented using a set of built-in computer models. An important aspect of the MOC is also its application in the planning and quality management of manufacturing activities. Computer models of algorithms allow not only assessing in advance the effectiveness of planned activities taking into account the risk, but also managing the real process by assessing its development in dynamics.

These successes in the realization of the MOC are based on the already developed software and hardware, and the ideas presented in the e-book regarding the latest development of the MOC tools give us a reason to talk about its good prospects in the further use for IMSs.

Undoubtedly, this e-book will serve as a good possibility in the study of methods and technologies for constructing and applying CPS models by university students, scientists and engineers.

I would like to express my gratitude to all the participants of the "CybPhys" project who make their significant contribution to the training of specialists in CPS and IMS, in general.

Kyiv, Ukraine, April 2021
Alexander Palagin
Academician of the National Academy of Sciences of Ukraine,
Deputy Director for Research of V. M. Glushkov Institute of Cybernetics

Contents

Glossary of Abbreviations

ARM	Advanced RISC Machine
AS	Aggregate System
BDD	Binary Decision Diagram
C3	Computation, Communication and Control
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CA	Control Algorithm
CALS	Continuous Acquisition and Life Cycle Support
CASM	Cyclical Alternative Network Model
CCS	Computer-Controlled System
CEN	Control E-Net
CIM	Computer-Integrated Manufacturing
CLB	Configurable Logic Block
CNC	Computer Numerical Control
COSY	Concurrent System
CO	Control Object
CoAP	Constrained Application Protocol
CPM	Critical Path Method
CPS	Cyber-Physical System
CPSS	Cyber-Physical Product-Service System
CPSoS	Cyber-Physical System of System
CS	Control System
CTL	Computation Tree Logic
DCTL	Duration Computation Tree Logic
EBW	Electron-Beam Welding
EL	E-net Language
EMS	E-net Modeling System
E-Net	Evaluation Net
ERP	Enterprise Resource Planning
FMS	Flexible Manufacturing System
FPGA	Field Programmable Gate Array
HIL	Hardware-in-the-Loop
HLA	High Level Architecture
HMI	Human Machine Interface
IA	Intelligent Agent
ICPS	Industrial Cyber-Physical System
IIoT	Industrial Internet of Things
IIR	Intelligent Industrial Robot
IMS	Intelligent Manufacturing System
IoE	Internet of Everything
IoS	Internet of Services
IoT	Internet of Things

IP	Internet Protocol
JVM	Java Virtual Machine
LC	Life Cycle
LTL	Linear Time Logic
M2M	Machine-to-Machine
MAS	Multi-Agent System
MC	Model Checking
MES	Manufacturing Execution System
MIL	Man-in-the-Loop
MMU	Memory Management Unit
MQTT	Message Queue Telemetry Transport
MOC	Model-Oriented Control
MPC	Model Predictive Control
OMAC	Open Modular Architecture Controls
OS	Operating System
PLA	Piecewise Linear Aggregate
PLMP	Piecewise Linear Markov Process
PN	Petri Net
PNML	Petri Net Markup Language
PLC	Programmable Logic Controller
PLTL	Propositional Linear Temporal Logic
PM	Predictive Model
RAMI	Reference Architectural Model for Industry
RHS	Receding Horizon Strategy
RISC	Reduced Instruction Set Computing
RT	Real Time
RTI	Run-Time Infrastructure
RTM	Run-Time Monitoring
SCADA	Supervisory Control and Data Acquisition
SOA	Service-Oriented Architecture
TCTL	Timed Computation Tree Logic
TPTL	Timed Propositional Temporal Logic
TQM	Total Quality Management
VO	Virtual Organization
VR	Virtual Reality
VS	Vacuum Subsystem
WSN	Wireless Sensor Network

Introduction

The concept of Intelligent Manufacturing Systems (IMSs) was formed under the influence of the growing capabilities of information technology, penetrating into all spheres of human activity. An important stage in the development on this way was the emergence of Flexible Manufacturing Systems (FMSs) (Hartley, 1984) in the 1990s. Further development of works in this direction led to the formation of Computer-Integrated Manufacturing (CIM) (Buffa, 1984). At this stage of development, a number of fundamental ideas, principles and technologies arose and were partially tested. In particular, Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) automated systems were created (Berg, 1985). The exchange of geometric data in electronic form between CAD and CAM systems was one of the first real examples of informational process integration.

However, the idea of building a modern enterprise was most developed during the implementation of the international program "Intelligent Manufacturing Systems" (Gaines and Norrie, 1995), deployed by the world's leading industrial powers at the turn of the century. Within the framework of the IMS program, more than 50 international projects have been developed, as a result of which the latest production technologies have already been developed or will be created, forming the concept of a modern production system.

The analysis of the stages of development of production systems shows that the main element that forms their essence is the Computer-Controlled System (CCS) (Astrom and Wittenmark, 1996). IMSs do not cancel or replace the principles of building production systems, formed in the process of development of FMSs and CIM, but make them more intelligent, flexible and progressive due to the wide use of advanced information technologies at all management levels, including automated technological control systems and Enterprise Resource Planning (ERP) (Busse and Torsten, 1998), which together form an integrated automated production. In fact, IMSs bring high-quality intelligent definitions to each of the properties inherited from FMSs and CIM, improving the automation of technological processes and increasing the level of information integration of enterprises.

From the point of view of automation, IMS distinguishes the development of distributed SCADA (Supervisory Control and Data Acquisition) systems (Boyer and Stuart, 1999) to the level of implementation of Open Modular Architecture Controls (OMAC) (Bailo and Yen, 1997; Pritschow, 2001). In matters of information integration, IMSs solve the problem of not only data exchange between various kinds of computer programs, but also the problem of supporting the full life cycle of products, improving the Continuous Acquisition and Life cycle Support (CALS) (Fuhs, 1995) technology in the direction of creating virtual organizations (Travica, 1997).

However, the main characteristic feature of the IMSs, which actually determines their name and makes the management process intellectually rich, is the widespread use of computer modeling at all stages of management decision-making. In IMSs, computers have received a completely new purpose. In addition to program control and integration functions, they are increasingly assigned the tasks of perception, recognition and display of information, as well as the formation of managerial decisions on appropriate behavior in various situations of development of the production process. Computer vision, computer graphics, simulation, human-machine interface, synthetic environment and virtual organization are now becoming the most important components of the management process. The method underlying the functioning of these components is computer modeling.

IMS as a program of international cooperation has led to the creation of a new technological phase, called Industry 4.0 according to one of the 10 projects in Germany's Hi-Tech state strategy under the Smart Manufacturing concept launched in 2011 (Kagermann et al., 2013). The goal is to make greater use of IT in manufacturing to increase the competitiveness of the economy. The essence of Industry 4.0 is that the material world today is merging with the virtual, leading to the creation of Cyber-Physical Systems that integrate into a common digital ecosystem. An important direction in the creation and enlargement of Cyber-Physical Systems is the development of methods and technologies for their modeling and simulation (Kazymyr, Shkarlet, Zabasta, 2020).

The e-book is aimed at solving an urgent problem related to the development of methods and technologies for the use of computer modeling in the control of IMSs, which are distinguished by a complex structure and behavior dynamics. The lack of a grounded theory that allows formulating the basic principles of using models in the control loop of the CCS makes this problem relevant today. Its solution will contribute to increasing the efficiency of high-tech production by improving methods and means of management, built on the basis of progressive information technologies.

Chapter 1. Intelligent Manufacturing Systems and Industry 4.0 Concept

1.1. Cyber-Physical Systems for Intelligent (Smart) Manufacturing Approach

Cyber-Physical Systems (CPSs) are developed to integrate real physical processes and virtual computational processes. Different objects used in modern daily life represent CPSs. The definition of CPS from Cyber-Physical Systems Week (CPS, 2019) is as follows: "Cyber-physical systems are complex engineering systems that rely on the integration of physical, computation, and communication processes to function".

CPSs, which integrate computing and physical processes, involve more physical components than the pure embedded systems. In embedded systems, the key focus is on the computing element, but in cyber-physical systems, it is on the link between computational and physical elements (Sultanovs, Skorobogatjko, and Romanovs, 2016). Cyber-Physical System parts exchange information with each other; therefore, the third component – communication (see Fig. 1.1) is added there. That is why, Cyber-Physical System is denoted by the symbol C3 (Computation, Communication and Control).

Cyber-Physical Systems are developed to integrate real physical processes and virtual computational processes. Concept of CPS is complicated, but it can be illustrated with a concept map (see Fig. 1.2). The concept map depicts different views and approaches to CPS, such as spheres of application, requirements to modeling and design, cyber security concerns, the main features of CPS, etc.

Nowadays, it is recognized that flexibility, modularity, and reconfgurability are the main challenges in the design of manufacturing systems. Intelligent manufacturing applies embedded software and hardware technologies to optimize productivity in the manufacture of goods or delivery of services. This is resulted in developing a flexible, modular and distributed control architecture for automated warehouse systems using Function Blocks and CPS perspective (Gunes et al., 2014; Basile et al., 2015). An introduction to the Cyber-Physical Product-Service Systems (CPSSs) and their application in an industrial case are provided in Wiesner et al. (2017).

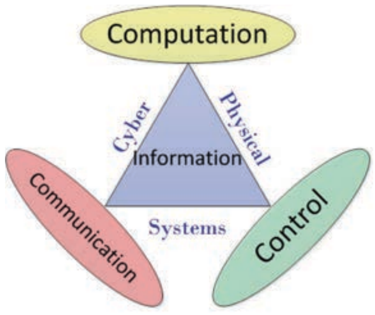


Fig. 1.1. Three main components of Cyber-Physical System (Wu and Li, 2011).

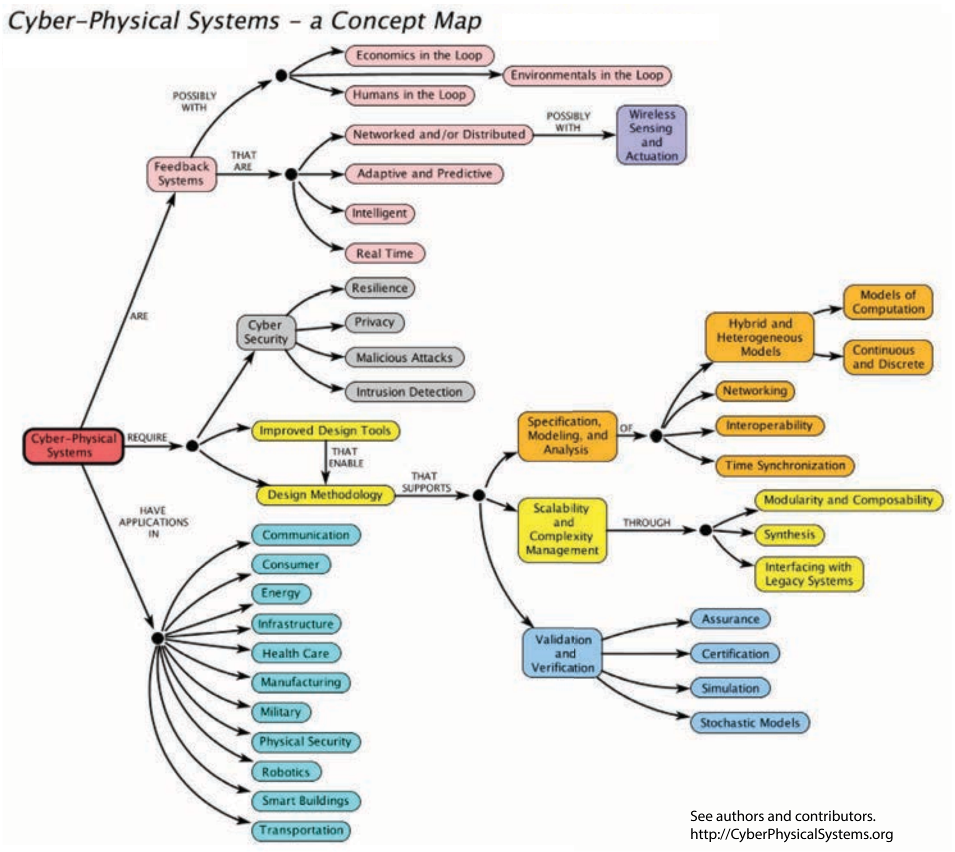


Fig. 1.2. A concept map of Cyber-Physical Systems (Berkeley, 2020).

They emphasize the multidisciplinary requirement engineering for the hardware, software, and service components as a key aspect for the successful and dynamic changes to CPSSs in industry. The majority of studies concerning CPS are focused on modeling, conceptualization, and utilization plans rather than on realization (Kang et al., 2016).

CPS perspective on the future industrial revolution will improve safety, productivity, and efficiency by connecting embedded system production technologies to pave the way to highly flexible workflow and efficient collaboration (Gunes et al., 2014).

It is pointed out that CPS can potentially revolutionize interaction with many complex systems, which the physical world critically depends on. According to Kim and Kumar (2013), CPS applications need to be designed considering the cutting-edge technologies, necessary system-level requirements, and overall impact on the real world. The goal of research is increasing reliability and safety, reducing resource consumption, or improving the overall performance of industrial processes.

Many modern cyber-physical applications demand guaranteed high performance, ultra-low energy consumption, high dependability, safety and security. It means that security and privacy, efficiency, and interoperability must be an integral part of the CPS. On the other hand, CPS is vulnerable to failures and attacks on both the physical and cyber sides, due to its scalability, complexity, and dynamic nature. Making use of a large-scale network (such as the Internet), insecure communication protocols, continious usage of legacy systems, application of commercial off-the-shelf technologies, are the other factors, which cause CPS vulnerability (Gunes et al., 2014). In addition, many of the applications of CPS are large-scale systems. It is not easy to integrate seamlessly heterogeneous systems, since each field of CPS has a self-contained set of models, languages and methods.

The Industry 4.0 project has been created as a strategic initiative, which represents a major opportunity for manufacturing the future. It refers to the deep integration of next generation information technologies (such as CPS) into industrial scenarios, solutions and procedures. By integrating with production, logistics and services in the current industrial practices, CPS will transform today's factories into an Industry 4.0 factory with significant economic potential. CPS will be able to transform existent factories into Industry 4.0 manufacturing with significant economic potential, by integrating production, logistics and services using the best industrial practices, and by integrating sensor data with enterprise information systems (Zhou et al., 2015). The millions of devices, not all-time smart, are interconnected, providing and consuming information available on the network. They will be able to exchange capabilities collaborating to reach common goals thanks to such an environment. Due to application of CPS, the production facilities, smart machineries, warehousing systems, business processes will be capable of autonomously exchanging information, triggering actions and controlling each other autonomously and independently (Lanting and Lionetto, 2015).

Furthermore, robotics for service is identified as one of the six disruptive civil technologies with potential impacts on the U.S. interests out to 2025 (Nic, 2008). The integration of humans and smart robots is essential to enable all actors of CPS to achieve better cooperation, collaboration, and organization to implement multiplex tasks (Chibani et al., 2013).

The review of literature shows that CPS is tightly integrated to some latest technologies, such as cloud, IoT, big data, M2M, and Wireless Sensor Networks (WSNs). These technologies affect each other in application; thus, future research should pay more attention to their interoperability and technological development.

1.2. Industry 4.0 Impact on Development of Smart Manufacturing

The Industry 4.0 methodology is recognized as a current driving force of the industry development, and represents the implementation of large-scale changes in the contemporary industry. These changes include digitization, automation, mechatronization and ICT integration at all levels of process control and services (Bassi, 2015).

Industry 4.0 represents the fourth industrial revolution in manufacturing industry (see Fig. 1.3).

According to Lee (2007), Lee, Bagheri, and Kao (2015), the Fourth Industrial Revolution is based on the following paradigms:

- *Interoperability* enables the integration and cooperation of intelligent machines, methods and human beings to interact through Internet of Things (IoT), Industrial Internet of Things (IIoT) and Internet of Services (IoS).
- *Virtualization* provides an opportunity to develop virtual model (or copy) of an intelligent factory. Such a model applies real data obtained from a plant and applied to the intelligent factory model for control of operations.
- *Decentralization* provides an opportunity for a device or a machine to carry out operations and decentralized (autonomous) control. Therefore, maximum qualified intelligent decisions on each subprocess for optimizing process production would be made.

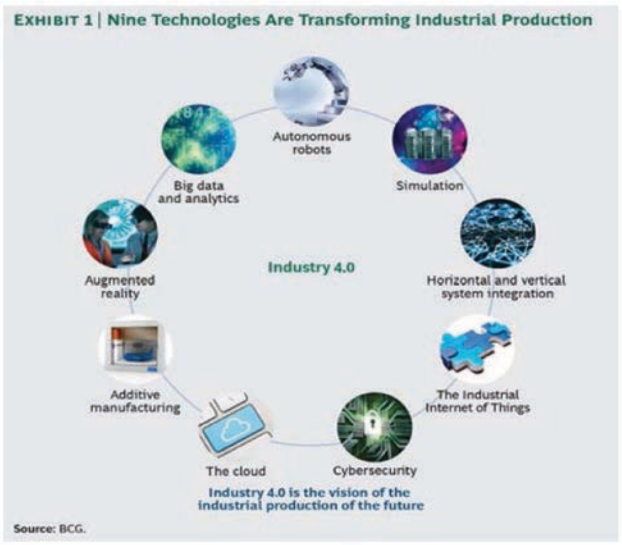


EXHIBIT 1 | Nine Technologies Are Transforming Industrial Production
Autonomous robots
Simulation
Horizontal and vertical system integration
The Industrial Internet of Things
Cybersecurity
The cloud
Additive manufacturing
Augmented reality
Big data and analytics
Industry 4.0
Industry 4.0 is the vision of the industrial production of the future
Source: BCG.

Fig. 1.3. Structure of technologies for manufacturing industrial processes included in Industry 4.0.

- *Real-time (RT)* data collection and analysis. Thanks to collected information, the real-time intelligent control and decision-making methods can be applied for optimization and reconfiguration, as well as can take into account failures and find optimal solutions such as component and device failures, transfer of production, etc.

- *Service oriented approach* is implemented due to communication and information exchange over the Internet of Things, by providing information to other parties of the company's services.

- *Modularity and reconfigurability* enable intelligent business to adapt flexibly to the production situation by changing software and hardware modules, by supporting the sharing, and reconfiguring processes (multi-criteria and multi-variant optimal intelligent decisions).

The application of the IoT to the manufacturing industry is called the Industrial Internet of Things (IIoT). IIoT is part of a larger concept known as the Internet of Things (IoT). The Industrial Internet of Things (IIoT) means the use of Internet of Things technologies in industrial processes, e.g., manufacturing, transportation, energy production, etc. The IIoT incorporates machine learning, cloud computing and big data technology, harnessing the sensor data, machine-to-machine (M2M) communication and automation methods and technologies. The target of IIoT is the improvement of product manufacturing, enabling supply chain efficiency due to exchange of information, mathematical modeling, optimal control, effective coordination and big data (Fig. 1.4).

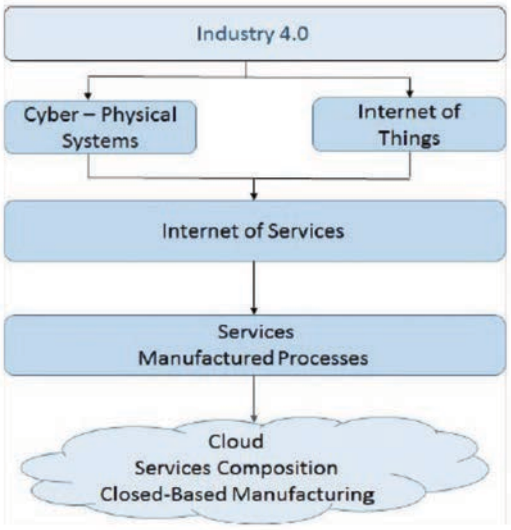

Industry 4.0
Cyber – Physical Systems
Internet of Things
Internet of Services
Services Manufactured Processes
Cloud Services Composition Closed-Based Manufacturing

Fig. 1.4. Basic scheme of the interconnection between IoT and Industry 4.0.

In literature, we can find several main principles of Industry 4.0, for example, the use of the Internet, production flexibility, virtualization of process, etc.

Extensive Use of the Internet

The extensive use of the Internet enables the capability to collect gigabytes of data per hour from millions of devices to be analyzed in real time, finding clusters of potential issues or problems to be used for predictive maintenance, shortening dramatically the loop of collecting diagnostic information in order to make a decision.

Flexibility – Handling High-Mix, Low Volume

One of the most attractive features of a Smart Factory is the capability to operate on small batches, down to batch-size-one. In comparison with previous experience, Industry 4.0 now is able to provide real-time, "zero-setup-time" production flexibility to meet the new demand of personalization and mass customization not just in a B2C perspective, but also in the B2B context. Industry 4.0 will be able to supply highly customized products (like Private Label), as well as pre-series and prototypes.

Traceability & Product Identification

Today data marking and reading ability are prerequisites for a Smart Factory where machinery, products and systems are connected along the Value Chain. Industry 4.0 enables "production flexibility" by capability to assign a unique ID to each component and also by ensuring the real-time control, and the complete value chain over the entire product life cycle. A unique component ID makes every single component individually identifiable alongside the entire production process allowing for dynamic, more efficient production paths, down to batch-size-one. It makes possible to retrieve information regarding the origin, storage, state and location of materials, components and products.

Communication, Virtualization and Cyber-Physical Systems

Industry 4.0 encourages the use of a reliable stable and powerful common language to drive the revolution across the globe through cloud technologies. For this purpose, an open source communication standard, based on the Ethernet, OPC-UA could aid users to share information across the entire infrastructure. Such communication standards provide capability for secure data overcoming software and hardware differences working on the client server model (OPC, 2020). Once a common communication framework is available, it is possible to connect details suppliers, assembly machines, and sensors to describe and define their functionalities and treat them as virtual computational entities. Therefore, it makes possible to create links between physical processes and their virtual representation.

1.3. Adoption of Cyber-Physical System Paradigm in Smart Manufacturing Environments

The main objective of this chapter is to explore the ways for the effective adoption of Cyber-Physical System of System (CPSoS) paradigms in smart manufacturing environments to enhance the efficiency, data analytics, connectivity, multiple task execution, self-decision making and real-time system interaction and to describe the main principles that will be needed for the implementation of CPSoS.

1.3.1. System of Systems for Industrial Applications

A System of Systems is an integration of a finite number of constituent systems which are independent and operable, and which are networked together for a period of time to achieve a certain higher goal. The System of Systems concept could be considered software design, management and exchange of information and control techniques to optimize the production and management of physical processes in the industrial environments. According to Haber et al. (2015), the demand of real-time, optimal and reliability solutions is based on global information, knowledge and parametrization to execute reconfiguration and control actions that introduce several challenges to the industrial SoS ecosystems (Haber, Juanes, Toro and Beruvides, 2015).

A paradigm for digital transformation and the interconnectivity of multiple devices based on the Internet of Everything (IoE) and Internet of Things concepts, and the integration between the cyber world (e.g., algorithms, software, apps, etc.) with the physical world (e.g., devices, machines, automobile, buildings, etc.) have been proposed as prerequisites for Smart Manufacturing (Colombo, Bangemann, and Karnouskos, 2013; Morkevicius, Bisikirskiene, and Bleakley, 2017).

1.3.2. Industrial Cyber-Physical Systems and RAMI 4.0

Industry 4.0 represents new paradigms of information and communication technologies, such as Industrial Internet of Things (IIoT), Industrial Cyber-Physical Systems (ICPSs), Service-Oriented Architectures (SOAs), cloud computing and big data implementations, as well as the introduction of innovative advances in the cybersecurity, distribution and decentralization of the information and computing capabilities in new industrial connected ecosystems (Zheng et al., 2018).

Architectural Model Industry 4.0 (RAMI 4.0) combines three core dimensions of product development and production in a cuboidal space, covering from product to connection with a global ecosystem. The RAMI 4.0 depicts multiple layer integration from assessments to business unit, the interconnection of the shop floor devices with condition-bwwased monitoring, Human-Machine Interfaces, open protocols, data analytics and management (Flatt et al., 2016). The dimension hierarchy levels are based on the layers defined in IEC 62264 and IEC 61512 (Fleischmann, Brossog, Beck, and Franke, 2017).

Manufacturing Execution Systems (MESs) and Enterprise Resource Planning (ERP) are still disseminated solutions to take into account visualization, planning and control tasks in several industrial scenarios (Iarovyi et al., 2016; Ramis and Lastra, 2017). In the current manufactory, MES provides many functionalities in relation to monitoring, resource allocation, task scheduling, data acquisition, maintenance, performance analysis and control operations on the shop floor, with real-time access to key performance indices to facilitate necessary reconfiguration actions (Arica and Powell, 2017). On the other hand, ERP manages and tracks all the information and operational services at a company, covering functional areas, such as human resources, logistics, finance and production in order to support decision making by company management.

1.3.3. Approach of Cyber-Physical System of Systems for the Industry

The high-level architecture of the proposed CPSoS is depicted in Fig. 1.5. As cyber and physical integration, the CPSoS is composed of different subsystems and components from the two domains, i.e., cyber and physical. These domains are connected by the IoE technologies. The physical domain is usually composed of physical resources that are used at different systems for altering and sensing the environment. Examples of physical resources per CPSoS might be refrigerators for adjusting the humidity and temperature of premises, conveyors that supply components at factory shop floors, industrial vehicles etc.

The cyber domain might be composed of a set of soft applications, including digital twins, and usually represented by cyber models. These models facilitate the study of system behavior in order to monitor its performance. The cyber models are helpful to achieve zero-defect manufacturing, since it becomes possible to find anomalies and adjust the physical equipment using the virtual model (Vafeiadis et al., 2017).

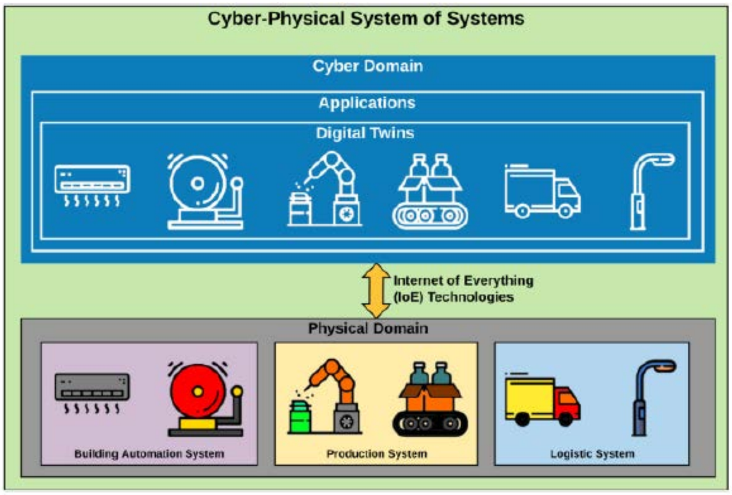


Fig. 1.5. A high-level vision of the proposed CPSoS.

1.3.4. The Main Components of the Approach

The web-services that collect data enable the devices, which belong to different systems, by connecting them to various types of sensors, actuators and data exchangers. Even human workers can also be considered resources that generate data, since they interact with different subsystems within the CPSoS, e.g., when they send notifications concerning the receipt of a package necessary for production. One of the main components that will support the interaction within CPSoS will be the IoT devices (Sheng, Mahapatra, Zhu, and Leung, 2015). Following the SOA paradigm, the IoT devices are often deployed at factory shop floors in order to implement both horizontal and vertical integration of enterprise systems. However, it should be taken into account that the proprietary solutions often represent an integral part of any CPSoS.

1.3.5. The Potential and Challenges of the Approach

In order to orchestrate isolated systems under a CPSoS, a set of qualitative attributes should be critical for the CPSoS:

Connectivity: Current isolated systems that affect the productivity of factories will be connected within IoT-based implementations for collection and exchange of data.

Digitalization: Systems will be digitalized in the cyber world with digital twins of resources in order to monitor and even control their behavior.

Modeling: Modeling techniques adopted by the CPSoS will help monitor system behavior, suggest the design of systems and process execution.

Flexible reconfiguration: The analysis of huge amount of data produced by physical resources and their digital models will permit the anticipation of required changes on runtime, providing flexible reconfiguration of systems.

Versatility and reusability: The digital twin of the system makes it possible to analyze its performance at the operation cycle and even to assign new tasks that were not considered during the system design cycle. The security of isolated systems may cause problems related to the integration of CPSoS; however, the employment of standard formats and risks, and threat modeling analysis will facilitate addressing these technological challenges (Ferrer, Afolaranmi, and Lastr, 2017).

Heterogeneity: The problems to be resolved concerning different types of sources of the data: a) data formats should be homogenized; b) to address data transformation, the gateways are necessary.

Integration: The integration of isolated systems will force the creation of new and adaptable interfaces among systems. In addition, the integration of legacy systems may be problematic for retrieving/receiving signals, data and information to/from different interested parties.

Interoperability: Apart from the issues at the interface, systems must communicate within similar protocols. Mutual platforms that are agnostic to protocols, such as OPC-UA1, may be of use for the implementation of CPSoS.

Security: Data security is a strong challenge for the integration of sensitive systems in the CPSoS. The data among different systems comprising CPSoS must be secured; therefore, the data cannot be accessed for malicious interests or retrieved without permission.

1.4. Migration Approach to SOA-based Process Control and Monitoring

A long list of requirements has been set for the next generation of industrial processes because of involvement of many actors and new stakeholders who create a large challenge for technology suppliers in the future. ISA-95, standardized through ISA (ISA 2020), represents a standard architecture for automation systems (Scholten, 2007). It is accompanied by a set of related standards, such as ISA-99, IEC 62443 (Staggs, 2020), which focuss on security of the systems. Nowadays, the key technology, which enables the integration within and in-between different levels of the ISA-95 architecture, is a Service-Oriented Architecture (SOA) (Erl, 2007). SOA was originally developed by IBM to enable data and information exchange between heterogeneous information systems. Nowadays, SOA is adopted as the main approach to plant automation due to application of a cloud technology and shared services.

1.4.1. The Proposed Architecture

The Service-Oriented Architecture enables the integration of industrial devices and IT systems in a cross-layer interaction mode from the shop floor to the business levels:

- The notation "service" is exposed as structural and behavioral properties and networking capabilities.

- SOA approach implements "services" as integrated in collaborative business relationships with other devices and systems in the CPSoS. Figure 1.6 shows the proposed architecture composed of services (marked as an "S" and depicted in a green cube), which are wrapping via web services many different devices and systems in an independent way no matter of the physical location of the devices and systems in the enterprise architecture.

The Internet Protocol (IP) set and web services are used in all layers and subsystems. Thanks to a cloud-based approach, a multi-level composition of System

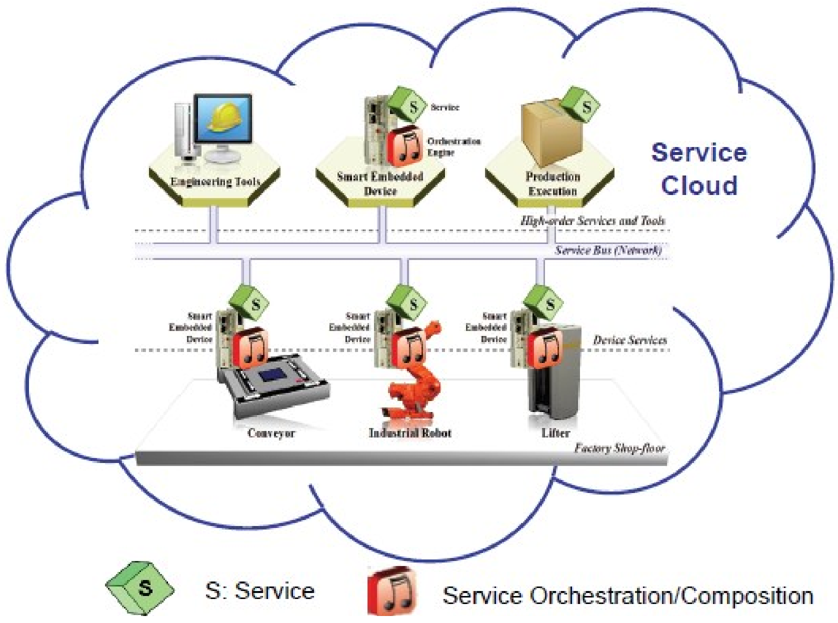


Fig. 1.6. SOA-driven architecture.

of Systems is possible together with Services of Services (see Fig. 1.6). The integration and interaction between business systems, such ERP and MES, and factory floor systems, such as SCADA, DCS, etc. are achieved thanks to the use of the cloud-based architecture. This interaction between the different level systems allows the CPSoS to develop additional functions, which were not initially envisioned by the constituent systems (Karnouskos and Colombo, 2011).

Having in mind that an SOA-based system behaves asynchronously (in opposition to the majority of currently implemented industrial process control and monitoring systems), it is a complex and challenging task to seamlessly integrate a large number of devices and systems from different manufacturers into a single SOA ecosystem.

The first step is the identification of the right wrapping ICT technologies and services. Some of standardized services can be identified as “generic services” because they are common for all devices and systems, and the other ones are labeled as “infrastructure services” by SOCRADES (www.socrades.eu) and NESSI (www.nessi-europe.com) projects. During the second step, we specify the mechanism for providing orchestration, choreography and composition. This mechanism has to include at least two main functions: to process information content of the services and to process the events related to the services. During the next steps, we have to define, specify and implement a mechanism for monitoring and control, which is based on an SOA approach.

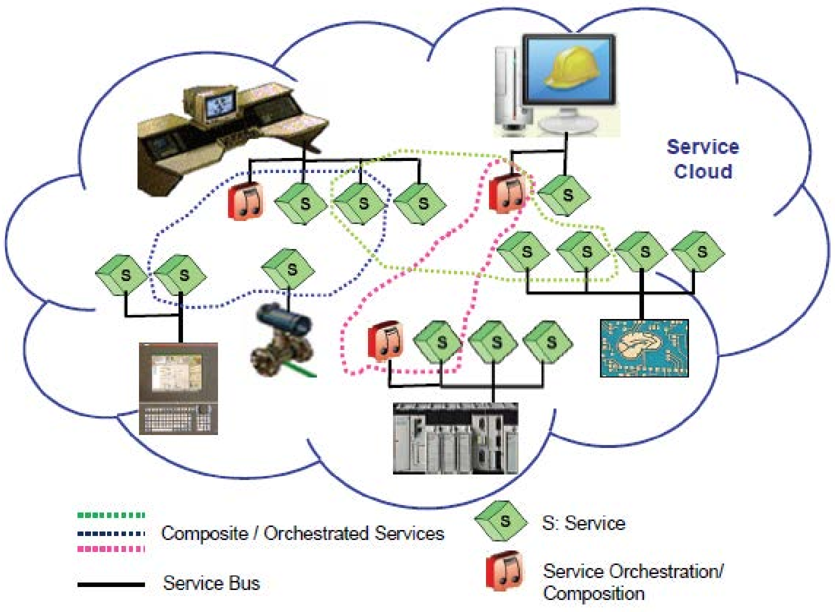

Service Cloud
S
Composite / Orchestrated Services
Service Bus
S: Service
Service Orchestration/ Composition

Fig. 1.7. Hierarchical composition of services enables abstract cross-layer functionality.

1.4.2. Migration Approach from Current Legacy to SOA-based Industrial System of Systems

A migration approach from current legacy to an SOA-based industrial system of systems (Jamshidi, 2008; Simanta et al., 2020) will follow a set of basic steps, which are summarized in Fig. 1.8. Current legacy industrial systems are mainly specified, implemented and running following the ISA95 standard (ISA 2020).

It means that migration to an SOA-based system cannot in general be performed at only one or some of the levels of the architecture shown on the left side of Fig. 1.8. This is because specifications and system characteristics at a defined level are closely related to specifications at other levels (e.g., control specification at Level 1 will only be well implemented when it considers information and actions performed at the neighbour levels like SCADA or MES above them). Thus, a migration strategy has to address how the migrated part can represent the legacy functionality and how it is involved in another level of the control system.

Several migrations are defined and specified for each system level. The definition and understanding of the differences within the legacy system is a necessary task, when migration of the system is planned. The other obligatory task is definition and understanding differences of the monitoring and control in the legacy system, which should migrate.

1.4.3. Industrial Information Distribution and RAMI 4.0 Concept

This concept builds upon the last generation of industrial monitoring and control systems in order to enable a smooth level of interaction between shop-floor devices and high-level enterprise systems. Originated from ISA 95, the Industry 4.0 initiative has proposed the Reference Architectural Model for Industry (RAMI) 4.0 (Schweichhart, 2016) and I4.0 component model (Model, 2020; Romanovs et al., 2019). The I4.0 component model captures the notion of an administration shell that abstracts the digitalized equipment and products with high levels of connectivity. The RAMI 4.0 captures a three-dimensional cube for modeling architectural solutions. It presents I4.0 components at different "hierarchies", which are designed over a complete "life-cycle" and must participate in a functional "layer". From a layer point of view, a single functional "layer" cannot be confined to a single level of the "hierarchy". Rather a "layer" is spread across many I4.0 components at different levels of the "hierarchy", which is shown in Fig. 1.9.

The RAMI 4.0 determines that connectivity and integration of the industrial systems should not be considered a purely vertical approach. According to RAMI 4.0, the components of industrial systems could communicate with one another vertically, horizontally and even diagonally. The components of RAMI 4.0 can participate in the information layer as data producers and consumers.

Work centers, cells, and stations achieve a particular level of autonomy and diminish their dependency on MES cloud in a case, when the information layer enables a seamless flow of data. Reducing coupling between physical work cells and software provides an additional advantage manifested through higher reliability of work centers, as malfunction in one work cell does not degrade the production of the whole work center.

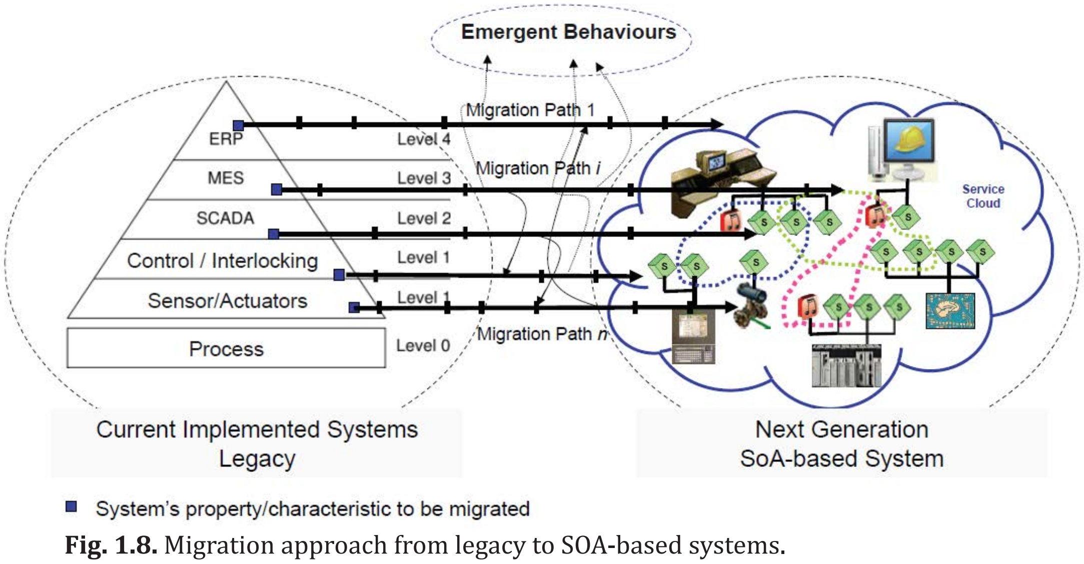


Fig. 1.8. Migration approach from legacy to SOA-based systems.

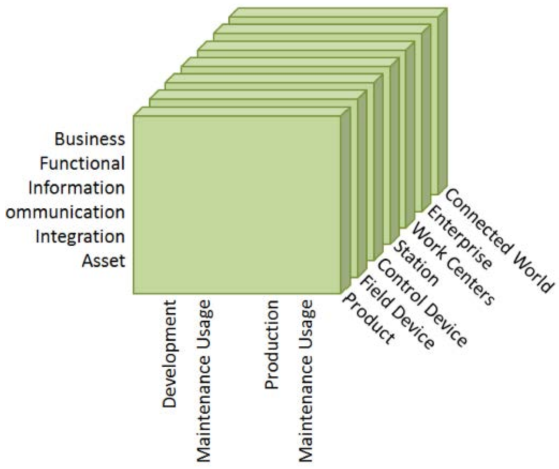


Fig. 1.9. RAMI 4.0 three-perspective cube (adapted from Basile et al., 2015).

To enable a flexible production process, it is important that engineers and specialists on the shop floor get access to local data; therefore, they do not need involvement of IT staff, when data routing change is needed. The requirements for decentralized information distribution of the flexible production process can be defined as follows:

- The shop machines can operate in an islanded mode: a centralized data store is not necessary. In a case of network performance degradation, the operations can be continued.
- The specification of data flow should be recorded in readable form both for human personal and for machine.
- Production process should be technology independent; therefore, the choice of technology should limit technical solutions.
- To enable granular access control support, industrial designers are allowed to make changes and restrict untrained or hostile changes.

1.4.4. Flexible and Secure Communication in Intelligent Manufacturing Systems

The CPPS are integrated and built on many existing technologies and components, such as industrial production environment, including industrial devices equipped with sensors and actuators, IIoT components, and backend systems, such as cloud platforms.

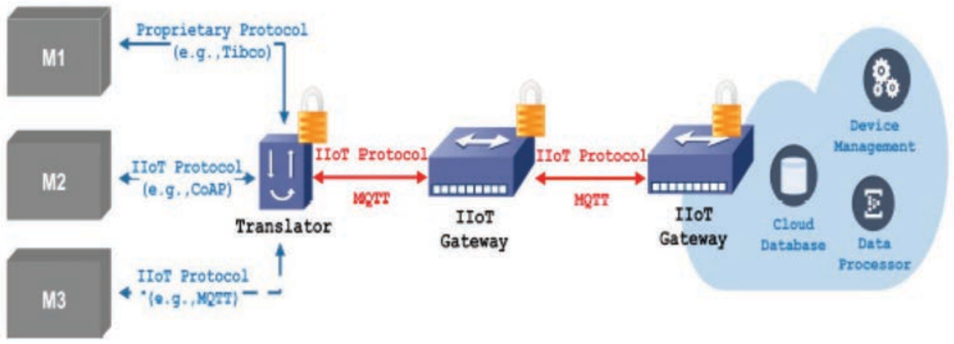


Fig. 1.10. CPPS end-to-end communication use case for an Industry 4.0 application scenario.

The use case depicted in Fig. 1.10 shows an Industry 4.0 application scenario, when industrial devises of the CPPS communicate in a flexible and secure manner. In the offered use case, data are exchanged between CPPS devices via the network, and delivered to the cloud for continued processing and analysis. The industrial devices, which are depicted as M1, M2 and M3, for communication with gateways, and the cloud backend system apply such protocols as MQTT (Message Queue Telemetry Transport) and CoAP (Constrained Application Protocol). The IIoT gateways distribute lifetime data among CPPS devices and send them throughout a network to the cloud storage.

The MQTT protocol demonstrated in Fig. 1.6 is a lightweight protocol widely used to accommodate constrained devices with low power and bandwidth requirements (Zabasta et al., 2017; Zabasta et al., 2018).

We can see that modern industrial devices M2, M3 use state-of-the-art protocols (MQTT and CoAP) when communicate inside of CPPS. On the other hand, a protocol translator is needed to translate an appropriate protocol of the legacy into a modern protocol. The protocol translator to be applied also for translation among different protocols is used in IIoT. The translation system solutions, e.g., Arrowhead protocol translation system, are described by Derhamy, Eliasson, and Delsing (2016).

The security of modern industrial devices is a crucial issue; thus, transmitted data must be encrypted. However, even software-based encrypted data are prone to attacks in order to reveal encryption tools. The work (Derhamy, Eliasson, and Delsing, 2016) offered to integrate special hardware called “Secure Element” in the protocol translator. Such secure elements will be able to protect encryption keys from hackers’ attacks, even in a case when physical interruption has happened.

1.5. Summary

In Chapter 1 "Intelligent Manufacturing Systems and Industrial 4.0 Concept", we have discussed the principal issues of development of smart manufacturing systems. The subchapter "Cyber-Physical Systems for Intelligent (Smart) Manufacturing Approach" has offered several definitions of Cyber-Physical System and provided a concept map of application and implication of CPS recognized by researchers in this field, which is followed by review of the relevant literature.

In its turn, the subchapter "Industry 4.0 Impact on Development of Smart Manufacturing" provides an analysis of paradigms of the Fourth Industrial Revolution, which reveals challenging advantages to manufacturing: interoperability as interaction through IoT, IIoT and IoS; virtualization that enables creation of a virtual model of the factory; production decentralization; real-time data collection and analysis; service-oriented communication in IoT.

Further, the subchapter "Adoption of Cyber-Physical System of System Paradigm in Smart Manufacturing Environments" explores the ways for the effective adoption of Cyber-Physical System of System paradigms in smart manufacturing environments. We have discussed the implementation of CPSoS that comprise necessary components: data analytics, connectivity, multiple task execution, self-decision making and real-time system interaction.

In subchapter "Migration Approach to SOA-based Process Control and Monitoring", we have analyzed the offered architecture. We have concluded that the SOA-based enterprise architecture allows devices and systems from the shop floor to the business levels to communicate and exchange data in a cross-layer interaction mode. One of the discussion topics is the understanding of the approach, which will make it possible to migrate from current legacy to an SOA-based industrial System of Systems. The selected migration approach has to take into account information distribution in industrial systems, when migration from legacy manufacturing to smart manufacturing takes place under Industry 4.0 concept.

In subchapter "Flexible and Secure Communication in Intelligent Manufacturing Systems", we have discussed a use case that illustrates a flexible and secure end-to-end communication, when the CPPS components are mapped to create the meta-model of industrial CPSoS. The use case depicts data transmission between devices and the private clouds for processing and analysis. The communication protocol used between the industrial devices and security issues have also been discussed.

Chapter 2. The Principles of Model-Oriented Control

2.1. The Main Characteristics of the IMS in the Control Context

The evolutionary scheme of the development and formation of IMS as the newest and most advanced representative of the class of production systems can be represented in the form shown in Fig. 2.1.

Following this scheme, we can distinguish three main characteristics of Control System (CS) of IMS in the context of solving control problems, which are presented in Fig. 2.2.

2.1.1. Open Modular Architecture Controls

The Open Modular Architecture Controls (OMAC) concept was first proposed by General Motors in the summer of 1994 in a document containing requirements for controllers used in the automotive industry (Taylor, 1998).

Later it was developed by European (European Open System Architecture for Controls within Automation Systems – OSACA) and Japanese (Japan International Robotics and Factory Automation – IROFA and Japan Open System Environment for Controller Architecture – OSEC) organizations (Lutz, 1998; Sawada and Akira, 1997). A number of promising OMAC-based programs are supported by the U.S. government.

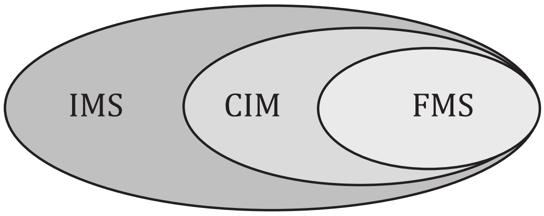
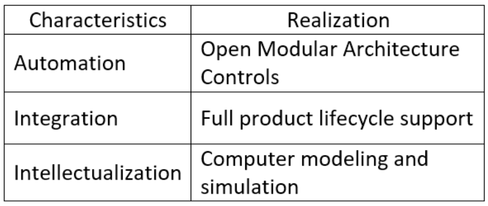


Fig. 2.1. Stages of IMS formation (Kazymyr, 2006).



Characteristics	Realization
Automation	Open Modular Architecture Controls
Integration	Full product lifecycle support
Intellectualization	Computer modeling and simulation

Fig. 2.2. Main characteristics of IMS Control System.

In general terms, OMAC fundamental requirements for automation systems are formulated as follows:

1. **Open** architecture that provides integration of widely used hardware and software in the market.

2. **Modular** architecture that allows you to easily change the distributed structure of the control system by changing the composition of its components and the connections between them.

3. **Scaleable** architecture that allows you to easily and efficiently change the configuration for specific needs.

4. **Economical** architecture providing low cost of controller equipment life cycle.

5. **Maintainable** architecture that can withstand harsh operating and maintenance conditions, thus ensuring minimal downtime.

The noted features of the open architecture can be implemented to the maximum extent by using PC-compatible industrial computers in control systems instead of the traditionally used systems based on Programmable Logic Controllers (PLCs). The main advantage of personal computers (PCs) in this case is associated with their openness and the ability to use the most modern hardware and software that meet international and regional standards.

2.1.2. Full Product Life Cycle Support

Analysis of the application of information technologies in the CCS shows that the second (after the use of open architectures) area of their development is a more complete coverage of all stages of the product life cycle (LC).

Despite the fact that in the instrumentation the range of tasks solved by the control system has significantly expanded, the issues of interaction with the customer, after-sales support of the product and many others, which form the basis of the quality management system, remain unresolved.

For the first time, work on the creation of integrated systems that could support the product life cycle had begun in the US defense complex. The new concept was in demand by life as a tool to improve the management of the logistics of the US Army. It was assumed that the implementation of the new concept CALS (Computer Aided Logistic Support – computer support for the supply process) would reduce the cost of organizing information interaction between government agencies and private firms in the process of formalizing requirements, ordering, supplying and operating military equipment. Having proved its effectiveness, this concept has consistently been improved, supplemented and, keeping the existing abbreviation CALS, has received a broader interpretation – Continuous Acquisition and Life Cycle Support (continuous delivery and information support of the product life cycle) (Fuhs, 1995).

Now, CALS has evolved into a global business strategy for the transition to electronic document management technology, ensuring the integration and sharing of information at all stages of the product life cycle. The development of the concept has led to the emergence of a new organizational form for the implementation of large-scale projects associated with the development, production and operation of complex products – a virtual organization (VO) (Travica, 1997).

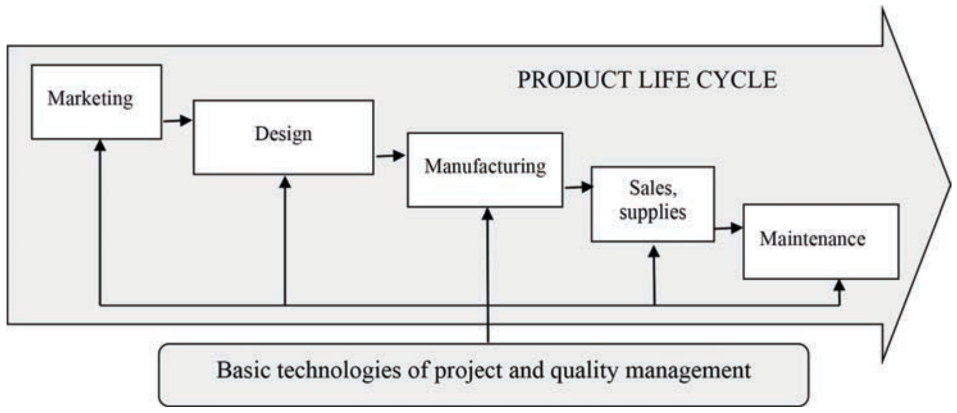


Fig. 2.3. The conceptual model of CALS.

In the conditions of the functioning of the VO, integration is carried out on the basis of global networks, in particular the Internet. Thanks to the use of Web-based information systems, it is possible to combine the information resources of geographically distributed divisions and organize remote management based on a single strategy.

The conceptual model of CALS is shown in Fig. 2.3.

CALS relies on two main process control technologies that are invariant with respect to the object (products):

- project and task management (Project Management/Workflow Management);
- quality management.

In many developed countries, CALS is considered to be a strategy for survival in a market environment.

2.1.3. Computer Modeling

The use of computers in control systems was characteristic of all the main stages of the IMS. However due to the growth of PC productivity, their role in control systems changed significantly. If, at the beginning of the automation period, the functions of computers were limited to the use of digital controllers and the maintenance of routine accounting and statistical tasks, then in the FMS and CIM, computers became the central link of the control systems. They started to provide not only control of CNC machines and industrial robots, but also the implementation of technological preparation of production.

Computers received an even more responsible appointment at the IMS. In addition to program control and integration functions, they began to be assigned the tasks of perception and recognition of information, assessing a dynamically changing environment and forming managerial decisions on appropriate behavior in various situations. The method underlying the solution of these problems is modeling because it is always assumed to use a model – some approximation to a real object. Modeling in this case is computer based, since it requires significant amounts of computations, the execution of which is possible only with the use of high-performance computer systems.

Figure 2.4 demonstrates the main types of computer modeling used in the process of IMS management and their distribution according to the projects of the world program “Intelligent Production Systems”.

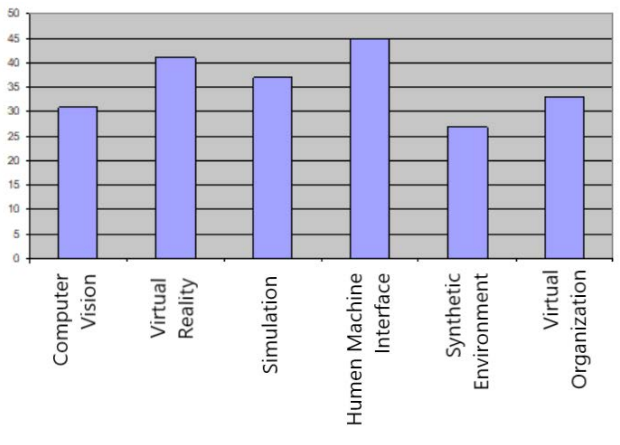


Fig. 2.4. Distribution of IMS projects in computer modeling types.

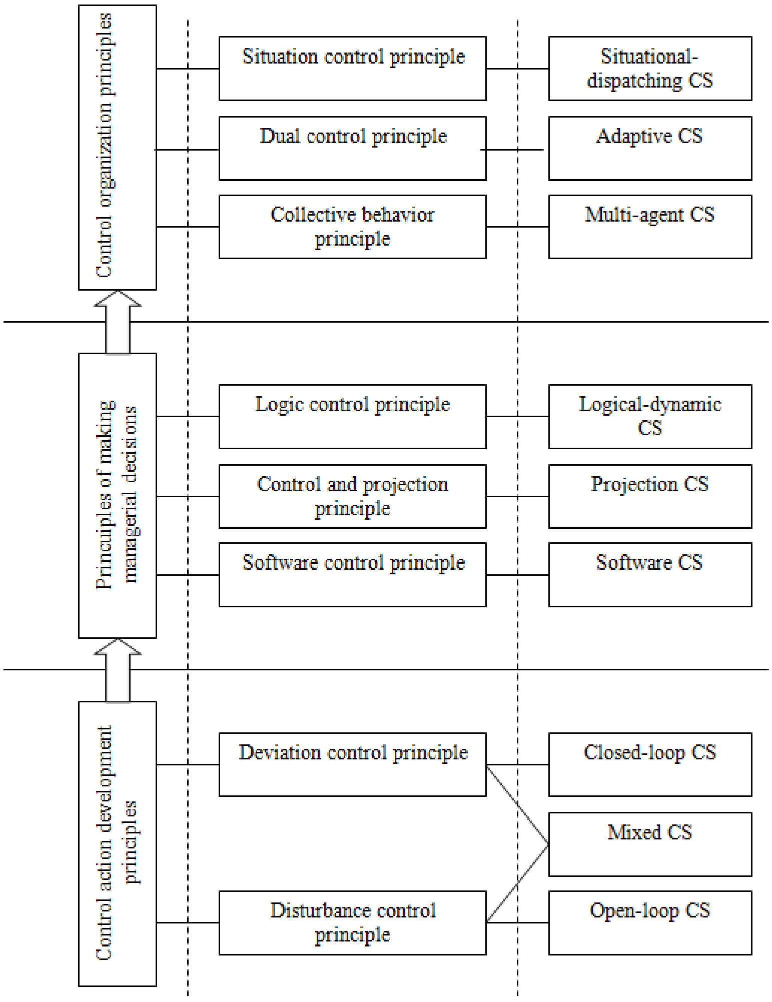
If earlier computer modeling was considered, basically, only the main method of analysis and synthesis of control systems, then in the IMS context it became possible to talk about the possibility of using computer models directly in the control loop as a means of developing control decisions. It determines the relevance of solving a whole range of tasks related to the development and use of computer models in the process of managing the IMS.

2.2. Features of IMS Control Process

The investigation of this issue is conducted in the scope of the analysis of current IMS control principles and CS structure, which facilitate IMS realization.

2.2.1. IMS Control Principles

Control principles characterize the control law, which provides the answer to the following strategic question: “What kind of dynamics switches the system to the necessary condition?”



Control organization principles
Situation control principle
Situational-dispatching CS
Dual control principle
Adaptive CS
Collective behavior principle
Multi-agent CS
Princuiples of making managerial decisions
Logic control principle
Logical-dynamic CS
Control and projection principle
Projection CS
Software control principle
Software CS
Control action development principles
Deviation control principle
Closed-loop CS
Mixed CS
Disturbance control principle
Open-loop CS

Fig. 2.5. IMS control principles and corresponding Control System classes.

We consider IMS control principles to be a hierarchical structure as shown in Fig. 2.5. This representation develops the already accepted terminology of control theory, provides additional structuredness of the terms used taking into consideration current trends in CCS development. Consequently, control principles determine the class of CCS, which can be implemented on the basis of these principles.

CCS control principles can be grouped into three levels, according to the level of impact on their structural and dynamic characteristics:

- control action development principles;
- principles of making managerial decisions;
- control organization principles.

We should point out that each subsequent level in the given hierarchy includes the previous one. As a result, the general concepts of CCS construction and functioning that facilitate its intended purpose presuppose the use of internal decision-making mechanisms, which are implemented directly by means of the control action development schemes. This way the target stability and constructive feasibility of the CCS are achieved.

In the scope of the objectives of the study, we are mostly interested in the control organization principles, which are considered to be the unifying core in the development of a control strategy applied to the IMS. We will discuss these principles in detail further.

Currently, the following control organization principles are singled out in the class of complex dynamic systems: *situation control*, *adaptive control*, and *multi-agent control*. Each of these principles is based on its own methods and sets of mathematical models.

Situation control is based on the notion of situation, classification of situations and their transformation. The definition of situation is based on the fact that it is not always possible to accurately determine the description of an CS using an equation of state in either discrete or continuous form. There are systems and particularly complex subsystems referred to as ill-defined or semistructured (Jakobson et al., 2007). As a rule, they are characterized by the following features: evident uniqueness of the Control Object (CO), absence of a clearly formulated criterion of optimality, high dynamism, incomplete description of the functioning process, and the presence of discretion.

For such systems, it is impossible to apply the traditional control method, which is based on any analytical model of the processes occurring in the object. On the other hand, CO can be studied to such an extent that it may be possible to describe the situations that develop in the system. The description of a situation, in this case, is defined as a collection of all information about the structure of the CO at a given moment in time, as well as the knowledge about the technological control scheme, which is represented by the rules for choosing control decisions. Formally, an elementary act of situation control can be written using the following expression:

$$s_i: x'(t) \xrightarrow{u(t)} x''(t). \tag{2.1}$$

The meaning of this expression is as follows. If $s_i \in S_i$ situation, caused by the state of CO, has developed in the CS, and the technological scheme allows for the use of control u from the control tolerance range U, then it is applied and CO goes into a new state $x'' \in X$.

Situation control principle presupposes the execution of the following sequence of steps:

• analysis of emerging situations (situation detection);

• correlation of the identified situation with the known class of situations (situation recognition);

• selection of the required control solution corresponding to the given class of situations (making a decision regarding the control choice).

Adaptive control is the second control method, which has earned a fundamental position in the modern theory of complex system control. Adaptation is a method of control under conditions of insufficient a priori information and consists in improving the quality of control by changing the structure and / or parameters of the CCS (Astrom, 2008).

The lack of a priori information leads to the need to combine, in a sense, the study of an object and the control over it. Therefore, adaptation implies duality of control, when control u, by changing the state of the system x, also affects the characteristics of information about the system P. The transition to a new state can be determined by operators H_1 and H_2 in the following way:

$$x(k+1) = H_1[x(k), P(k), u(k)], \tag{2.2}$$

$$P(k+1) = H_2[x(k), P(k), u(k)]. \tag{2.3}$$

In order for information about the system to accumulate over time, it is necessary to explicitly choose H_2 so that the description of system $P(k+1)$ would be more complete than $P(k)$. If a certain indicator of quality control is associated with the state $x(k+1)$, then due to greater control awareness, as a result of adaptation, this indicator can consistently improve. In this case, the sequence of transformations $[H_1, H_2]_k$, $k = 0, 1, 2, \ldots$ determines the process of adaptive control.

The specific content of the theory of adaptive control (in particular, operators H_1 and H_2) is revealed in a number of works devoted to the study of adaptive control systems of various classes. The most notable achievements of this line of research have been made in relation to non-searching adaptation methods, methods of direct and identification approach with adaptive control of complex systems, mainly, of the technological level.

Multi-agent control. At the present stage of IT development, in particular, due to the use of OMAC, the implementation of the aforementioned directions of interaction in the control process can be successfully carried out due to the use of intelligent agents (IAs). There are many definitions of IA, all of which, however, are based on the properties of autonomy and purposefulness. In this aspect, there are three functions that characteristize IA (Wooldridge, 2002): 1) perception of the dynamics of the environment; 2) actions that change the environment; 3) reasoning for the purpose of interpreting observed phenomena, solving problems and determining actions. The first two directly correspond to the tasks solved by control elements, in particular those related to the "empowerment". The third function significantly expands the internal structure of control elements and creates prerequisites for their interaction.

The interaction of agents is organized within the framework of multi-agent systems (MASs). The questions of the MAS theory were subjected to scrutiny in many works. A significantly fewer number of publications are devoted to the issue of practical implementation of the MAS, especially in the field of manufacture control.

Still, based on the mentioned works, we can conclude that a multi-agent control system can be generally viewed as the following tuple:

$$S =< W, A, \varOmega, \varLambda, K >, \tag{2.4}$$

where W – a set of control objects;

A – a set of control agents;

$\varOmega$ – a set of responsibility connections;

$\varLambda: A \to W$ – agent localization according to control objects;

K – information channels between the agents.

Among the many known agent architectures, InterRap architecture is considered the most suitable one in terms of facilitating agent interaction. It includes three levels of control: reactive, planned and cooperative. At the reactive and planned control levels, it is logical to use object control models that belong to the agent's area of responsibility. At the cooperative level, the presence of a local control object model may prove to be insufficient. Therefore, at this level, the agent forms its own CS model $\{W_a^*\}$ either with the help of data coming from sensors or through information exchange between agents. This requires certain actions and calculations to determine the state of control objects of the system, for example, sending requests to other agents via a specific information channel, receiving responses about the state of control objects, determining the most relevant information, etc. In the process of developing a common solution, one of the agents can take on the role of coordinator.

2.2.2. The Structure of IMS CCS

Intelligent industrial robots (IIRs) can be considered the most notable example of the IMS class. Unlike conventional industrial robots, all actions of which are determined only by the control program without any subsequent adjustment, the final actions of IIR are adjusted using perception and control units. IIR should be able to recognize and assess the environment, analyze emerging situations and adapt to the environment, make informed decisions in order to prevent the occurrence of emergency situations, model their behavior and interact with external environment, including cooperating robots. The methods of situational, adaptive and multi-agent control are most fully and comprehensively applied to IIR, determining the principles of control organization.

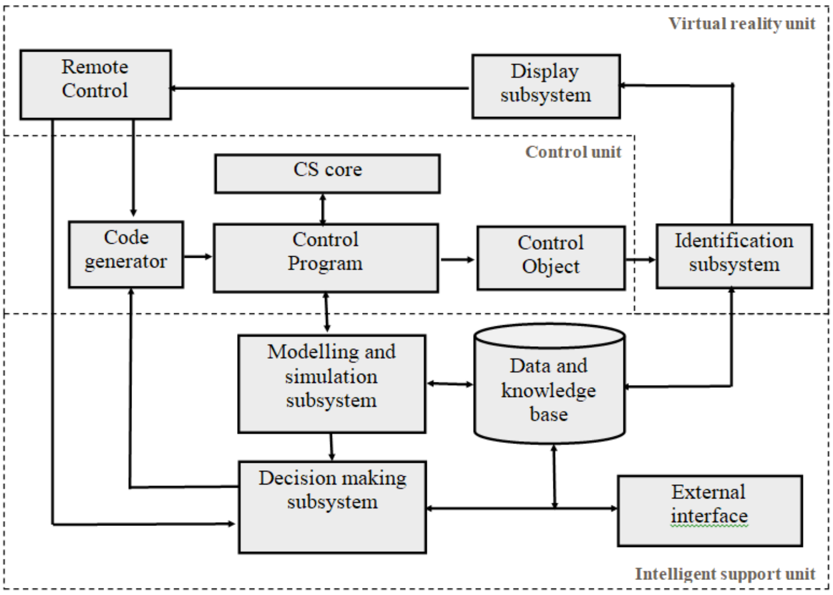


Fig. 2.6. Structure of IMS CCS.

Generally, IMS CCS can be attributed to the class of open-loop control systems, which are not covered by inverse association. They implement disturbance control, and in order to achieve invariance of the control system with respect to external conditions, it is necessary to know the exact position of the control object. This issue can be addressed by creating a CCS based on the principle of dual control. A distinctive feature of such a CCS is the dynamic nature of its behavior, which is manifested when the control program adapts to the external conditions of the control object functioning, especially by means of self-learning and self-tuning.

The structure of the IMS CCS, used to solve the aformentioned problems, is represented in Fig. 2.6.

This structure can be divided into three main components:

1. ***Control unit.*** It consists of an inalterable CCS core, which ensures the execution of control programs, and situation control programs, an alterable part of the CS, which is dynamically modified. Depending on the emerging situation, the control process is managed by means of intentionally changing the current control program via the code generator. The latter is capable of broadcasting both single commands and complex programs.

2. ***Intelligent assistance unit.*** Its functions are aimed at ensuring the adaptive properties of the control system by means of making control decisions based on the previous experience, analyzing the predicted results of current control scenario implementation and assessing the real situation. Unlike the control unit, the intelligent assistance unit is capable of producing not only individual commands, but also the entire control programs, partially or completely updating the variable part of the control system. In any case, the decision to change the control process is made by the decision-making subsystem. The modeling subsystem is incorporated into the control loop. Its task is to assess possible options of the control process development in real time, supplementing the information about the state of control object and external environment, which is sent from the identification subsystem. The generalizing component of the intelligence assistance unit is data and knowledge base, which stores and accumulates the necessary information about the parameters and properties of CO, CCS as a whole, its individual components and the results of model experiments. If necessary, information can be added to the data and knowledge base through the use of the capabilities of external, in regards to this CCS, intelligence. It can be either an adjacent CCS included in a distributed intelligent structure or global intelligent environments, for example, the Internet. The intelligent interface should ensure the intellectual openness of the IMS CCS.

3. ***Virtual reality unit.*** The main task of this unit is to affect the control system in real time in accordance with the virtual representation of the control object state and its position in regards to its surrounding. The source information for this process is the data coming from display and identification subsystems. In fact, these subsystems close the control organization through themselves. Control actions are sent from external control centers via a control channel (radio / hydroacoustic / infrared communication, remote manual control devices, etc.) to the code generator for subsequent translation into separate control commands. The influence of external control can be also extended to the decision-making system.

The discussed structure of IMS CCS can be projected onto the traditional for industrial robots control levels, specifically, strategic, tactical and executive levels. At the strategic level, production plan is drawn up, taking into account the goals and objectives of the entire factory environment. The control over the quality of the manufactured product life cycle is also carried out at this level. At the tactical level, a sequence of technological operations is formed to ensure the implementation of the received task. The executive level facilitates the direct implementation of the preassigned technological operations by means of working mechanisms and devices. It is important to keep in mind that all the aforementioned control actions are performed using the CCS.

A typical example of IMS and CPS area of application is an electron-beam welding (EBW) machine, which is turning from an experimental type of research into a powerful industry based on complex industrial technologies and a high level of the production process organization.

2.3. The Concept of Model-Oriented Control

2.3.1. The Method of Model-Oriented Control

Model-Oriented Control (MOC) is the essence of a system-based approach to control problems. In order to control the system, it is necessary to build its mathematical model. Only on the basis of the created model, the required control strategy can be developed.

However, due to a vast variety of properties of a real system, its model cannot be the exact copy of the system. Even the simplest production control situations, upon a detailed examination, appear to be far too complex. Therefore, the model should be able to describe reality with the highest possible accuracy, highlighting a limited number of variables for this purpose. Ultimately, the tasks of IMS modeling are to establish relationship between the input and the output of a system, which ensure the achievement of the set goal with a given accuracy, and to determine the dynamics of the system that would describe the real process in accordance with the previously accepted assumptions.

In this regard, computer modeling has the following advantages (Dorf and Bishop, 1998):

1. The behavior of the system can be observed under a variety of conditions.

2. By examining the model, it is possible to make assumptions about the way the system will behave in real conditions.

3. Comprehensive system tests can be performed in a relatively short period of time.

4. Modeling results can be obtained at much lower costs compared to a full-scale experiment.

5. The behavior of the system can be studied under hypothetical conditions that are unlikely to occur.

The aforementioned advantages make computer modeling the undisputed leader among other methods of system modeling, such as:

- analytical models are used at the executive level (differential equations, transfer function coefficients of linear systems and structural schemes based on the Laplace transform, including signal graphs);

- models of operation research are used at the tactical and strategic levels.

However, the goal that we set in this study is not only to consider computer models only as a means of analyzing and synthesizing CCS, but also to use their capacity directly in the control process. In this case, knowledge of the mathematical apparatus alone is far from sufficient; the task is to learn how to apply it correctly in practice.

We usually start the analysis by looking at a real situation and trying to map it onto a certain mathematical model that allows us to find a solution to the problem we are facing. The result of the analysis of the chosen model is expressed in the form of a control solution, which is then tested for optimality using an experiment that allows us to evaluate the obtained control quality. If the required quality is not achieved, the model will be parametrically adjusted or structurally reorganized, which is a more complicated process. Under this approach to the use of models in control, which is called operational, there is no guarantee that the model used will always remain relevant, i.e., adequate to the real conditions of the system functioning and, which is just as important, providing the required control quality.

The essence of MOC, which is considered a new approach to developing a management strategy for IMS (Kazymyr, 2006), consists in the widespread use

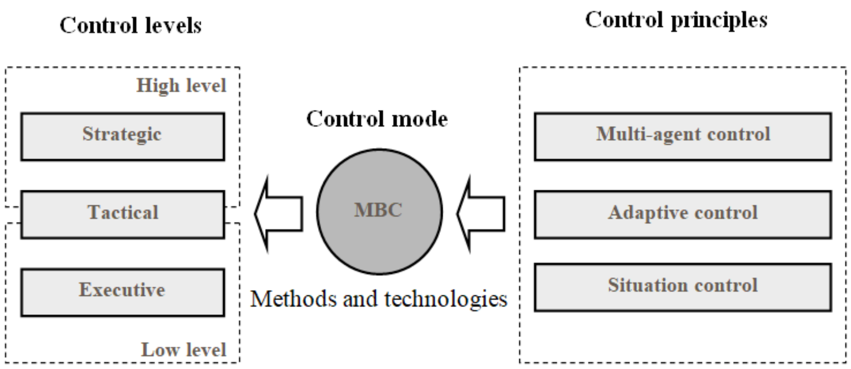


Fig. 2.7. The role and place of MOC in the IMS control context.

of computer models in the control loop directly in the process of making control decisions in real time based on situation, adaptive and multi-agent control principles. Note that in this case the MBC does not determine any new principles of control organization, but only acts as a certain way of their implementation, which combines the methods and technologies for constructing and using computer models in the control loop. Figure 2.7 shows the role and place of MBC in the control structure of IMS.

The performed analysis of the existing hierarchy, principles and features of the structural construction of IMS CCS demonstrates that control over them is facilitated by three main types of models, embedded directly into the control loop:

- **Implementation models,** which are simulation models of the control process in the state space executed by control devices. They allow us to describe and to implement a control algorithm in the dynamics of its development, taking into account changes in the state of both the control device and the CO.

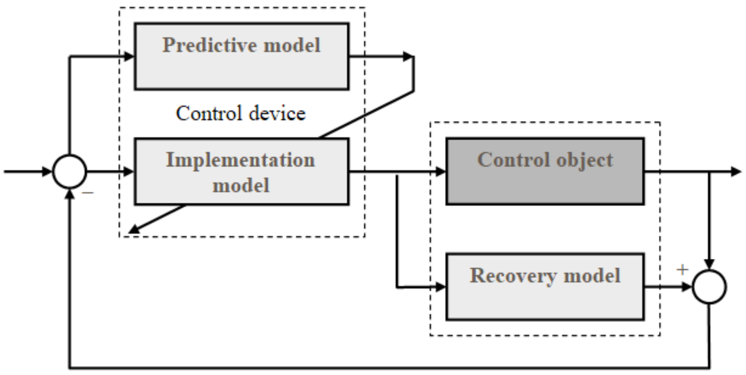


Fig. 2.8. Model-based control diagram.

- **Predictive models** assess the future behavior of the control process for satisfying the specified properties. Through this type of models, it is possible to dynamically and timely change control algorithms in order to prevent undesirable development of the control process or adjust it in the required direction.

- **Recovery models,** which solve the problem of replenishing the missing information about the CO and, in some cases, eliminate the effect of feedback lag, actually closing the feedback through the model itself.

A diagram explaining the use of these models in a control loop is shown in Fig. 2.8.

In this case, the continuity of the processes of developing and implementing control actions is ensured, which ultimately positively affects the effectiveness of control.

2.3.2. Implementation Models

There are many ways to define the notion of an implementation model. In our case, we will base the definition on the mathematical statements adopted in the general theory of systems. If $S \subset X \times Y$ and $S^{'} \subset X^{'} \times Y^{'}$are certain general systems, and $h = (h_x, h_y)$ is a homomorphism that defines the set of mappings $h_x: X \to X^{'}$ and $h_y: Y \to Y^{'}$, where h_x is surjective, then the system $S^{'}$ is called a model of the system S only if the following condition is satisfied:

$$h_x \forall (x, y)((x, y) \in S \Rightarrow h(x, y) \in S^{'}). \qquad (2.5)$$

This definition can be extended to dynamical systems as well. Moreover, if h is an isomorphism, then the systems S and $S^{'}$ are equivalent.

An important feature of the homomorphic model is that it completely preserves the algebraic structures ($X, X^{'}, Y, Y^{'}$are certain Ω -algebras) that are of particular interest to us, and allow us to neglect secondary details.

Let us construct a homomorphic model of the IS, which will include the internal and external description of a stationary system with a set of states X, a set of control inputs U, and a set of outputs Y. For the internal description, we will use the following pair of functions:

$$s_1: X \times U \to X, s_2: X \to Y\,. \qquad (2.6)$$

For the external description, the following function is used:

$$s^{'}: U^{'} \to Y. \qquad (2.7)$$

This function represents the set input sequences $U^{'} = (u(1), u(2), \ldots, u(n))$ in regards to the set of outputs Y.

The given description implies that there are isomorphisms $h_1 : X \to U^{'}$ and $h_2 : X \times U^{'} \to X$, which allow for the transition from internal description to external and vice versa. By using the property:

$$s_1(x, u(1)u(2)) = s_1(s_1(x, u(1)), u(2)), \tag{2.8}$$

with the help of the following equation:

$$s^{'}(u(1)u(2)\ldots u(n)) = s_2(s_1(x, u(1)u(2)\ldots u(n))), \tag{2.9}$$

it is possible to match the external description $s^{'}$ to any state x.

Now let us consider the inverse problem. For a given input-output function, it is required to find *implementation* $s^{'}$, i.e., a system with such a state x for which, for example, condition (2.7) is satisfied. The equation of state is now considered not as given, but as an unknown property characterizing the dynamics of the system. Thus, the *implementation problem* for the input-output dependence consists in finding the dynamics of the system, the representation of which in the state space would provide the same input-output dependence.

This problem is easily solved for a linear discrete system determined by the following equation of state:

$$x(k+1) = Ax(k) + Bu(k), \tag{2.10}$$

where A and B are *n-by-n* and *n-by-m* matrices, respectively. The phase trajectory of such a system is described by the expression:

$$x(k) = A^k x(0) + \sum_{j=0}^{k-1} A^{k-j-1} Bu(j), \tag{2.11}$$

where x – the required state. If we assume that the system starts to move from a zero state equal to zero, i.e., $x(0) = 0$, then expression (1.10) can be rewritten as follows:

$$x(k) = [A^{k-1}B, A^{k-2}B, \ldots, B](u(0), u(1), \ldots, u(k-1))^T. \tag{2.12}$$

Taking $s^{'} = [A^{k-1}B, A^{k-2}B, \ldots, B]$ into consideration, we conclude that the set of states, reachable from the zero state in k steps, coincides with the set of values of the linear transformation

$$s^{'} : U^k \to X. \tag{2.13}$$

Based on the notion of dynamomorphism, by specifying the commutativity conditions for category diagrams, solutions can be obtained for equations of state of not only linear, but also bilinear and fuzzy systems (Skyttner, 2001). However, when the equation of dynamics cannot be specified analytically and has the most general definition of $x = f(x, u)$, the task of constructing an implementation model for an input-output model is not trivial.

The same can be stated about the input-output models in the form of transfer functions that are represented as the ratio of the Laplace transform of the output parameter to the Laplace transform of the input parameter at zero-initial condition. They exist only for linear stationary systems and do not carry any information about the internal variables and the nature of their change. Therefore, they cannot be considered a general model and are used only in the design of individual elements of CCS, mainly regulators, both analog and digital. In addition, it should be noted that the control processes occurring in IMS, generally, have an algorithmic representation that is beyond the known analytical solutions.

2.3.3. Predictive Models

The implementation models discussed above are used in the design of CS with specific properties. To solve this design problem, it is necessary to possess a complete set of information about the properties of the CO and external effects on the control system. Since there are restrictions to such information in the control process, the use of traditional methods becomes insufficient and a predictive control strategy is required. Control that uses models to predict the behavior of a certain process in the future is called Model Predictive Control (MPC) (Garcia et al., 1989).

The prerequisite for the establishment of this line of research is considered to be adaptive control with the use of models, including implementation ones, self-organization of models using the group method of data handling and adaptive predictive models, which are based on the solution of Lyapunov equation.

In recent years, many variations of MPC technology have appeared, which have developed the following areas, uniting them in some way (Qin and Badgwell, 1997):

- Extended Prediction Self-Adaptive Control;
- Generalized Predictive Control;
- Model Algorithmic Control, etc.

The aforementioned technologies differ mainly in the type of models for representing the processes and the methods for solving optimization problems in the decision-making process, which may include certain types of restrictions. The most important part that unites them is the application of the Receding Horizon Strategy (RHS). The essence of this strategy is demonstrated in Fig. 2.9.

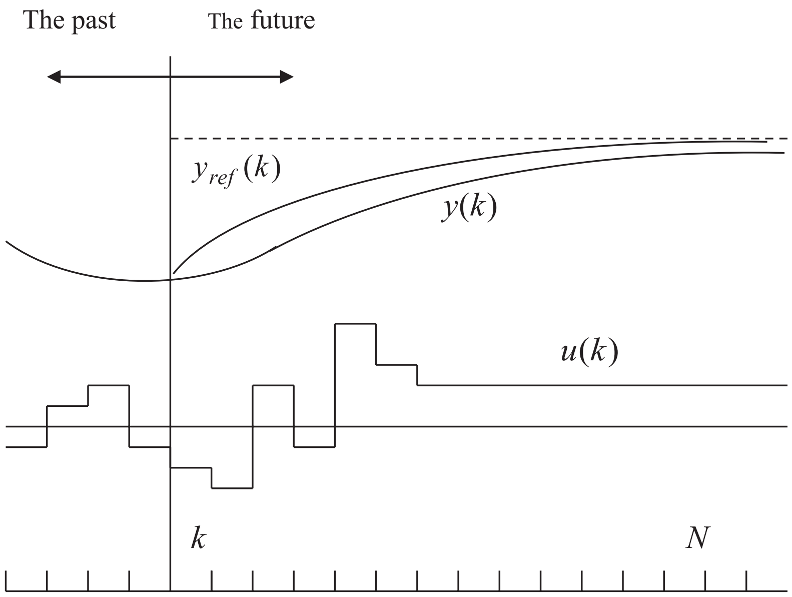


Fig. 2.9. Receding Horizon Strategy.

The main RHS features are as follows:

- At each given moment of time k, the process output $y(k+j)$ is predicted within the finite time horizon $j = \overline{1, N}$. The value of N is called the prediction horizon. Prediction is executed using a process model, which should be accessible. The projection depends on the inputs and outputs in the past, and also on the future control scenario $\{u(k+j|k), j = \overline{0, N_c - 1}\}$.

- The basic trajectory

$\{y_{ref}(k+j|k), j = \overline{1, N}\}$

is used to calculate trajectory deviation

$e(k+j|k) = y_{ref}(k+j|k) - y(k+j|k)$,

where

$y_{ref}(k|k) = y(k|k) = \hat{y}(k)$,

and $\hat{y}(k)$ is the measured output value.

- The control sequence

$\{u(k+j|k), j = \overline{0, N_c - 1}\}$

is calculated on the basis of measurements in such a way that the prediction error could be minimized.

- The first element $u(k|k)$ of the calculated optimal control sequence $\{u(k+j|k), j = \overline{0, N_c - 1}\}$,

which is applied to the real process, determines the control actions only for the step k. All other elements of the calculated control vector can be forgotten because all subsequent sampling sequences are shifted, the new output value $y(k+1)$ is measured and the whole process is repeated. This leads to the computation of a new control input $u(k+1|k+1)$, which may generally differ from the previously computed value $u(k+1|k)$.

2.3.4. Recovery Models

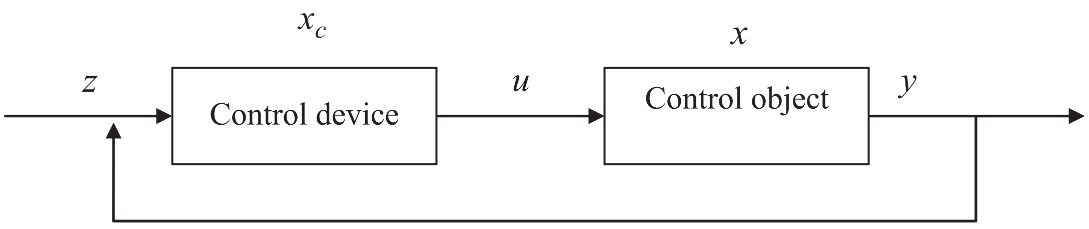


Fig. 2.10. CCS scheme with negative feedback.

The challenge facing the CCS is to find the control law that brings the CO closer to the target or keeps it close to the target. The easiest way to do this is using the feedback principle, when the output of the CO serves as the input for the control element. The CCS scheme utilizing negative feedback is shown in Fig. 2.10.

If we represent the desired input as z, the real output as y, the control input signal as u, the state vector of the control element as x_c, and the state vector of the CO as x, then the CS model with feedback can be written in the form of two pairs of equations:

- for control element:

$$x_c(k+1) = A_c x_c(k) + B_c \alpha(z(k), y(k)), \tag{2.14}$$

$$u(k) = C_c x_c(k); \tag{2.15}$$

- for control object:

$$x(k+1) = Ax(k) + Bu(k), \tag{2.16}$$

$$y(k) = Cx(k). \tag{2.17}$$

The function $\alpha(z(k), y(k))$ in Eq. (1.14) establishes the type of the target relationship between the desired system output and its actual output. In case of negative feedback:

$$\alpha(z(k), y(k)) = z(k) - y(k). \tag{2.18}$$

Applying Eq. (1.15) to Eq. (1.16), we will get:

$$x(k+1) = Ax(k) + BC_c x_c(k). \tag{2.19}$$

Subsequently, by using Eq. (1.14), we will get:

$$x(k+1) = Ax(k) + BC_c(A_c x_c(k-1) + B_c \alpha(z(k-1), y(k-1)). \tag{2.20}$$

Equation (2.20) reflects the fact that in order to calculate the state of the CO at step $(k+1)$, it is required to know its state at step k, as well as the state of the control element, the desired and actual values of the output at step $(k-1)$. This means that in the process of determining the required control, the control element lags behind the current state of the CO. The larger the sample spacing becomes, the greater delay grows. Therefore, for discrete closed-loop systems, there is no direct transition from $u(k)$ to $y(k)$ in the output expression, which would allow the model to be strictly correct. In addition to the aforementioned fact, the drawbacks of feedback include an increase in the complexity of implementation and a decrease in the gain ratio.

However, the main condition for control with feedback, which is difficult to be observed for the objects of any complexity, is that the current values of the state variables or the output variable, at the time when the control action is applied to them, are assumed to be known. A more realistic situation is when not all of the state variables can be measured. In this case, the "recovery methods" are used, which are implemented with the help of the observers.

Generally, the role of the observer is taken by another dynamic system, which is able to restore the state vector of the observed system using its input and output values. In fact, the observer imitates the controlled system. For linear continuous and discrete systems, there are analytical methods for solving the recovery problem, which provide for rather strong restrictions imposed on the structure of the observer. However, for nonlinear systems, solutions become less trivial, if not impossible. Naturally, at some stage, it is possible to use predictive models that allow for obtaining some approximation of the system state. However, it should be kept in mind that predicive models do not add information about the current state of the CO; on the contrary, they need this information to improve prediction quality. In this regard, there is a need to build computer recovery models that will be able to reproduce the state of a CO of the most complex structure, using the capabilities of computer modeling. In this case, the CO and its recovery model are considered to be a single entity.

2.4. Summary

In recent years, a class of intelligent industrial systems has been formed that brings high-quality intellectual definitions to each of the properties inherited from FMS as well as CIM, improving the automation of technological processes and enhancing the level of information integration for manufacturers. The use of computer models in the industrial system control is the main factor that determines the intellectual aspect of IMS.

The analysis of control principles that are applied to industrial systems has revealed that the basic principles for IMS CCS are the principles of situation, adaptive and multi-agent control, which relate to control organization and determine its strategy. The structure of IMS CCS is proposed as a means of implementing these principles; the proposed structure can be used at all control levels, including ERP and technological CCS.

Instead of the traditional operational approach to the use of analytical models and methods of operation research, a new model-oriented approach is proposed for the implementation of a particular control strategy for IMS. This approach requires the use of computer models embedded directly into the control loop, which can dynamically change according to the conditions of the system functioning, while constantly remaining relevant from the control point of view.

At the conceptual level, computer models, used in IIS control loop, can be divided into three main types:

- implementation models, which simulate the control process executed by control devices and, at the same time, set the control algorithm taking into account changes in the state of both the control device and the control object;

- predictive models, which allow for assessing future behavior of the control process in order to satisfy the specified properties and provide timely dynamic changes in control algorithms for the purpose of preventing undesirable development of the control process;

- recovery models that solve the problem of recovering the missing information about the CO and eliminating the effect of feedback lag, basically, closing the feedback through the model itself.

The use of the aforementioned models in the control loop ensures the continuity of development and implementation of control actions, which ultimately has a positive effect on control efficiency.

The analysis of existing formal methods, which can be used for the creation and use of embedded models, has revealed that they are generally based on linear models inherent in systems with a single control level. At the same time, actual processes that occur in ISS usually have an algorithmic definition and demonstrate dynamic and structural phenomena that require coordination of decisions at different control levels.

Currently, there is an issue of Model-Oriented Control over IMS, which consists in the development of methods, technologies and software for creating and using computer models in the control over CCS with a complex structure and behavior pattern. The defined problem presupposes the solution of the whole range of theoretical and practical problems, which can be grouped into several categories related to the use of implementation, predictive and recovery models in the IMS control loop.

Chapter 3. Implementation Models of Control Algorithms

This chapter outlines the existing methods for specifying control algorithms, taking into account the structure and functioning process of the IMS CS. Furthermore, the grounds for the choice of a general mathematical scheme for describing control algorithms in the form of an aggregative system are given. A class of modified E-networks, which are called Control E-Nets (CEN), is defined, and the structural and functional features of CEN are described. A study of CEN as a means of describing piecewise linear aggregates is carried out. At the end of the chapter, the functional completeness of the mathematical apparatus of CEN is analyzed in relation to the informal theory of sequential interacting processes.

3.1. Control Algorithms and Methods of their Description

3.1.1. Implementation Model Requirements

Control algorithms form the basis for the functioning of control devices that play the role of control elements in modern control systems. Generally, a control algorithm (CA) is defined as a clear, unambiguous rule, an instruction or an indication of what actions should be done and how to do them to achieve a given goal in the current situation. CA, also known as control law, determines the development and implementation of control actions.

Any algorithm implements some kind of a control process. Therefore, the more accurate this implementation is, the closer to the set management goals the CCS will function. This is the reason behind many examples of the use of formalized schemes to describe algorithms that, to a certain degree, could model a control process. Among the most notable of them, we should single out Logical Scheme of Algorithms (LSA) and Matrix Scheme of Algorithms (MSA), as well as their varieties: Parallel LSA (PLSA), Parallel MSA (PMSA) and Parallel Graph-Scheme of Algorithms (PGSA) (Baranov, 1994).

It should be noted that all of the listed approaches to the description of algorithms, in some way, implemented in the automaton model of formalized description were proposed by V.M. Glushkov. Later, the automaton model was widely used in PLC when creating PLC programming languages, such as Instruction List (IL), Structured Text (ST), Ladder Diagram (LD), Functional Block Diagram (FBD), Sequential Function Diagram (Sequential Function Chart – SFC) (Dixon, 2018).

Similarly to other automata models, PLC languages provide an advanced apparatus for describing discrete systems and processes, but do not reflect cause-and-effect relationships at the level of internal processes. Systems with parallel functioning and asynchronously interacting components (which is especially important for distributed systems) are not adequately described in terms of classical automata theory. In addition, the composition of models and their hierarchical representation within the framework of this apparatus are significantly complicated. Furthermore, it should be pointed out that all PLC programming languages are deprived of the possibility of performing any formal analysis of described algorithms, and also do not allow for a dynamic change in the control program during its execution, since they do not trace the relationship between the mathematical basis and the specification language.

Therefore, it becomes necessary to use other, more powerful formal methods for describing control processes. When considering these methods, we will take into account the following requirements for CA implementation models:

4. CA implementation models must comprehensively represent the dynamics of control process development, taking into account parallel and asynchronous functioning of control elements. At the same time, mechanisms for synchronizing their work should be accounted for, if necessary.

5. The models used should provide formal description of the hierarchical relationships between control levels in IMS CCS. This means that the CA implementation models of various levels should allow for the use of information signals to solve the problems of coordination and interaction synchronization.

6. The used CA models are required to allow for the identification of situations requiring control decisions and provide operational influence of control elements by changing the values of state variables.

7. CA implementation models must also be the specification of the control program that can be executed by the control device. This requirement makes it possible to implement a continuous cycle of using the same models both at the design stages of CA and in the process of their application.

8. CA implementation models must be based on a formal system that allows for early preventive acquisition and evaluation of the process development protocol in order to dynamically change CA during the implementation of situational, adaptive and multi-agent control principles. This requirement should be based on the possibility of widespread use of recovery and predictive models within the accepted formal system for constructing CA implementation models.

3.1.2. Methods of Control Process Formalization

As mentioned in Chapter 1, due to their particular complexity, control processes of IIS cannot be described exclusively in the categories of functional relations between individual parameters and variables, for example, using differential or difference equations. Therefore, an algorithmic approach with greater flexibility should be considered the main method for constructing IIS implementation models. Within the framework of this approach, three main groups of methods can be distinguished: algebraic, network and hybrid.

Some of the earliest examples of algebraic methods are abstract formal process definitions in the form of a trace structure (Kaldewaij, 1986). In this approach, the alphabet of the process is a set of events, and the study of algebraic properties is carried out using the lattice theory. This approach was not properly developed, giving way to formalism that more comprehensively takes into account parallelism and the time factor. However, the notion of traces is still used today to construct the proof for the properties of parallel processes.

Among other options for a formal definition of a process in terms of events, one should highlight the study by Janicki and Lauer (1992), in which it was attempted to examine the system of parallel processes using the formal apparatus COSY (Concurrent System). Although the COSY specifications are intended to describe parallel processes taking into account synchronizing aspects, they nevertheless ignore such important requirements for industrial system control algorithms as hierarchy and dynamic program change. However, it should be noted that it was in the above work that the question of program verification in the dynamics of its execution was first raised.

The most actively used process algebras are the Hoare calculus (communicating sequential processes) (Hoare, 1985) and the calculus of interacting Milner systems (Calculus for Communicating Systems) (Milner, 1989). Process algebras provide advanced sets of operations and syntax analysis methods. At the same time, they are not able to display “true parallelism”, which is the result of the partial ordering of events, while the analysis of dynamic capabilities is based on operational rules being more difficult and inconvenient compared to network methods.

An example of the use of algebraic specifications, for the purpose of describing dynamic processes, is mutating algebras, or Abstract State Machines, proposed by Gurevich (Gurevich, 1994). There are directions for development of ideas of Gurevich machines, connected with object-oriented data representation (Asteziano and Zucca, 1995) and dynamic algebra corresponding to the state of a dynamical system. However, this approach is closer to defining the semantics of programming languages rather than to describing control systems. Nevertheless, some of the ideas can be used for analyzing the dynamic properties of control processes.

The analysis of theoretical process models can be supplemented by a whole group of formal systems designed to describe parallel computations. This group distinguishes completely abstract concepts of control spaces and algebraic programming, partially abstract models of bulk-synchronous parallel (BSP) processes (Valiant, 1990) and LogP (Culler et al., 1993), models with a limited form of Bird-Meertens parallelism (BMF) (Bird, 1993) and pipeline computation models based on the algebrodynamic approach. However, the listed models are simply the effective means of increasing the performance of parallel programs, rather than the basis for constructing high-level descriptions of CA.

The network approach in the formalization of control processes deserves special mention. The application of network formalization methods in the field of industrial automation is the subject of many scientific publications. Among the most mathematically developed formal network models of processes are Petri Nets (PNs) (Reising, 1985; Brauer, 1987) and their extensions: temporal (Zuburek, 1980), colored (Jensen, 1981), predicate, high-level Petri nets (Kramer and Schmidt, 1991). PNs effectively reflect the parallelism and logic of control processes, taking into account asynchronous interactions. However, although PNs served, in their time, as a prototype for the creation of the PLC programming language, in the practice of describing control systems they received limited application due to the lack of the ability to quantitatively process data during network transitions and the difficulties of controlling the routing of process development.

The most powerful PN extension that removes the restrictions noted above are the Evaluation Nets (E-Nets) or E-networks, and their modifications (Nutt, 1972). Possessing all the capabilities of temporary PNs, E-networks are able to display not only control flows, but also data flows, giving the grounds for considering them as the basis for constructing CA implementation models. In addition, E-networks significantly surpass other network methods in the implementation of logical functions and form a universal algorithmic system equivalent to a Turing machine. Thus, E-networks not only allow describing algorithms of any complexity, but also have the ability to dynamically restructure them.

There are several interpretations of PNs from the class of E-networks (Pro-networks) (Noe, 1980), macro-E-networks (Beyaert et al., 1981), the development of which is associated with an increase in the descriptive power of this apparatus. However, due to the complexity of the analysis of E-networks directly in the field of CS modeling, there has been a departure from the pure theory of E-networks towards the use of various other PN extensions: predicate-temporal, loaded, control and hardware networks, which are significantly inferior in terms of structural expressiveness.

Network methods of formalizing control processes are not limited to the use of Petri nets and their extensions to describe technological processes. Suffice it to say that the earliest applications of network models were associated with the construction of network diagrams for scheduling purposes. The most advanced in this regard are PERT networks and GERT networks (Phillips and Garcia-Diaz, 1990). However, due to the limited logical capabilities, their application excluded the use of decision-making elements, which are important from the point of view of control tasks.

Models of distributed computing, which imply the use of a certain specification language based on the representation of CA in the form of a graph, can also be considered network formalization methods. They can be divided into traditional (modular) and object models (Kerzner, 2003). Among the first, one should single out the general model, the components of which are described in the language of Z-scheme specifications, the system model in the form of an acyclic graph formally created using the compositional theory, the formal model of a modular structure that defines connections and relationships between modules using the specifications of the assembly programming theory. All these models have a clear practical orientation associated with the construction of distributed programs and their interfaces, but do not have a mathematical apparatus for analyzing the properties of the described processes.

Hybrid models lay the foundation for a unified mechanism for the algorithmic description of processes at the executive, tactical and strategic levels. Examples of this approach provide a combination of continuous and discrete components. But most importantly, they form a mathematical scheme that allows for combining different models within a generalized structural representation.

The most notable hybrid models are the aggregative approach (Buslenko, 1978), the continuous-discrete model based on the discrete event approach implemented in the continuous-discrete system (CDS) modeling system, and A. Pnueli's transition system (Henzinger, 1993). It should be noted that all these models can be presented in terms of each other. However, we will be interested in the aggregative approach, since a clear mathematical basis for setting multilevel descriptions inherent in the structure of the CCS can be built for it. As for Pnueli's transition system, this model, used in verification systems based on temporal logic, also fits into a generalized aggregate scheme and is actually implemented within the framework of the CA formalized description method, which is developed in the book.

Analyzing the existing approaches to the description of hybrid systems, one cannot fail to note the emerging trend of using artificial neural networks (Gomi and Kawato, 1993; Hagan et al., 2002) and genetic algorithms (Tajima, 1996; Wang et al., 2003) in solving control problems. Allowing for the implementation of a function of any complexity, these methods are mainly used in CS to construct control devices of different variants. However, neural network control devices often have unacceptably long training time and are still used mainly as expert systems. As for genetic algorithms, they, like all evolutionary methods, should be applied in cases that are difficult to formalize or when a rough estimate is required for making decisions in real time. Currently, there is a search for ways to improve the efficiency of these methods, in particular, fuzzy logic (Rajashekaran and Vijayalksmi, 2004).

3.1.3. The Tasks of Control Algorithm Description

To obtain a formalized description of the CA of a distributed CCS, it is recommended to use the methodological concept of aggregation, which is based on the set theoretic interpretation of a model: a model is a set, between the elements of which certain relations are specified.

If we consider only two levels of CA interpretation (where macromodel is a structure consisting of separate control elements, and a micromodel is the processes of functioning of these elements), then, in the accepted concept of aggregation, the abstract formal model of CA can be defined as follows:

$$DCS = (\Sigma, E, \gamma), \tag{3.1}$$

where Σ – CA structure represented by a graph with the set of control elements A and the set of arcs $\Gamma \subseteq A \times A$;

E – a finite set of control processes that are implemented by the elements A;

$\gamma : \Sigma \rightarrow E$ – aggregative representation, which unites formal definitions Σ and E, thereby specifying the distribution of functioning processes in the structure of system elements.

It is possible to use the theory of aggregates as a formal system providing the construction of macromodels, and the E-network as the basis for constructing micromodels that reveal the dynamics of the aggregate behavior. This way, the problem of developing a method for a formalized description of CA implementation models will include the solution to the following problems:

- modification of the mathematical apparatus of E-networks in order to use them to control industrial processes;

- development of aggregating visualization that allows using E-networks as a means of describing the dynamics of aggregates;
- analysis of the properties of the modified E-networks as a formal system aimed to describe the behavior of the IIS control elements.

3.2. Formal Definition of Control E-network

3.2.1. Control E-network Structure

A control E-net (CEN) is defined as a set of five elements:

$$CEN = (P, T, F, V, U, M_0), \tag{3.2}$$

where $P = \{P_S, P_R\}$ – a finite non-empty set of places consisting of disjoint subsets P_S (simple places) and P_R (decisive places), $P_S \cap P_R = \emptyset$; a set of simple places can contain a subset of input places $P_{in} \subseteq P_S$ and a subset of output places, which are called limits, and it is assumed that $P_{in} = \emptyset$ and $P_{out} = \emptyset$, $P_{in} \cap P_{out} = \emptyset$;

T – a finite non-empty set of transitions, which can consist of transitions of five types {"T_T", "T_F", "T_j", "T_X", "T_Y"}, the so-called ordinary E-networks [101], and two types of additional transitions-queues "T_{QL}" and "T_{QF}", $T \cap P = \emptyset$;

$F: P \times T \cup T \times P \to \{0,1\}$ – an incidence function;

$V = V_I \cup V_O$ – a finite set of network variables consisting of disjoint subsets V_I (input) and V_O (output) signals $V_I \cap V_O = \emptyset$;

$U = \{r, \sigma, \alpha, \tau, z\}$ – a set of control mappings defining transition firing rules;

$M_0: P \to \{0,1\}$ – an initial marking function that specifies the presence or absence of tokens in places.

The CEN structure is equivalent to an oriented bipartite graph, where one set of vertices is P, the other is T, and the arcs between the vertices of the two named sets are determined by the incidence function F. From the definition of the incidence function, it follows that CEN is an ordinary network – there are no multiple arcs between places and transitions.

Let us associate each transition $t \in T$ with a set of its input places $\bullet\, t = \{p \in P | F(p,t) = 1\}$ and a set of output places $t\, \bullet = \{p \in P_S | F(t,p) = 1\}$, and denote the entire set of places incident to the transition t as t as $P_t = \bullet\, t \cup t\, \bullet$.

The following limitations are imposed on the CEN structure:

- for any two transitions $t_i, t_j \in T$ where $i \neq j, t_i \bullet \cap\ t_j \bullet = \emptyset$ takes place; i.e., a certain place cannot be the output place for two or more transitions at the same time;

- for any two transitions $t_i, t_j \in T$ where $i \neq j, \bullet\ t_i \cap \bullet\ t_j = \emptyset$ takes place; i.e., there is no place that can be the input place for two or more transitions at the same time;

- for any transition $t_i \in T$ and any place $p_j \in P$, $P_t \neq \emptyset$ and $p_j \in \bullet\ t_i \cup t_i \bullet$ take place, i.e., isolated vertices do not exist in the CEN structure;

- for any transition $t_i \in T, P_R \cap t_i \bullet = \emptyset$ takes place, i.e., the decisive place cannot be the output place of the transition and, therefore, an arc cannot be a part of it;

- for any place $p \in P_R$, $F(p, t_j) = 1$ takes place, where t_j is of a type T_x or T_y, i.e., decisive places are only associated with transitions of a specific type.

Ordinary E-networks, the application of which was focused only on modeling problems, were, in fact, autonomous, i.e., they did not interact with their surroundings. In control E-networks that take on the role of CA implementation models, all actions performed by the network must be consistent with the current state of the CO and the external environment. By external environment, we will understand the CEN, which can interact with the given network by passing tokens through the boundary input P_{in} places. Interaction with CO is performed via the corresponding network variables V. Furthermore, for discrete input signals, the designation DI is used, for analog input signals – AI, discrete output – DO, analog output – AO, in the way that $V_I = DI \cup AI$ and $V_O = DO \cup AO$. Discrete signals can take values from the set {0,1}, and analog signals can take values from the set $\Re$ of real numbers.

3.2.2. Dynamics of Control E-networks

The dynamic properties of a network are determined by changing its token and depend on the values of the control mapping components.

As a token in a control E-network, we define a vector M=$(M(p_1), M(p_2), \ldots, M(p_n))$, where $n = |P_S|$. Place $p_i \in P_S$ is called free (does not contain a token) if $M(p_i) = 0$; otherwise, in case of $M(p_i) = 1$, the place is considered occupied. For a given token M, the set of marked places will be determined as $P_M = \{p \in P_S | M(p) > 0\}$.

For all CEN places, $M(p_i) \leq 1$ is fulfilled, which translates CEN into a class of safe nets, which are now commonly called Condition / Event-Net (C/E-Net) Petri nets (Reising, 1985). Note that in ordinary E-networks, in addition to simple places capable of storing only one token, the existence of queue places of infinite capacity is allowed reducing the security condition, which, as we will see later, turns out to be important from the point of view of the network functional properties.

Similarly to ordinary E-networks, each token located in the CEN place is assigned a descriptor, or a tuple of numeric attributes, which determines the information content of the token $d_i = (d_{i1}, d_{i2}, \dots, d_{ij}, \dots, d_{iN})$, where d_{ij} is the value of the attribute j which belongs to token i. As tokens move across the network, their attribute values may change.

When the network is running, the tokens can move from input transition places to output ones, changing the marking of the network. Since the number of CEN places is finite, the number of its possible markings is also finite and equal to $2^{|P_s|}$, including the initial marking $M_0 = (M_0(p_1), M_0(p_2), \dots, M_0(p_n))$.

As the CEN attainability set, we mean the finite non-empty set RS of all markings attainable from the initial marking M_0, including the initial marking, i.e., $RS \subseteq M$. An attainability graph of CEN is a graph $RG = (RS, G \subseteq RS \times RS)$ that includes attainable markings as vertices. The arcs $g \in G$, where $g = (M_i, M_k)$, show that the marking M_i is directly reachable from the marking M_k.

The structural component of the control E-network, which determines its dynamics, is a set of control mappings $U = (r, \sigma, \alpha, \tau, z)$, which includes five functions associated with network transitions:

- r – a decision transition function;
- σ – a function of transition firing readiness;
- α – a transition trigger function;
- τ– a transition delay function;
- z – a transition transformation function.

Decision function

$$r: P_R \to \{1,2,3, \dots\} \tag{3.3}$$

is associated with decision places that do not contain tokens and control the operations of associated transition types T_x and T_y by calculating the values of the so-called decision functions $r: P_R \to \{1,2,3, \dots\}$. The decision function can be calculated, taking into account the values of the attributes of tokens and network variables, i.e., $r(q) = f(d_p, V_I), q \in P_S, p \in P_R$. The decision function value determines the direction of movement of the token when the transition is triggered. The limits of possible values of the decision functions depend on the default number of places incident to the transition $r_0(q) = 1$.

Transition readiness function is the predicate

$$\sigma: T \to \{0,1\}, \tag{3.4}$$

which determines transition firing readiness: if $\sigma(t) = 0$, then the transition $t \in T$ to operation is not ready; otherwise, if $\sigma(t) = 1$, then the transition t is ready to firing. Each transition type has its own definition of the readiness function. The value of predicate (3.4) depends on the marking of simple places incident to the transition, as well as the value of the decision place of the transition, if such a place exists, and is calculated each time the marking of the network is changed. Thus, $\sigma(t) = f[M(p), r]$, if $r \in \{\bullet\, t, t\, \bullet\}$ for transitions, where $p \in \{\bullet\, t, t\, \bullet\}$. The markings of the input and output places, at which firing of transitions takes place, will be called admissible.

The trigger function is absent in the definition of transitions of an ordinary E-network. Its use in CEN is caused by the necessity to take into account the state of CO when determining the conditions for firing of the transition in addition to the analysis of the admissible marking. The activation function is the predicate

$$\alpha: T \to \{0,1\}, \tag{3.5}$$

which is calculated for each transition and determines the possibility of its triggering: if $\alpha(t) = 0$, then the transition $t \in T$ remains inactive; otherwise, if $\alpha(t) = 1$, then the transition t is triggered. When calculating the trigger function, the values of the network input signals are calculated, i.e., $\alpha(t) = f(DI, AI)$. By default, the trigger function is equal to 1, and the transition is triggered for any values of the input signals.

The delay function calculates the transition τ delay time based on the token attribute values located in the network places, as well as the values of the network variables, i.e., $\tau(t) = f(d_p, V_I), p \in P_S$. As a special case, the default delay time can be set to zero. In general, the delay function can be represented as a mapping

$$\tau: T \to \Re^+, \tag{3.6}$$

where T – a set of network transitions;

$\Re^+$ – a set of positive real numbers, which includes zero.

Transition transformation function

$$z: T \to \delta \tag{3.7}$$

specifies the sequence of operations $\delta: \{d_p \cup V\} \to \{d_p \cup V\}, p \in P_M \cap P_t$ that are performed on network variables and token attributes as they are moved from input places to output places of a transition. Setting the default transform function z_0 does not change the token attribute values.

Taking into consideration the control mappings, the execution of any transition $t \in T$ includes the sequential passing of the following four phases:

- readiness, when the transition is not at a delay phase and the condition of its firing $\sigma(t) = 1$, determined by the specific type of transition, is fulfilled;
- activity, when the readiness phase started and $\alpha(t) = 1$;
- delay, when the countdown, until the transition firing, began; the phase duration is determined by the transition delay time τ (t), which must be calculated before the beginning of the delay phase; the state of the transition places does not change until the end of the delay phase;
- firing, when, after the expiration of the delay phase, an instant change of the transition place marking occurs by moving the tokens from their input places to the output ones in accordance with the firing rules for transitions of this type; at the same time, the values of the token attributes placed in the output places are changed in accordance with the specified transition transformation procedure.

Dynamic properties, which are traditional for E-networks, determined by the ability of tokens to move between places and by transition firing rules, are expanded in control E-networks due to the possibility of dynamically changing the control functions of transitions. The decision, trigger, delay and transformation functions are functions of time that can change during the execution of a network.

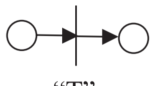
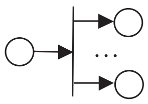
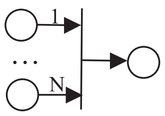
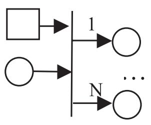
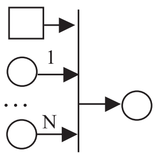
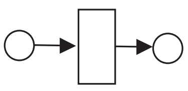
3.2.3. Basic Set of CEN Transitions

Control E-networks preserve the basic set of transitions of ordinary E-networks, which is expanded by the introduction of queue transitions that perform the functions of queueing tokens with different service discipline. This makes it possible to use CEN in order to implement the possibilities inherent to Petri nets of the Place/Transition-Net (P/T-Net) type.

Description of the firing schemes for all types of CEN transitions is given in Table 3.1.

For the basic set of ordinary E-network transitions, the specifications described in Reising (1985) are generally preserved. Therefore, we will dwell only upon the characteristics of the additional types of transitions-queues introduced as an extension of the CEN basic set. There are four types of transitions-queues, which are implemented within the two main types of these transitions:

Table 3.1 Basic Set of CEN Transitions

Transition graph	Condition $\sigma(t)=1$	Firing scheme
 "T"	$(M(x)=1)\wedge(M(y)=0)$ $x\in \bullet t, y\in t\bullet$	$(M'(x)=0);$ $(M'(y)=1);$ $z_0 : d_{yj}=d_{xj}$
 "F"	$(M(x)=1)\wedge$ $\forall y\in t\bullet:(M(y)=0)$	$(M'(x)=0);$ $\forall y\in t\bullet:(M'(y)=1);$ $z_0 :\forall y\in t\bullet: d_{yj}=d_{xj}$
 "J"	$\forall x\in \bullet t:(M(x)=1)\wedge$ $(M(y)=0)$	$\forall x\in \bullet t:(M'(x)=0);$ $(M'(y)=1);$ $r_0 : d_{yj}=d_{x1,j}$
 "X"	$(M(x)=1)\wedge$ $(M(y_r)=0)$	$(M'(x)=0);$ $(M'(y_r)=1);$ $z_0 : d_{y_r,j}=d_{xj}$
 "Y"	$(M(x_r)=1)\wedge$ $(M(y)=0)$	$(M'(x_r)=0);$ $(M'(y)=1);$ $z_0 : d_{yj}=d_{x_r j}$
 "QF","QL"	1) $(M(x)=1)\wedge(M(y)=0)$ 2) $(M(x)=1)\wedge(M(y)=1)$	1) $M'(x)=0; M'(y)=1$ 2) $M'(x)=0; M'(y)=0;$ $M'(y)=1$ $z_0 : d_{yj}=d_{yj}$

- the QF transition implements the FIFO (first in, first out) token service order and priority service in ascending order;

- transition-queue QL implements the LIFO service discipline (last in, first out) and priority service in descending order.

Firing of the transitions-queues can take place in two ways: the first is when a token arrives at the input place x, the output place is free and the second way is when the output place is occupied. In the first case, the transition is in the state of constant readiness to work, since the only condition for its firing is the presence of a token in the input place: $M(x)-1$. When the transition starts, the token is placed in the queue, while when the queue is empty and the output place is free, the token immediately takes the place, which means that the rule of moving tokens is executed:

$$M'(x)=M(x)-1, M^{'}(y) = M(y) + 1 \tag{3.8}$$

If, at the moment a token enters the queue, the output place is already occupied by one of the tokens in the queue, the rule for moving tokens will be as follows:

- the token from input place x is placed in the queue;
- the token from output place y is placed in the queue;
- a token selected in accordance with the accepted queue servicing discipline is placed in the output place from the queue. As a result, the sequence of operations will look like this:

$$M'(x)=M(x)-1; M^{'}(y) = M(y) - 1; M^{'}(y) = M(y) + 1 \tag{3.9}$$

It should be noted that whenever an output place y is freed, a token from the queue (if there are any) is placed in it according to the service discipline selected for this transition.

There are the following limitations associated with the use of transition queues:

1. The transformation procedure is executed when the token is placed in the input place.

9. The time delay function is not defined for transition queues.

10. For these transitions to work properly, transitions with zero delay time must be in the same structural bond as their output places. Otherwise, transition firing rules may be violated.

A control E-network is traditionally depicted in the form of a graph, in which circles stand for simple places, squares – for decisive places, while transitions are represented by vertical lines. Transition-queues are denoted by rectangles with a bar: a vertical bar placed closer to the output indicates a transition-queue of "QF"type, a bar put closer to the input – "QF", transitions-queues with priorities are denoted by a rectangle with a corresponding diagonal. Links between places and transitions are represented by directed arcs. The boundary (both input and output) places of the network are additionally marked with triangles associated with them. The presence of a token in a simple place is indicated by a dot. In the attainability graph, the vertices in the form of ellipses correspond to the attainable markings of the network, and the arcs between them are marked with the index numbers of transitions, and as a result of firing of these transitions, a specific marking is achieved.

Figure 3.1 demonstrates an example of CEN and the corresponding attainability graph.

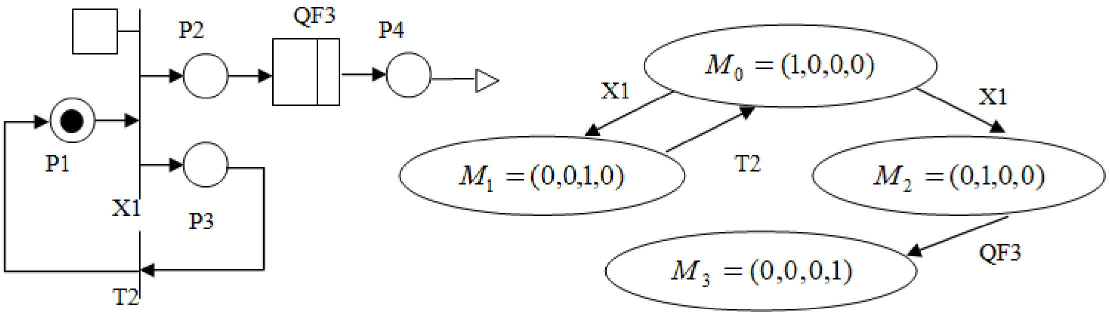


Fig. 3.1. CEN graphical representation and its attainability graph.

3.2.4. CEN Semantics

Let us give semantic definitions to CEN components in the aspect of CA description. These definitions are based on the principle of situation control. Following this principle, it is necessary to define the concepts of the state of both CO and CA, the current situation in the control system, as well as the control decisions that affect the changes of the current situation.

Definition 3.1. The current situation taking place in a conrol system is a system of sets denoting the states of CO and CA at the present moment of time.

Definition 3.2. CO state is a set of all available information about the CO displayed in the set of input signals of the network $V_I = DI \cup AI, V_I \subset V, DI = \{DI^1, \ldots, DI^{|DI|}\}, AI = \{AI^1, \ldots, AI^{|AI|}\}$ at the present moment of time.

Definition 3.3. CA state is the current marking $M_i \in M$ of the control E-network with the corresponding token attribute values.

Consequently, the whole set of situations that can be described by a certain CEN is determined by the following expression:

$$S \subseteq \{DI\} \times \{AI\} \times M, \tag{3.10}$$

where $\{DI\}$ – discrete input signal space;

$\{AI\}$ – analog input signal space;

M – network markings set.

and current situation $s_i \in S$ is a vector

$$s_i = (DI_i, AI_i, M_i), \tag{3.11}$$

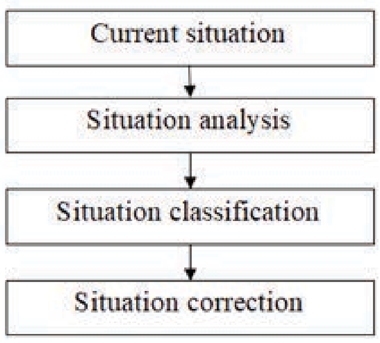
where DI_i – a vector of input discrete signal values in t situation;

AI_i – a vector of input analog signal values in t situation;

M_i – a vector of network place markings in t situation.

The aforementioned definitions (3.4–3.6) reveal the semantic meaning of CEN places, the marking of which sets the current state of the CA, which together with the input signals determines the current situation.

Next, let us discuss the semantics of CEN transitions, for the purpose of which we will use the situation control scheme proposed in [29]. In the interpretation of control E-networks, this scheme can be represented as shown in Fig. 3.2.



$$s_i = (DI_i, AI_i, M_i), s_i \in S$$

$$\{t \mid \sigma(t) = 1 \wedge \alpha(t) = 1\}$$

$$r(p), p \in P_U, p \in \bullet t$$

$$M \xrightarrow{t} M', z(t)$$

Fig. 3.2. The scheme of situation control method by means of CEN.

In the process of their firing, network transitions sequentially perform the following functions related to the implementation of the principle of situational control for the current situation:

1) *Analysis function* takes place in case the readiness of transitions to firing under

condition $(\sigma(t) = 1) \wedge (\alpha(t) = 1)$ is checked. If this condition is met, it means that there is a situation that requires intervention of the control system in the process. Otherwise, the processing of the situation does not start.

2) *Classification function* is triggered when the variant is determined, according to which the transition is fired taking into account the values of decision functions $r(p), p \in P_R$. At the same time, transitions refer the current situation to a certain class of decisions.

3) *Correction function* is executed after the delay phase, when transition firing takes place. During this process, the marking of the network places is changed and the CO is affected by setting the values of the output signals $V_o = DO \cup AO$ using the conversion procedure $z(t)$.

Functions $\sigma(t)$, $\alpha(t)$ and $z(t)$ can be combined in a single operator $A(t) = f[\sigma(t), \alpha(t), \phi(t)]$, which will indicate a certain action associated with firing of transition t. Consequently, the result of performing an elementary control step, implemented according to the scheme shown in Fig. 3.2, can be represented as follows:

$$S_i \xrightarrow{A(t)} S_j. \tag{3.12}$$

In addition to the introduced semantic definitions, there are the following fundamental features of CEN, which are important in terms of demonstrating their functionality:

1. *Locality.* CEN retains the principle of locality inherent to Petri nets. This means that changes in place markings have only a local effect on the further behavior of the network, since place markings only affect the transitions directly associated with them.

11. *Direct impact*. CEN does not use qualifiers, such as those available in SFC, that implement impulse, constrained, deferred, or persistent actions. The listed qualifiers can be implemented in CEN by means of designing an appropriate network structure using transition delays. In contrast to CEN qualifiers, similarly to Moore machines, output signals are a direct function of current states, which are represented by situations in this study.

12. *Recurrence.* CEN is executed in a cycle, so that all conditions provided by the transition patterns and activation functions are relevant to the current network cycle. The generated output signals are not initialized until the end of the cycle. This means that the resulting value of a particular output, corresponding to the current cycle of the network operation, will be determined by the last calculation of this output in the sequence of transition firing. As for the input signals, their values do not change during a single network operation cycle.

13. *Lack of aftereffect.* This property means that the taken control decision depends only on the states of CA and CO at the current moment of time and does not depend in any way on the decisions made earlier. The entire history of control process development is concentrated in the current marking of the network and there is no need to analyze the sequence of markings that precede it.

14. *Determinism*. Although the control process model itself, built in the form of a control E-network, can tolerate various behaviors depending on the parameters of the external environment, the actual implementation of the control system itself has to be strictly deterministic. This means that the control decision should not depend on non-determinism that is inherent to Petri nets and associated with the sequence of transition firing.

3.3. Control E-networks as the Functional Basis of PLA

3.3.1. Constructing Aggregate Mapping

Following the set task, we will use the aggregative approach to construct multilevel specifications for CA. For this purpose, however, it is necessary to solve the problem of constructing aggregate mapping that matches the formalism of the internal model in the form of CEN with the formal representation of aggregates.

An aggregate is represented as an object defined on a set of states Z, input signals X and output signals Y. The evolution of an aggregate is determined by the operators of transitions $H: X \to X$ and outputs $G: X \to Y$, which are generally random. The arrival of an input signal to the aggregate, which is considered to be an external event, can cause a change in the states of the aggregate, which also affect the internal events. The state of an aggregate for a certain moment of time $time > time_0$ is defined as specific implementation in accordance with this distribution law. The operator H determines the change in the states of the aggregate itself, and the operator G determines the rules for issuing output signals, which, in turn, can be transmitted to the output of other aggregates.

Among the variety of aggregates, the class of Piecewise Linear Aggregates (PLAs) stands out. Some specification of the general approach in this case provides the necessary formal basis for solving the problem. PLAs allow for modeling a wide class of objects and provide the ability to build multi-level aggregate systems, which can be considered input-output models.

The functioning of PLA is a Piecewise Linear Markov Process (PLMP) defined in *time* by the following expression:

$$x(time) = (q, x_q), \tag{3.13}$$

where $q \in Q$ – a certain discrete value called the ground state;

$x_q = (x_{q1}, x_{q2}, \dots, x_{q|q|})$ – a vector of complementary dimension $|q|$ coordinates with respect to the ground state;

$|q|$ – a non-negative value called the ground state rank.

PLA is characterized by a linear uniform change in the values of the vector x_q coordinates:

$$\frac{dx_q}{dtime} = -\alpha, \tag{3.14}$$

where α – a positive constant.

If we use control E-networks as formalism that specifies PLA behavior, it is necessary to ensure the construction of aggregate mapping in the form of PLMP.

Statement 3.1. The process of CEN functioning can be represented in the form of PLMP.

To prove the validity of this statement, we introduce the definition of the ground state.

We define the CA ground state as the marking $M_i^o \in$ of the control E-network fixed at the end of the execution cycle. The cycle is executed until there is at least one transition ready to firing.

The process of CA functioning, as specified by the control E-network, consists in performing a finite set of operations on the network variables and token attributes, which can only change as a result of network transition firing. If we assume that the change in the transition delay time occurs only at the time of the start of a new cycle, then each ground state of the control unit can be assigned a certain number of delayed transitions, which will not change in this cycle under any circumstances. On the other hand, the conditions for firing transitions, including the start of the delay phase, depend primarily on the state of place marking associated with the transition, and do not depend on how this state began. This satisfies the accepted principle of the lack of aftereffect in relation to CEN behavior. After entering the delay state, the transition can exit it only after the delay time, which is calculated in the active phase.

Let us assume M^o is the set of CEN markings corresponding to the ground states of the CA. Let us suppose $M_i^o \in M^0$ is the marking of the network at the end of the cycle iteration *i*, and $T_i \in T^*$ – a certain subset of CEN transitions delayed at M_i^o, which are connected by the mapping

$$\Gamma: M^o \to T^*, \tag{3.15}$$

where T^* is the degree set *T*. Then the sigma-algebra Ω of subsets selected from *T*,

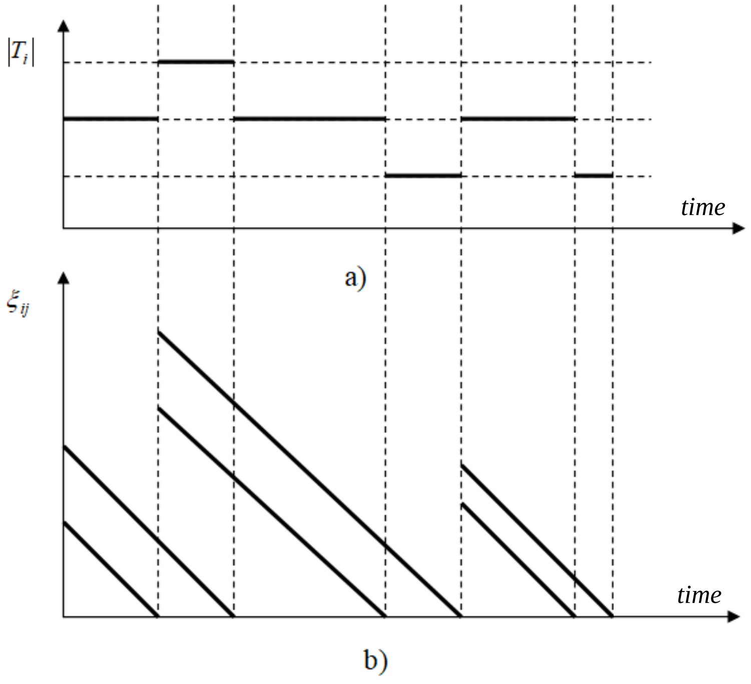


Fig. 3.3. Diagrams of the change in the number of delayed transitions (a) and the additional coordinate vector (b).

the elements of which determine the observed sets of delayed transitions, defines a measurable phase space of states $(T, \Omega), T \in \Omega$. From the definition of sigma-algebra, it is important to note that the operations of union, intersection and taking the complement, performed on the elements of the class Ω, are not derived from this class. It is also assumed that the class Ω contains each element $t_j \in T$.

Let us suppose ξ_{ij} is a variable equal to the time until the end of the transition $t_j \in T_i$ delay with marking M_i^o. Then the process of firing transition, i.e., the occurrence of events leading to a change in the CA state, can be represented as follows:

$$z_i(time) = (M_i^o, \xi_i(time)), \tag{3.16}$$

where M_i^o – network marking at the end of i operation cycle;

$\xi_i(time) = (\xi_{i1}(time), \xi_{i2}(time), \ldots, \xi_{i|T_i|}(time))$ – a vector of additional coordinates (transition delay times);

$|T_i|$ – a number of transitions delayed with marking M_i^o.

Let us suppose that we set the rate of change $\xi_{ij}(time)$ to a negative one, i.e.,

$$\frac{d\xi_{ij}(time)}{dtime} = -1, \tag{3.17}$$

then the process of CEN functioning will be represented by PLMP defined by Eq. (3.16).

The trajectory of CEN functioning process with respect to Eqs. (3.16) and (3.17) can be represented in the form of two diagrams shown in Fig. 3.3.

3.3.2. PLA Concretization by Means of Control E-networks

Since the CEN functioning under certain conditions can be narrowed down to PLMP, it is possible to describe the dynamics of PLA behavior. Furthermore, the E-network itself can be represented in the form of an aggregate system consisting of PLA. This example is demonstrated further.

Statement 3.2. CEN transition can be narrowed down to a complex system consisting of elementary aggregates that form a PLA.

To prove this statement, it is necessary to turn to the theorem (Buslenko, 1978), which reveals that in case the sets of ground states, input and output signals are finite, the PLA can be represented as a complex system consisting of three types of aggregates: memory elements, delay elements, and instantaneous piecewise linear converters.

CEN transition, the readiness function of which is equal to one, is defined as a tuple:

$$t = (P_t, \tau, z), \tag{3.18}$$

where P_t – a set of places incident to the transition;

τ – transition delay time;

z – a transition transformation procedure.

Let us describe each element in this transition definition in terms of PLA. We will consider a simple place as a memory element. Such an aggregate is capable of storing information in the form of some real number or a vector x with a countable number of elements. In this case, the vector elements will be the attributes of the token located in place. The input of the aggregate, represented by a memory element, can receive two types of signals: $(1, x)$ and an empty set, while their arrival can only alternate, since CEN is a secure network. If a signal $(1, x)$ is received at the input of the unit, i.e., a token with a vector of attributes $x = (x_1, x_2, \ldots, x_n)$ has been received, then, according to the locality principle, the state of the aggregate will be x until the arrival of the next input signal. Only the 0 signal can arrive at the input of the aggregate afterwards (during the process of moving the token). If, at this moment, the state of the aggregate is x, then a signal x is immediately sent to the output of the memory aggregate, and the internal state of the aggregate becomes zero, which means there is no token in place.

The delay element, also defined as an elementary aggregate, corresponds to the transition delay time. Such an aggregate, at any moment $time$, has a certain state $x(time)$ in the form of a non-negative number. Furthermore, $x(time)$ decrements until it reaches zero in the next cycle. At this moment, a signal y with a fixed value equal to 1 is sent to the output of the delay aggregate, after which the aggregate remains in this state until the arrival of the next input signal.

Three types of signals can be received at the input of the described aggregate:

- x_1, which sets the transition delay time equal to the result of calculating the delay function τ provided that the value of the transition activation function $\alpha = 1$;
- x_2, which changes the value $x(time)$ by the amount equal to the duration of the previous execution cycle;
- x_3, when for a non-delayed transition, provided that the value of the transition activation function $\alpha = 0$, the delay time $x(time)$ is set to infinity $x(time) = \infty$.

Consequently, for the listed input signals, the transition operator of this aggregate will implement the following changes in its state:

$$x(time+0) = H[0, x_1],\ x(time+0) = \tau,$$

$$x(time+0) = H[x(time), x_2],\ x(time+0) = x(time) - x_2;$$

$$x(time+0) = H[0, x_3],\ x(time+0) = \infty.$$

The output signals generated by the delay element are implemented by the output operator according to the following rule:

$$y(time+0) = G[0, x_1],\ y(time+0) = 0;$$

$$y(time+0) = G[x(time), x_2],\ y(time+0) = 1; x(time) = 0;$$

$$y(time+0) = G[0, x_3],\ y(time+0) = 0.$$

As a linear converter, we will consider the transformation function defined for the transition. The discussed elementary aggregate is able to receive and send signals in the form of vectors, respectively, for input signals $x = (x_1, x_2, \dots, x_m)$ and for output signals $y = (y_1, y_2, \dots, y_m)$. In addition, the lengths of these vectors can exceed the lengths of the input and output place signal vectors. In any case, the first n elements of this aggregate input signal vector always coincide with the output signal vector of the transition input place, from which the token is moved. The remaining $(m - n)$ elements of the vector y can be the values of the network variables (signals). When the input signal x arrives at the output of the converter, signal y is immediately sent, where y is the conversion result of $y = z(x)$ in accordance with the given conversion function z.

Thus, it has been proved that any CEN transition can be represented in the form of a PLA consisting of three aggregate types: memory elements, a delay element and an instantaneous converter.

The CEN transition, specified by tuple (3.18), is called an elementary PLA.

Since the union of two or more PLAs is a PLA itself, the CEN constructed from transitions represented in the form of elementary PLAs will also be a PLA, provided that it implements the PLMP. The transition operator of such an aggregate is defined as mapping:

$$H: M \to Z, \tag{3.19}$$

where M – a set of network markings, Z – a set of PLA states;

and the output operator G – mapping

$$G: M' \to Y, \tag{3.20}$$

where Y is the set of output PLA signals, $M' \subseteq M$.

The introduced concept of an elementary PLA makes it possible to define the rules of structural composition and decomposition of CEN of arbitrary complexity, consisting of separate transitions. These rules must comply with the general rules for constructing aggregate systems.

A signal at the output of the aggregate can be sent in case of transition firing, the output place of which is the boundary place of the aggregate. Similarly, the input signal will be received by the boundary input place of the aggregate. The aggregate system (AS) in this case is formed by connecting several PLAs in accordance with the conjugation scheme, which designates the mutual connections of the aggregates using pairs of sets: a set of boundary input places and a set of boundary output places of the aggregate.

The operator $\Delta \subseteq P_{in}xP_{out}$, which establishes links between the boundary places of CEN, is the conjugation operator.

The conjugation scheme is used to formalize the complex structure of a distributed control process defined on a set of aggregates. The internal behavior of each aggregate is determined using the implementation models specified in the form of the corresponding control E-networks.

For an unambiguous display of the dynamics of a distributed CA, we will make the following assumptions, which are determined by the rules for the AS composition, and consequently, individual transitions that include the description of the aggregates.

Assumption 1. Communication channels in a system consisting of aggregates are ideal, i.e., signals are transmitted instantly and without distortion.

Assumption 2. No more than one elementary channel connecting two adjacent boundary places can be connected to the input boundary place of any aggregate in the system. Any number of elementary channels can be connected to the output place, provided that no more than one of the mentioned elementary channels is directed to one input boundary place of the same system aggregate.

Assumption 3. If the moments of the external (associated with the arrival of the input signal – token) and internal (reaching the trajectories of the aggregate of a certain subset $M^{'} \subseteq M$) event occurence coincide, then the internal event has priority over the external ones, i.e., the change of state is executed first in accordance with the rule of internal event occurrence, and then, the actions initiated by external events are performed.

Aggregate systems, similarly to PLA, have an important closure property, which presuposes that the AS can be generally described as an aggregate; therefore, the union of a finite number of AS is also an AS. It creates the basis for the construction of complex hierarchical CCS models based on the aggregate approach. The ability to represent various types of CA structures in the form of AS (for the aggregate, it will be a system of elementary PLAs) allows performing structural transformations of models based on the assumptions made regarding the features of their construction and functioning.

3.3.3. Completeness of the CEN Formal Theory

When describing a method for specifying CA by means of PLA, concretized by the control E-networks, we have considered only the formal side of the used mathematical apparatus, leaving aside the conceptual meaning of the displayed process. At the same time, the practical implementation of this method will, in a certain way, be associated with the features of modeling control processes. From this point of view, control E-networks should be considered a formal system. Therefore, it is necessary to answer the question of adequacy, or completeness, of this formal system of the discussed subject area.

We will investigate this issue in terms of the theory developed by Hoare (Hoare, 1985) to study interacting sequential processes. Preliminarily, it is necessary to note that any formal theory is determined by:

1) the alphabet;

2) a decidable set of axioms;

3) a finite set of inference rules.

As noted before, we will interpret the process of CEN functioning in the form of a finite set of transitions fired, which lead to a change in place marking. Since the firing of transitions is associated with the movement of tokens, each of them can be associated with a certain subprocess that includes events in which this token can be a part of. Thus, the process of E-network functioning will generally consist of interacting sequential subprocesses, each of which is associated with the movement of a certain token. To terminate such a subprocess, it is enough to destroy the token that is associated with it and to spawn a new process – the token should be created.

Let us suppose that a set of transitions T is the specified alphabet of the process. By the process protocol we define a finite sequence of symbols from the alphabet T, which fixates the sequence of events (transitions). Let us assume that T^* defines the set of all finite protocols consisting of the set T elements. Let us define the degree-set of set T as the set of all its subsets $PT = \{X | X \subseteq T\}$.

According to Hoare (1985), the process is uniquely determined by three sets, which specify its alphabet, its divergence, and its failures, which, using the introduced definition, can be represented as follows:

$$Z = (T, K, L), \qquad (3.21)$$

where T – process alphabet;

K – relation between T^* and PT;

L – subset T^*, which meets the requirements for representing all divergences and failures of a process.

Divergences are understood as possibilities, including the possibilities of a non-deterministic choice of the process development direction after a specific event. Failure, in turn, is the refusal of the process to perform further actions.

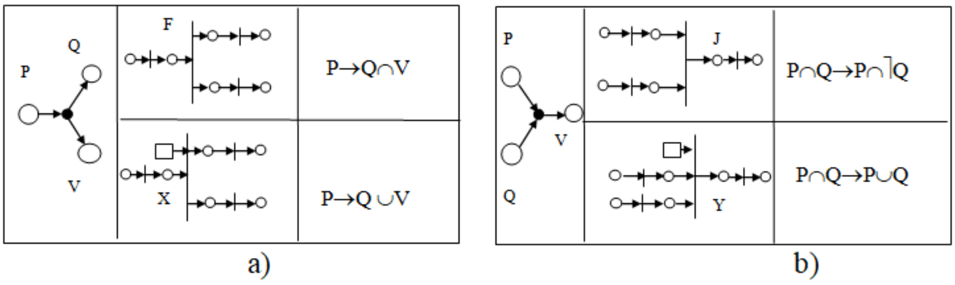


Fig. 2.4. CEN schemes of process development variants:
a) divergence; b) failure.

Generally, different processes have their own finite alphabets T, but they can only be composed of the number of transitions in the basic set $T_{CEN} = \{"T_T", "T_F", "T_j", "T_X", "T_Y", "T_{QF}", "T_{QL}"\}$. Thus, as given axioms of the formal CEN system, we can note and consider the elements of set T_{CEN} that includes the types of transitions in the basic set. Based on the set T, networks of varying complexity can be built using inference rules (transition composition rules inherent in the AS). In this case, each transition $t \in T$ is also an element of a set of protocols T^* with alphabet T, i.e., $t \in T$.

To prove the completeness of the formal CEN system with respect to the informal theory of sequential interacting processes, it is required to show that it is possible to represent any process divergences and failures defined in accordance with expression (3.21) with the help of transitions of the basic set T_{CEN}. Figure 3.4 shows CEN schemes that allow displaying any variants of process development.

The conducted research gives ground to conclude that the formal theory of CEN, in the given interpretation, has the property of completeness in the representation of interacting sequential-parallel processes characteristic of a CCS.

3.4. Summary

The analysis of the existing methods of control process formalization, performed while taking into account the requirements for CA implementation models, demonstrates that the aggregate approach has the biggest advantages in this regard, within which both the multilevel structure of CA and their dynamic properties can be taken into account. E-networks, which are an extension of Petri nets, are the most suitable means to reveal the internal structure of aggregates, allowing for the use of operational transformation of information circulating in the network, simultaneously with setting a complex logical sequence of its processing.

Following the accepted concept of aggregation, which forms the basis of the proposed formal definition of CA, two interrelated tasks have been formulated and solved:

- the theory of E-networks has been expanded in terms of their use for describing control systems interacting with the control object by means of input and output signal exchange;

- aggregate mapping, ensuring the embedding of E-networks into the scheme of PLA operation, has been constructed.

In the course of solving these tasks, a class of control E-networks has been proposed that is characterized by:

- an extended set of transitions, including transitions-queues, the use of which provides the network security;

- additional use of the activation function as part of the control mapping of transitions, which ensures the execution of the network consistent with the current state of the CO;

- the possibility to structurally change the network by dynamically changing the functions of the control mapping of transitions.

The semantics of control E-networks fully meets the principles of situational control over the IMS functioning processes, ensuring the development and implementation of control actions.

The given interpretation of CEN functioning process in the form of PLMP, taking on the role of aggregate mapping, makes it possible to consider E-networks as the functional basis of PLA. Based on the introduced definition of the transition, it has been proved that it is equivalent to the system of aggregates forming an elementary PLA. This circumstance has made it possible to determine the rules for the structural composition of control E-networks of arbitrary complexity, which must comply with the rules for constructing aggregate systems.

The study of CEN functional capabilities has revealed that, from a mathematical point of view, this apparatus can be considered a formal system that has completeness with respect to the informal theory of interacting sequential-parallel processes. This makes it possible to apply a unified formal approach to the description of CA at all levels IMS control.

Chapter 4. Predictive Models and Dynamic Model Checking

The method for specifying control systems using control E-networks allows not only specifying a formalized description of the control element behavior, but also subjecting this description to analysis in order to predict the future behavior of the control system. The basis for the construction of predictive models should be the corresponding implementation of the Receding Horizon Strategy, taking into account the peculiarities of CEN functioning in real time.

The problem of real-time evaluation of the dynamic properties of distributed control algorithms can be solved on the basis of the joint use of the mathematical apparatus of control E-networks and temporal logics. As a result, the initial description of CAs in the form of aggregate models with E-network concretization of aggregates is used in two qualities: as a model for the implementation of the control process and as a model for predicting its development.

4.1. Dynamic Properties of Control Algorithms and their Assessment

4.1.1. Control Algorithms as a Reactive System

A distinctive feature of the control algorithm specified by CEN, which is actually a formalized expression of the control program, is that its implementation depends not only on the state of the network, but also on the external environment, in particular, the state of the CO. For this reason, the control algorithm should be considered a reactive system, even a reactive control program.

In the general case, reactive systems are understood as real-time systems (Harel and Pnueli, 1985; Jansen and Gollmar, 2020) thar should develop a response to changing input information. Errors in the operation of such systems when they are used to control hazardous types of production (for example, this applies to control systems for nuclear reactors, chemical production, space technology, etc.) can lead to catastrophic consequences associated with large material losses and even human casualties. No less serious consequences can be in case of failures in the operation of telecommunication, information and financial systems. Therefore, extremely high requirements are imposed on the reliability and error-free functioning of reactive systems.

All informal requirements for the behavior of a reactive system are divided into two main classes (the classification was proposed by L. Lamport (1983)):

- the safety requirement – guarantees that a certain property is preserved in all states of the system;

- the requirement of liveness – guarantees that some event sometime in the future will necessarily occur in the system, or, in other words, some property will be true over some achievable state of the system.

Safety includes such properties as:

- a mutual exclusion, which guarantees that parallel processes will never simultaneously end up in the same critical section;

- an absence of deadlocks (absence of interlocking), which means that at least one of the processes can always continue functioning;

- a partial correction, meaning that if at the beginning of the process some precondition is true, then at the end of the process the postcondition will be executed.

Liveness is represented by the following properties:

- absence of infinite waiting (guaranteed response), which means that a request for resource allocation is always satisfied in a finite time;

- unconditional justice, which means that the process must be performed infinitely often, regardless of its internal state;

- total correctness, which means that if the process begins its functioning under a true precondition, then it will definitely end, and the postcondition will be fulfilled.

There is another classification of dynamic properties proposed by Alpern and Scheider (1985). It contains the following classes of requirements for the hierarchical behavior of the system (Borell classification):

- safety – a statement that a certain property is preserved in all states of the process of each computation from a set of possible computations;

- liveness – a statement that the property is checked in at least one state of each calculation;

- intermittence – the union of the safety and liveness classes on a set of calculations;

- recurrence – a statement that a certain property happens in infinitely many states of each computation from a set of computations (includes the safety, liveness, intermittence classes);

- persistence – a statement that a certain property happens in infinitely many states of each computation from a set of calculations, starting from a certain state (includes the safety, liveness, intermittence classes);

- progress – a union of the persistence and recurrence classes.

It is easy to see that the requirements in the Borell classification are covered by the safety and liveness requirements introduced by Lamport, and, therefore, the latter represents two basic informal requirements that can be put forward in relation to the dynamic properties of the control system. However, the specificity of CA allows one to single out two additional properties that are important from the point of view of control theory: the ability of the CA to resume its work from the initialization state (Frey and Litz, 2000) and the ability of the CA to perform its functions with any changes in the external environment. The last property is usually called robustness.

The properties of reactive systems described above, which also apply to CA, have one important feature. All of them require consideration of the process in the dynamics of its development in time. Such time dependence of the properties of control processes requires the use of special methods for their specification.

4.1.2. Methods for Specifying the Dynamic Properties of Reactive Systems

The common name for these methods comes from the term "temporal". Therefore, dynamic properties in this terminology are interpreted as temporal (Mordechai Ben-Ari, 2012).

The founder of the formal temporal approach is A. Pnueli, who was the first to propose the temporal logic of linear time (Linear Time Logic – LTL) (Pnueli, 1986). Linear Time Logic, also called Propositional Linear Temporal Logic (PLTL) (Manna and Pnueli, 1989), is the development of modal logic and is based on propositional calculus. Its practical appeal lies in the use of natural linear ordering on an infinite sequence of states and the classic modal operators (F – "eventually" and G – "global"), which can be applied to elementary statements connected by Boolean connectives.

A more precise definition of PLTL logic has two basic temporal operators: X – "next" and U – "until", with which the modal operators can be defined. The PLTL formula is determined by induction in such a way:

$$\varphi = p|false|\varphi_1 \to \varphi_2|X\varphi|\varphi_1 U \varphi_2. \quad (4.1)$$

Other temporal operators are expressed using these basic operators as follows:

$$F\varphi \equiv trueU\varphi, \quad (4.2)$$

$$G\varphi \equiv \neg F \neg \varphi. \quad (4.3)$$

The semantics of temporal logic is determined on the logical model given by the Kripke structure (Clarke, 2008):

$$M =< AP, S, s_0, W, I >, \quad (4.4)$$

where AP – a set of elementary statements;

S – a set of states (interpretations) of the system;

s_0 – an initial state;

$W \subseteq S \times S$ – a reachability relation between states;

$I: S \to 2^{AP}$ – an interpretation function that determines the values of propositional variables for each state.

However, PLTL is not very suitable for analyzing the behavior of concurrent processes, which are characterized by tree representation of many events. Therefore, for the evaluation of parallel processes, E. Clarke and E. Emerson proposed the Computation Tree Logic (CTL) (Clarke and Emerson, 1981). When the behavior of a reactive system is represented by computation trees, the liveness requirement is divided into two subclasses: when a state satisfying a given property is reachable on each branch of the computation tree of the system ("A-liveness") and when it is reachable at least on one branch of the computation tree ("E-liveness").

CTL formulas are inductively defined as follows:

$$\varphi = p|false|\varphi_1 \to \varphi_2|EX\varphi|E\varphi_1 U\varphi_2|A\varphi_1 U\varphi_2, \tag{4.5}$$

where $p \in AP$ – an elementary statement;

$p \in AP$ – CTL formulas;

E and A – quantifiers "existence" and "always", respectively.

Studies have shown that most of the practical properties of reactive systems can be expressed by CTL formulas (Ravn et al., 1993). Examples of formal definition of the properties of safety and liveness classes are given in Table 4.1.

Table 4.1 Determination of Some Properties of Reactive Systems by CTL Formulas

Property	**CTL formula**
Safety	
Invariance	AG(at_1→φ)
Mutual exclusion	AG(¬(critical1∧ critical2))
Partial correctness	(start ∧ φ) →AG(finish ∧ ψ)
Liveness	
Guaranteed response	(AG(request→AF(grant)))
Unconditional justice	AF(executed)
Total correctness	(start ∧ φ) →AF(finish ∧ ψ)

The semantics of temporal operators when applied to the analysis of computation trees is defined similarly to PLTL, but the consideration of quantifiers on a set of tree paths is added.

The disadvantage of PLTL and CTL logics is that time is taken into account in them implicitly and only at the semantic level, while the practical application of this formal apparatus, especially in real-time control systems, requires the specification of actions within specific time boundaries. In order to satisfy this requirement, T. Henzinger developed a version of the real-time linear logic – Timed Propositional Temporal Logic (TPTL) (Henzinger, 1991) and P. Alure proposed a version of the real-time computation tree logic – Timed Computation Tree Logic (TCTL) (Alur, 1991), which was the result of their joint work (Alur and Henzinger, 1990).

The TCTL formula is defined as follows:

$$\varphi = p|false|\varphi_1 \rightarrow \varphi_2|EX\varphi|E(\varphi_1 U_{\sim c}\varphi_2)|A(\varphi_1 U_{\sim c}\varphi_2), \qquad (4.6)$$

where $\sim$ means one of the binary relations $((<, >, \leq, \geq, =))$ that limits the duration of the formula.

Further developments of TCTL to the real-time domain were the Parametric Timed Computation Tree Logic (PTCTL) (Wang, 1996), Metric Temporal Logic (MTL) (Chang et al., 1994) and Parameterized Real-Time Computation Tree Logic (Emerson and Trefler, 1999).

Simultaneously with the development of temporal logics, another formal approach to the specification of the properties of reactive systems, which has a generalized name *Duration Calculus* – interval logics (Chaochen et al., 1993), took place. This mathematical apparatus was analyzed most fully in the ProCoS (Provably Correct Systems) project (Bowen, Hoare et al., 1996). Duration Calculus takes into account the peculiarity of the control program in real time and the possibility of parallel execution of several threads. This logic extends the traditional predicate logic with the ability to specify time intervals with an indication of the constraints imposed on them, as well as the definition of sequences of input and output signals.

The semantics of the Duration Calculus is determined by a specific interpretation of this calculus, which, in principle, should correspond to the specifics of the functioning of the system. Each state is interpreted through a function of time.

Using the Duration Calculus logic security requirements for real-time control systems can be easily described. At the same time, it is inferior to temporal logic from the point of view of practical implementation of automatic verification algorithms, since for interval logic the problem of satisfiability is algorithmically unsolvable (Paulson, 1998). In this regard, it becomes necessary to combine the merits of each considered formalism used to describe the dynamic properties of reactive systems, including CA, within the framework of a unified approach.

4.1.3. The Problem of Automatic Verification of CA Dynamic Properties

Verification is the process of certifying that a system meets specified requirements or has specified properties (Pritsker, 1995). As a kind of verification, validation is also considered, which implies checking the system for compliance with its specification. However, if verification answers the question "Is the system built correctly?", then the purpose of validation is to verify that the "correct system" is actually built. In the future, we will consider the first task, namely, the task of verification, which should be assigned to the control system itself and should be solved in the dynamics of the development of the control process. In this case, we will assume that system validation is the prerogative of the system design stage.

It is impossible to verify reactive systems by testing, since the number of possible variants of their behavior can be infinitely large. Therefore, in the theory of reactive systems, there are several alternative approaches to the problem of verification: deductive methodology, evidence-based design, model approach, and runtime monitoring.

Deductive methodology implies that the requirements for the behavior of an already existing system (program) are formulated in the form of a theorem to be proved by means of mathematical logic. In fact, it involves formal verification of the informally obtained description of the algorithm. The most prominent representatives of deductive methods are: automated theorem proving, multiset rewriting, thread spaces, and belief logic (Bibel and Wolfgang, 2007). At first glance, these methods are preferable, since they do not require additional steps to build a model. However, they are at the same time more difficult to implement and impose serious restrictions on the range of tasks to be solved.

Evidence-based design, like deductive methods, uses declarative languages; however, its goal is to synthesize correct algorithms using well-formed formulas, with the help of which the requirements for the behavior of the system are specified. In classical works on the theory of reactive systems, such algorithms are called decision procedures (Kroening and Strichman, 2008). Examples of solutions in this area are the proof of the feasibility and the feasibility of the specification, as well as improved technologies based on the resolution method. However, the complexity of the existing methods for the synthesis of algorithms is such that one can hardly hope for their application in solving practical problems.

Another area of CA verification is the model approach. In its current form, it means that a model is built according to the specification of the system, which is then tested for satisfying the specified properties in each of its states. Although it sounds paradoxical, the model approaches are especially powerful precisely because they knowingly limit themselves, working not with a system, but with some final model.

This allows checking the desired property not only in the most probable situations (as, for example, simple testing does), but, in general, in all possible states. That is why model technologies are used in critical applications (for example, in Havelund (2001), a formal analysis of a spacecraft flight control program is described).

The main method of the model approach in automatic verification of the dynamic properties of programs (algorithms) is Model Checking (MC) (Clarke et al., 1999). The formation of the MC took place simultaneously with the development of temporal logics. In fact, it boiled down to the development of algorithms for checking temporal formulas on a logical model specified in the form of a Kripke structure, as well as assessing the complexity of these algorithms. In particular, it was established that the MC problem for temporal logic CTL was solvable in linear time with respect to the number of model states and the complexity of formulas. However, for TCTL logic, the MC complexity becomes PSPACE complete (Alur and Henzinger, 1993).

There are several alternative approaches to solving the MC problem for various temporal logics. The main ones are the automaton-theoretic approach (Wardy and Wolper, 1994), using various modifications of the Büchi and Rabin tree automata, as well as symbolic computations (Khoussainov and Nerode, 2012), the μ-calculus (Grädel et al., 2007) based on Binary Decision Diagrams (BDDs) (Bryant, 1992). There are also examples of the application of temporal logics to verify the properties of Petri nets. For example, one of the extensions of the temporal logic TCTL proposed by W. Penchek (Penczek, 1990) is projected onto a model specified in the form of a temporary Merlin network.

Recently, verification tools based on MS have become widespread and have demonstrated the ability to detect rather subtle errors that occur in very unlikely situations. The most widespread tools are instrumental systems focused on the use of temporal logics, such as SMV (Carnegie Mellon University) (Gluch and Srinivasan, 1998), VIS (University of California, Berkeley) (Brayton et al., 1996), FormalCheck (Hardin, 1996), Spin (Holzmann, 1997), Java PathFinder (Visser et al., 2004), Bandera (Hatcliff and Dwyer, 2001). The principle of operation of these systems consists in the selection of an automaton model corresponding to it (for example, a Buchi automaton) according to the program code, on which the temporal formula is checked. However, full verification of specifications using MS is associated with overcoming the state space explosion problem, and, therefore, it cannot be effective for real-time systems generating NP-complete problems.

Run-Time Monitoring (RTM) is a class of methods that check system properties directly during program execution. Unlike MS, these methods are online, i.e., statements about the behavior of a process are embedded directly into the program code. The most prominent representative of RTM systems is Temporal Rover (Drusinsky, 2000), which allows for real-time checking of the properties of programs written in Java, C, C ++, VHDL, Verilog and ADA. This is achieved by converting the program code with the temporal properties written in the form of comments into the equivalent code that checks them. A limitation of Temporal Rover is that its state model cannot evaluate complex properties like "x () should never run after y ()". Another prominent example of the RTM class is Java Path Explorer (JPaX) (Havelund and Rosu, 2001). It is able to test only linear time properties and, like Temporal Rover, does not allow evaluating complex properties.

Having considered the existing approaches to the problem of verification of the dynamic properties of CA, we can draw the following conclusions:

1. Deductive methodology, like evidence-based design, tries to formalize the system as a whole and then uses all the advantages of a formal description, instantly obtaining qualitative assessments of the processes under study. However, these approaches are extremely expensive, limited in application, and are not yet compatible with the method of AC specification we have chosen in the previous section using E-network implementation models.

2. The model approach is well developed, both theoretically and practically, as it supports the use of the mathematical apparatus of temporal logics for the specification of program properties and has many applications, but only at the stage of system design.

3. RTM is more adequate to the real process, and it is reduced simply to checking the feasibility of specified requirements for the current case of program execution without building an exhaustive state model. However, in this case, it is not possible to predict deviations in the behavior of the system due to the absence of its model.

Taking into account the conclusions made, let us formulate the problem of dynamic verification of CA as the creation of a method for assessing the dynamic properties of reactive programs, which would make it possible to apply the MC mechanism within the framework of the RTM methodology based on predictive models that implement the Receding Horizon Strategy. To solve the problem, it is necessary:

- to clarify the temporal properties of the E-network implementation model from the point of view of its application for dynamic verification of control systems;

- to develop a dialect of temporal logic focused on control processes (analysis of complex properties taking into account interval constraints);

- using the developed logic, to construct a predictive model that takes into account the specifics of the interaction of the CA with the external environment;

- to develop an algorithm for assessing CA properties in real time.

The main difference of the proposed verification method should be the use of the CA implementation model, which is, at the same time, its specification. On the one hand, it saves time because no additional creation of automaton is required for the existing control program. On the other hand, it eliminates the inaccuracy of the model, which is always present, when it replaces the real system. In our case, we will have a complete coincidence of the model and the real process, the implementation of which is carried out according to its own model.

4.2. Temporal Model of Control E-networks

To develop and apply a formal methodology for the verification of control algorithms, they must be modeled by mathematical objects and the relationships between them. In our case, such a model is the description of the CA in the form of a control E-network. The E-network itself has already dynamics that characterize the development of the control process over time. However, to take into account the dynamic exertion of CEN, it is necessary to make a number of additional clarifications concerning three main points of its behavior:

• which time model is actually implemented using CEN;

• how the given time model is displayed on the network state;

• what is the calculation scheme provided by CEN.

4.2.1. Time Model of CEN

Time is the category around which all judgments about the dynamics of the behavior of the reactive system are formed.

In CEN time is entered as a function of transition delays. Although by definition it is considered a continuous value, in fact, in the network, time changes discretely. When performing CEN, the time is split into separate intervals, the duration of which does not have to be constant. In the future, each time interval will be called duration of the CEN execution cycle. This duration is measured by the CEN internal clock. We will assume that these clocks are started simultaneously with the start of the CA and are common for all network units, setting a single system time. Through execution cycles, the continuously changing real time, in which the control system operates, is coordinated with the discrete nature of the CEN operation, which is a model for implementing the CA.

Let us fix the moment of the beginning of the network execution cycle and assume that all subsequent actions related to its execution will relate precisely to the moment of the beginning of the execution cycle. It is clear that these actions take a certain amount of time. When the network finishes firing of all active transitions, and it comes to a static basic state, characterized by the fact that no transition can trigger any more in the situation that has arisen, the real time will change – it will no longer correspond to the fixed moment of the start of the execution cycle. The next cycle will start execution exactly from this moment in time, which in the future we will consider the time of a new execution cycle. For CEN, the time will change abruptly, running sequenti$l$lly through a set of values $\{time_1, time_2, \ldots, time_l, \ldots\}$, where $time_l$ is the time of the   -th execution cycle when $time_i < time_j$ for $i < j, time_i \in \Re^+$.

From cycle to cycle, the situation can change for two reasons. First, during the execution cycle, the values of the input signals may change, which affect the calculation of the transition activation functions. As a result, some transitions can become active, triggered, or go into a delayed state. Second, the delay time on transitions already in the delay will change, which will decrease by the amount of the duration of the previous execution cycle. If the delay time of some transition when changing the cycle drops to zero, then this transition can be triggered in the current execution cycle. The condition for triggering the delayed transition in the current execution cycle will be the true value of the next predicate

$$time_b + \tau_0(t) \leq time_c, \tag{4.7}$$

where $\tau_0(t)$ – the calculated value of the delay time at the transition when it is activated;

$time_b$ – the value of start time of the delay at the transition;

$time_c$ – the value of current time of the network execution cycle.

Let us assume that the delay time at the transition changes discretely and synchronously with the network execution cycles according to the rule:

$$\tau_l(t) = \begin{cases} 0, \text{ if } \tau_{l-1}(t) \leq (time_l - time_{l-1}) \\ \tau_{l-1}(t) - (time_l - time_{l-1}), \text{ otherwise.} \end{cases} \tag{4.8}$$

The change in the delay time at the transition from the moment of the beginning of its delay phase ($time = l + 0$) to triggering can be represented in the form of a diagram shown in Fig. 4.1.

Figure 4.1 demonstrates that, in fact, when the network is operating, a discrete approximation of a continuous linear change in the delay time at the transition is carried out.

If the condition $\tau_{l-1}(t) = (time_l - time_{l-1})$ was satisfied for the last cycle within the transition delay phase, then we would have an exact discrete model of continuous time. However, in reality, it may turn out that the time difference between the last two network execution cycles will be greater than the residual delay time at the

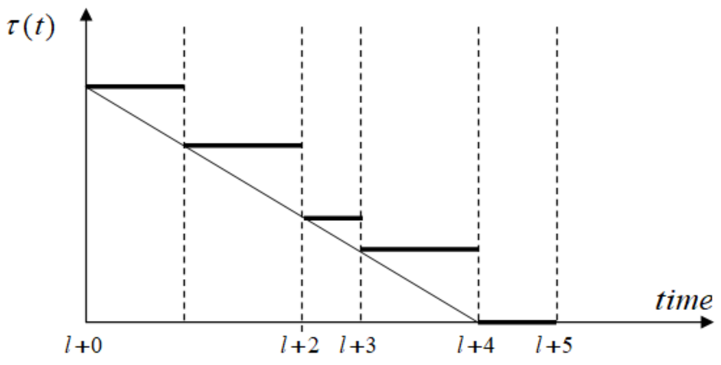


Fig. 4.1. Variation of transition delay time.

transition, i.e., the condition $\tau_{l-1}(t) < (time_l - time_{l-1})$ will be met. In this case, we will have a clearly expressed error of the discrete model, which is the value

$$\delta_M = (time_l - time_{l-1}) - \tau_{l-1}(t), \tag{4.9}$$

where $time_l$ is the start time of the *l*-th cycle of network execution.

The shorter the duration of the real-time execution cycles, the smaller the model error. As for the actual duration of execution cycles, it is determined solely by the time spent on checking the readiness and firing active transitions. The duration of the execution cycles can also be affected by the additional inclusion of any other calculations, for example, checking the properties of the CA using the prediction model.

The described time model differs from the discrete time model, which assumes that all time instants are selected from the domain of integers, as well as from the fictitious clock model, in which time is measured by the number of steps taken by the system from one state to another. This model is closest in its meaning to the dense-time model, which is characteristic of discrete-event systems.

How does this model of time relate to the functioning of CEN? This question is answered by the state model of CEN.

4.2.2. State Model of CEN

The state model of CEN must take time into account. To construct it, we use the well-known recurrent equation describing the dynamics of the Petri net (Murata, 1989):

$$M_k = M_{k-1} + BU_k, \tag{4.10}$$

where M_{k+1} – the state (marking), which is achieved at the $(k+1)$ -th control step;

M_k – the state (marking) at the k -th step;

U_k – control applied at the k -th step;

B – the matrix corresponding to the F set of arcs in the network.

Let us consider CEN, all transitions of which have zero latency. We will consider the execution of any network transition as a control step, as a result of which the marking of its places changes. If we restrict ourselves to one cycle of network operation, then it can include several sequentially executed steps as long as there are conditions for triggering transitions. This follows from the fact that a stepwise change in the marking of places can create conditions for the triggering of new transitions.

We will assume that the control vector $U_k = (U_{1k}, \dots, U_{jk}, \dots, U_{jm}), j = \overline{1, |T|}$, shows the possibility of triggering the transitions of the network at the k -th step: if $U_{jk} = 1$, then the transition t_j at the k -th marking is triggered, if $U_{jk} = 0$, then the triggering of the transition t_j does not occur. Each component of the vector U_k corresponding to a certain transition of the network is a logical function that depends not only on the transition scheme, but also on its readiness and activation functions, which can be represented as the following product:

$\forall t_j \in T[\sigma_{jk} \wedge \alpha_{jk}] \Rightarrow U_{jk} = 1,$ (4.11)

where σ_{jk} – a readiness function of transition t_j at the k -th step;

α_{jk} – an activation function of transition t_j at the k -th step.

Let us introduce the transformation matrix $B = \left\| b_{ij} \right\|, i = \overline{1,n}, j = \overline{1,m}$ with dimension $(n \times m)$, where n is the number of network places and m is the number of network transitions. The elements of this matrix determine the effect that the triggering of transitions has on the state of the CA, indicating in which places the tokens are added and from which they are removed. Let (-1) mean that the token is removed from the place, (1) – the token is added, (0) – the marking of the place remains unchanged. As a result, each column of the matrix B displays how the network marking changes when the corresponding transition is triggered.

As an example, let us consider the CEN as shown in Fig. 4.2.

This fragment of the network includes three transitions and seven places. We will assume that the decision functions of both transitions of the type "T_x" have a value equal to 1. We will also assume that the activation functions of all transitions are 1. Under these conditions, all three network transitions will be sequentially triggered,

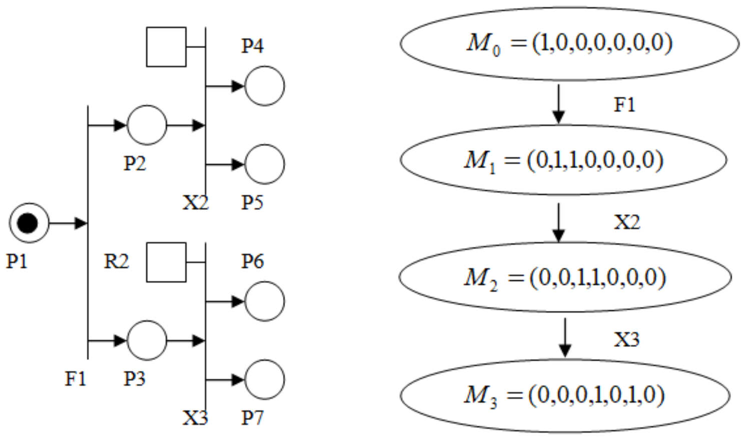


Fig. 4.2. Fragment of CEN and graph of its markings.

as a result of which the marking will change as shown in the graph reachability.

For a given CEN, we construct a transformation matrix and three control vectors corresponding to the given sequence of transition firing as shown in Fig. 4.3.

By Eq. (4.10), it is easy to check that for given B and U_j, the reachability graph is shown in Fig. 4.2.

Several transition firings can occur within a run loop until a token is reached where no transition is ready to fire under existing conditions. Moreover, the triggering of the same transitions can occur repeatedly under the influence of changing internal conditions of the network, which are recorded in the attributes of the tags. In addition, the values of decision procedures may change during the cycle. All this will lead to the modification of the matrix B from step to step. Therefore, the CEN equation of state for the run loop becomes:

$$M_l = M_{l-1} + \sum_{k=1}^{K_l} B_k^l U_k^l, \tag{4.12}$$

$$B = \begin{pmatrix} -1 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \qquad \begin{array}{l} U_1 = (1,0,0) \\ \\ U_2 = (0,1,0) \\ \\ U_3 = (0,0,1) \end{array}$$

Fig. 4.3. Transformation matrix and three control vectors.

where B_k^l – a transformation matrix at the k -th step of the l -th cycle;

U_k^l – a control vector at the k -th step of the l -th cycle;

K_l– a number of execution steps during the l -th cycle.

Let us introduce the operation of component multiplication of vectors •, which for two vectors a and b of the same dimension allows us to construct a new vector $c = a \bullet b$ whose components are calculated by the formula $c_i = a_i \cdot b_i$.

Then the vector U_k can be represented as follows:

$$U_k = (\sigma_k \bullet \alpha_k)^T, \tag{4.13}$$

where $\sigma_k = (\sigma_{1k}, \dots, \sigma_{mk})$ – a vector of transition readiness functions;

$\alpha_k = (\alpha_{1k}, \dots, \alpha_{mk})$ – a vector of transition activity functions.

Taking into account Eq. (4.13), Eq. (4.12) takes the form:

$$M_l = M_{l-1} + \sum_{k=1}^{K_l} B_k^l (\sigma_k^l \bullet \alpha_k^l)^T, \tag{4.14}$$

where σ_k^l – a vector of transition readiness functions at the k -th step of the l -th cycle;

α_k^l – a vector of transition activity functions at the k-th step of the l -th cycle.

Since during one cycle the values of the input signals do not change, the vector α_k^l will not change from step to step and can be written as α^l.

Then we finally get

$$M_l = M_{l-1} + \sum_{k=1}^{K_l} B_k^l (\sigma_k^l \bullet \alpha^l)^T. \tag{4.15}$$

Now let us consider the case when the delay time is set for transitions or at least some of them. We will assume that the determination of the moment when the transition is triggered is performed according to Eq. (4.7). Moreover, the delay time at the transition changes discretely and synchronously with the network execution cycles according to rule (4.8). It is also clear that the transitions that are in the delay cannot in any way affect the change in the network marking, despite the fact that they have passed the active phase. To provide for this circumstance, we introduce an additional function, which we will define as follows:

$$h(t) = \begin{cases} 1, \text{ if } \tau(t) > 0 \\ 0, \text{ otherwise.} \end{cases} \quad (4.16)$$

Then, taking into account the property of associativity of component multiplication of vectors, the CEN equation of state can finally be written in the following form:

$$M_l = M_{l-1} + \sum_{k=1}^{K_l} B_k^l (\sigma_k^l \bullet \alpha^l \bullet h^l)^T, \quad (4.17)$$

where $h^l = (h_1, \ldots, h_m)$ – a delayed transition vector.

Comparing (4.12) and (4.17), we obtain the final expression for the control vector CEN at the k -th step of the l -th cycle:

$$U_k^l = (\sigma_k^l \bullet \alpha^l \bullet h^l)^T. \quad (4.18)$$

From Eq. (4.18) it follows that the value of the control vector is determined by the marking of places, the state of the input signal CEN and the state of the transition. In addition, this definition fully corresponds to the previously accepted representation of the network behavior in the form of PLMP, when the execution cycles determine the main states of CEN and the vector of delayed transitions.

For temporary Petri nets, the equation of state could be a sufficient basis for revealing the dynamic properties of the process by analyzing the reachability graph built on the set of net markings. However, in the case of CEN, the transformation matrix is not stationary during network execution. Due to the multiplicity of variants of the "T_x" and "T_Y" types of transition firing schemes, the transformation matrix can vary from step to step, i.e., for CEN, no less important than changing the tokening is the very moment of the transition. In this regard, there is a need for a more detailed consideration of the process of network functioning from the point of view of the sequence of steps performed. This can be carried out on a computational model, which defines the operational semantics of CEN that ultimately determines the construction of a logical system for analyzing the dynamic properties of CA.

4.2.3. Computation Model of CEN

In contrast to the state model, which considers changing the markings of the network places, the transition firing sequences must be analyzed in the CEN computation model. Actions with data represented by the values of network variables and token attributes can only occur as a result of the transition transformation procedures. In this case, the conditions for performing network transitions may change. The resulting control action depends on how the calculations will be organized. Recall that in the general case, we consider distributed systems in which processes interact exclusively by passing messages. In our case, implementation models in the form of

CEN are used in the role of such descriptions, and the messages are tokens passed between the units. From a computational standpoint, this means that there can be no shared variables. Therefore, we will restrict ourselves to considering one process represented by an aggregate concretized by CEN. Other processes will implement similar computation models.

First of all, we note that according to the time model we have adopted, all operational actions in the system occur instantly. The model time does not change during the execution of the transition transformation procedures. Only after all the actions planned at the time of the execution cycle have been completed, the model time changes by linking it to the current real-time value. This means that the calculations are synchronized with the moments when the CEN execution cycles change, i.e., each cycle corresponds to its own portion of calculations. In the same execution cycle, multiple transitions, which are active and have timed out, may fire. From the point of view of model time, all these operations should be considered parallelly executable actions related to one computation.

Usually, when considering parallel processes, two main types of operational semantics are used: interleaving and partial ordering. In interleaving semantics (Emerson, 1990), the parallel execution of several actions is replaced by a non-deterministic choice of the order of their sequential execution. In partial order semantics (Peled and Pnualy, 1994), a partial order relation is established on the set of basic actions, reflecting the cause-and-effect relationships (CER) between events. The CER makes it possible to quite simply analyze many properties of computations associated with parallel execution of operations. In particular, it specifies the necessary conditions for ordering actions in alternating computations. For the specification of these properties, a new type of non-classical logics was even developed in due time – the logic of causality or causality logics (Alur et al., 1995).

The relationship between the semantics of interleaving and partial ordering for abstract parallel computation was studied by Bechet (1997), who formulated necessary and sufficient conditions under which the model of alternating computations was completely characterized by a single causal model, taking into account some restrictions imposed on causal relationships. This issue was studied separately for ordinary Petri nets. This result, however, cannot be directly extended to control E-networks, which, although they belong to the class of safe networks, have a number of functioning features that require additional rules. The difficulties that distinguish E-nets from ordinary Petri nets in terms of implementing parallelism are due to the use of type transitions and, when triggered, a check is performed for the absence of tokens in places (check for zero). In this case, an arbitrary (non-deterministic) order of execution of active transitions that are in the chain of actions with a conditional transition can lead to the wrong choice of the variant of its operation and, consequently, to the wrong direction of development of the control process.

Let us define a set of additional CER, which must correspond to the sequence of triggering of CEN transitions to ensure the deterministic execution of the parallel process implemented by the network. Let us denote $A = \{a_1, a_2, \dots\}$ a countable set of elementary actions performed on CEN transitions during one execution cycle. In fact, any elementary action $a \in A$ will correspond to the transformation procedure ϕ of some transition $t \in T$ of CEN. Let us introduce a marking function $\rho: T \to A$ that assigns some action $\rho(t) = a$ to each transition $t \in T$ in the network.

Let us introduce a number of definitions.

Definition 4.1. An oriented graph, each arc of which is tokened with one of the symbols of the set, is called a computational graph. The vertices of the graph are called states of computations.

The fact that a transition marked by an action a leads from a Γ graph state x_1 to a state x_2 will be denoted by $x_1 \xrightarrow{a} x_2$. We will assume that the activation functions are true for all CEN transitions, and the delay time is equal to zero. Under these conditions, we split the readiness function σ of a transition $t \in T$ marked with an action a into two components: a precondition φ_a, which is a predicate that is true for an admissible marking of the transition input places, and a postcondition ψ_a, which is a predicate that is true for an admissible marking of the transition output places. Recall that the admissible marking satisfies the transition firing scheme. In this case, the action a connecting the states x and y can be performed under the condition $\Gamma, x| = \varphi_a$ and $\Gamma, y| = \psi_a$. We assign predicates φ_a and ψ_a to the states of the computational graph. In fact, these predicates express the generalized state of transition places. The presence of a path between the states x and y will be denoted by $x \xrightarrow{*} y$, and for each path $\mu = x \xrightarrow{a_1} x_1 \xrightarrow{a_2} x_2 \xrightarrow{a_3} x_3 \cdots \xrightarrow{a_n} y$ connecting a pair of states x and y, and we will use $A(\mu)$ to denote the set of actions $\{a_1, a_2, \dots, a_n\}$ that token its arcs.

Definition 4.2. Any path in the graph Γ outgoing from the initial state x_0 is called computation, and the sequence of actions that tokens the transitions of this computation is called a trace.

Definition 4.3. A computational graph is called deterministic if no two different computations have identical traces.

We note right away that each computational graph can be transformed into a computation tree that generates the same set of traces.

We also denote as $C_\Gamma(x)$ the set of all computations of the graph Γ that end in a state x, and as $D_\Gamma(x)$ – the set of all computations starting in the state x.

Definition 4.4. A prefix $A(x) = \bigcup_{\mu \in C_\Gamma(x)} A(\mu)$ of the state x is a set of actions immediately preceding x.

Definition 4.5. The suffix $B(x) = \bigcup_{\mu \in D_{\Gamma}(x)} A(\mu)$ of the state x is the set of actions immediately following x.

The set of traces of all possible computations generated by graph Γ will be denoted as $\Pi(\Gamma)$. This set will define the functioning of CEN in interleaving semantics. For us, it will be important that the implemented sequence of actions should unambiguously determine the achieved state, which is dictated by the principle of CEN determinism.

Definition 4.6. Any binary relation $O \subseteq A \times A$ possessing the properties of antireflexivity and transitivity will be called the relation of the CER on the set of actions A of the marked CEN.

An action a_j is dependent on the action a_i if $(a_i, a_j) \in O$. Let us denote $O^{-1}(a) = \{a' : (a', a) \in O\}$ a set of actions on which a given action $a \in A$ depends. A sequence of actions, or trace, $\pi = (a_1, a_2, \dots)$ obeys O if all actions of the sequence are pairwise different and for any action $a_i \in \pi$ the condition is satisfied $O^{-1}(a_i) \subseteq \{a_1, a_2, \dots, a_{i-1}\}$, i.e., all actions of the set A, on which a_i depends, precede it in this sequence. Let us denote by using $\Pi(O)$ a sequence of actions that obeys the CER O. If the system of parallel processes is compared with the family of relations of the CER Θ, then the functioning of CEN in the semantics of partial order will be characterized by a set $\Pi(\Theta) = \bigcup_{O \in \Theta} \Pi(O)$.

The functional characteristics of the computational graph and the family of CER relations are interconnected. We will assume that the CER approximates the computational graph if $\Pi(\Gamma) \subseteq \Pi(\Theta)$. For the correct disclosure of the parallelism contained in CEN, it is necessary that the computational graph be consistent with the family of relations of the CER, i.e., $\Pi(\Gamma) = \Pi(\Theta)$ should be done. It is obvious that the union and intersection of any set of approximating CER Γ is also an approximating CER.

Let us denote $O(\Gamma) = \bigcup_{\Pi(\Gamma) \subseteq \Pi(O)} O$ the maximum ratio that approximates the computational graph Γ and formulate in the form of axioms the laws of operation of CEN, which set restrictions on the structure of the computational graph.

Axiom of uniqueness (A1). Each elementary action a occurs no more than once on any trace $\pi \in \Pi(\Gamma)$.

According to the axiom of uniqueness, repeated execution of the same transition transformation procedure during some computation corresponds to two different actions. This condition eliminates the possibility of loops occurring and can be provided by renaming actions if you need to execute a certain transition multiple times in one loop. It should be borne in mind that actions that may refer to the same transition must differ in the values of the parameters of the transformation procedure.

Axiom of commutativity (A2). If the actions a and b are not connected by a relation F^{+} that is a transitive closure of the incidence relation of the network $F^{'} \subseteq P \times T \cup T \times P$, then the result of the calculation does not depend on the order of their sequential execution.

This axiom means that no action blocks another if they are not in some ordering relation. This situation applies to all tagged CEN transitions.

Follow-up axiom (A3). For any triple of states x, y, z, such that both $x \xrightarrow{a} y$ and $x \xrightarrow{b} z$ are allowed, the actions a or b can be performed only after all other actions belonging to $B(y)$ and $B(z)$ have been performed. The succession axiom defines the condition for the execution of transitions of type "X", which consists in the fact that before their execution all traces of actions that belong to the suffixes of the transition post-conditions must be checked.

Precedence axiom (A4). For any triple of states x, y, z, such that both $x \xrightarrow{a} z$ and $y \xrightarrow{b} z$ are allowed, actions can be performed only after all other actions belonging to $A(x)$ and $A(y)$ have been performed.

This axiom defines a condition for the execution of transitions of type "Y", which consists in the fact that before their execution, all traces of actions that belong to the prefixes of the transition preconditions must be checked.

Priority axiom (A5). For any states z and $z^{'}$ that satisfy axiom A3 or A4, there is an order relation $\prec$ that establishes the sequence of their execution.

If axioms A1 and A2 define the general properties of the computational graph, then axioms A3–A5 are the CER. Moreover, each of these relations represents one of the variants of causality on the set of CEN actions.

*Theorem 4.1.*_If the graph Γ of computations satisfies axioms A1–A5, then it is deterministic.

Evidence. Let us suppose that some state x has been reached as a result of computation. It means that there is some trace $\pi = (a_1, a_2, \ldots, a_n)$ that leads from the initial state x_0 to x. Let us show that any other computation leading to this state has an equivalent trace. Let this other computation have a trace $\pi^{'} = (a_2, a_1, \ldots, a_n)$ that differs in the sequence of the first action. If this trace does not contain actions on transitions of types "X" or "Y", then, following axiom A2, actions a_1 and a_2 can be arbitrarily rearranged in a sequence of actions. As a result, we get $\pi = \pi^{'}$. By induction, similar transformations can be performed for other actions. If the traces π and $\pi^{'}$ contain transitions of types "X" or "Y", then according to axioms A3 and A4, all actions corresponding to these transitions will be located at the end of the sequences of traces. We will divide the trace $\pi^{'}$ into two non-intersecting traces $\pi_1^{'}$ and $\pi_2^{'}$ so that $\pi_1^{'}$ will contain all transitions except types "X" and "Y", and $\pi_2^{'}$ – all transition types "X" and "Y". For $\pi_1^{'}$ the above-described permutation of actions can be applied. As for $\pi_2^{'}$, it will be completely equivalent to the trace $\pi_1^{'}$, since the mutual arrangement of actions inherent in the transitions types "X" and "Y" is uniquely determined by axiom A5.

Corollary 4.1. The sequence of CEN actions, built for the execution cycle in accordance with axioms A1–A5, uniquely determines the achieved state of computation, the characteristic of which coincides with the set of actions of this sequence. The introduced axioms A1–A5 form the basis for constructing the dynamic synchronization algorithm of CEN, which is described below. This algorithm ensures the deterministic operation of the network according to the PLMP scheme.

4.2.4. Algorithm of CEN Dynamic Synchronization

We assume that T is a finite set of CEN transitions. Let us conditionally divide this set into two subsets:

1) $T_R \in T$, $T_R = \{T_X, T_Y\}$ – a set of transitions with decisive places, which we will call a set of conditional transitions;

2) $T_S \in T$ – a set, which we will call a set of simple transitions, $T_S = \{T_T, T_F, T_J, T_{QF}, T_{QL}\}$.

Since the PLMP corresponding to the operation of CEN is determined by the sequence of transitions for which the delay time has expired, a subset of active transitions can be selected from the sets T_R and T_S. All transitions that are not in a delayed state at the moment $time$ the marking is changed are classified as active transitions. In contrast, transitions that are delayed, i.e., those for which $\xi_{ij} > 0$, we will call passive. At the moment $time$ they do not affect the development of the process in any way and are not taken into account. Thus, for each moment in time at which the network marking is changed, subsets of active conditional and simple transitions can be distinguished, which we will denote $T_R^A \subseteq T_R$ and $T_S^A \subseteq T_S$, respectively. In the course of the model operation, the composition of the sets T_R^A and T_S^A will change. In this algorithm, we will use two auxiliary lists: the list of simple transitions, where active simple transitions from the set T_S^A are entered, and the list of conditional transitions, where active conditional transitions from the set T_R^A are entered. Items are listed in the order in which they are received, i.e., the first items on the list are the items that arrived first. When you retrieve the first item from the list, the next item becomes the first item in the list. When the lists are cleaned, all the elements in them are deleted. We will assume that the algorithm starts to work at the moment $time + 0$ after the movements of the tokens in the transitions, the delay time of which expired at the moment $time$, are performed.

The synchronization algorithm in each cycle of CEN execution includes the following steps.

1. Start. Selecting a set of active transitions. From the set of network transitions T, determine the subset of active transitions $T^A = T_S^A \cup T_R^A$ that are not in delay at a given moment $time + 0$.

2. Formation of the list of conditional jumps. From the set T^A, select a subset of conditional transitions T_R^A, the elements of which are added to the list of conditional transitions in the order they are listed.

3. Formation of a list of simple transitions. From the set T^A, select a subset of active simple transitions T_S^A, the elements of which are added to the list of simple transitions.

4. Checking the condition "The list of simple transitions is empty". If this condition is met, then go to Step 10.

5. Checking the readiness for firing of the first transition in the list of simple transitions. Analyze the marking of the transition places and the value of the activation function. If they do not meet the conditions for firing the transition, then go to Step 8.

6. Determination of the delay time of the first transition in the list of simple transitions. The transition delay time is calculated. If the delay time is zero, then go to Step 9.

7. Correction of the composition of many active transitions. Transfer the first transition in the list of simple transitions to a delay and exclude it from the set of active transitions $T^A \backslash t$.

8. Correction of the list of simple transitions. Exclude the first transition in the list of simple transition from the list $T_S^A \backslash t$. Go to Step 4.

9. Firing the transition first in the list of simple transitions. Change the marking of the transition places in accordance with the transition firing rules. Perform the transition transformation procedure. Go to Step 3.

10. Checking the condition "The list of conditional jumps is empty". If this condition is met, then go to Step 14.

11. Checking the readiness to fire the first transition in the list of conditional transitions. Analyze the tokening of the entry and exit places of the transition. Determine the state of the decision place by calculating the value of the corresponding decision procedure. If the transition firing condition is not met, then go to Step 13.

12. Initiating the execution of the first branch in the conditional branch list. If the

calculated transition delay time is equal to zero, then change the marking of its places in accordance with the rules for firing transitions of this type. Perform the transition transformation procedure. If the transition delay time is not zero, then convert the transition to a delay.

13. Correction of the list of conditional branches. Exclude the first transition in the list of conditional transition from the list $T_R^A \backslash t$. Go to Step 3.

14. Stop the algorithm. With the network marking created at the moment, further triggering of its transitions is impossible.

Taking into account the above algorithm, we will make two remarks about the structure of CEN. First, among the transitions with zero delay included in the chain of structural links with conditional transitions, there may be transitions that also belong to one of the types "X" or "Y". In this case, for the synchronization algorithm to work correctly, the sequence of triggering conditional transitions must be additionally specified. This can be done without introducing complicated specifications of jump priorities, by providing at the step of compiling a list of conditional jumps to include them in the list in accordance with the numbering that must be set when describing the network. If conditional transitions are separated by transitions with a non-zero delay so that their inclusion in one group event is excluded, this requirement may not be taken into account.

Second, the operation of the algorithm implies compliance with the network security condition (there can be no more than one token in the place). This was made possible by the introduction of simple place queue transitions, which replaced the queue places previously used in practice.

4.3. Verification of Control Algorithm Properties Using Predictive Models

Models of time, states and computations constitute the basis on which the theory of formal verification of the dynamic properties of CA is built. Based on these models, it is possible to justify the used logical system. This clause describes a new control-oriented CTL logic extension that is used in conjunction with the developed predictive models that implement the Receding Horizon Strategy.

4.3.1. DCTL Logic: Syntax and Semantics

While LTL and CTL contain expressions about the relationship of pure Boolean assertions, in CA it is required to consider logical assertions taking into account the time sequence and duration of execution, as well as the simultaneous control of parameter threshold values. For example, a statement of the form "Within one minute from the moment of event A, the value of the variable X should not change by more than 5 %" already needs additional definitions, even in the case of TCTL logic. Such a requirement combines the time dependence between the sequence of events in the system with an assertion based on the temporal properties of the variable X. Such a need to take into account the duration of retention of some assertion is typical of the robustness property when it is required to evaluate the characteristics of the system in the dynamics of their change.

Another example might be related to the following statement: “Within any 30 second observation cycle, the duration of event B should not exceed 4 seconds”. This statement combines the requirements for two interconnected intervals, which also cannot be described using the known temporal real-time logics. Our goal will be to build an extension of the TCTL temporal logic by giving it the ability to set interval constraints for formulas, similar to how it is done, for example, in Duration Calculus.

Syntax of Duration Computation Tree Logic (DCTL)

The DCTL alphabet consists of:

- an infinite countable set of propositional variables $AP = \{p, q, r, \dots\}$ that denote predicates and two logical constants *true* and *false*;
- signs of logical connectives of the propositional calculus $\wedge$, $\vee$, $\neg$, $\rightarrow$;
- signs of binary relations $<, >, \leq, \geq, =$;
- quantifiers of generality A (“everywhere”) and existence E (“eventually”);
- temporal operators: F (“sometimes”), G (“always”), X (“in the next cycle”), U (“until”);
- separators <,> and <;>;
- brackets (), [], (], [);
- interval expressions of the form $\sim c$, constructed using constants $c \in \Re^{+}$, brackets and signs of binary relations $<, >, \leq, \geq$.

An elementary DCTL formula is any propositional variable or the constants *true* and *false*. Other DCTL formulas are defined by induction:

$$\varphi = p|\neg p|p_{\sim c}|\varphi_1 \rightarrow \varphi_2|E(\varphi_1 U_{\sim c}\varphi_2)|A(\varphi_1 U_{\sim c}\varphi_2). \tag{4.19}$$

Temporal operators G (“always”) and F (“sometimes”) are defined in a traditional way for temporal logics by Eqs. (4.2) and (4.3).

In DCTL, interval expressions that limit the time domain under consideration will be applied not only to temporal operators F, G, U but also to propositional variables.

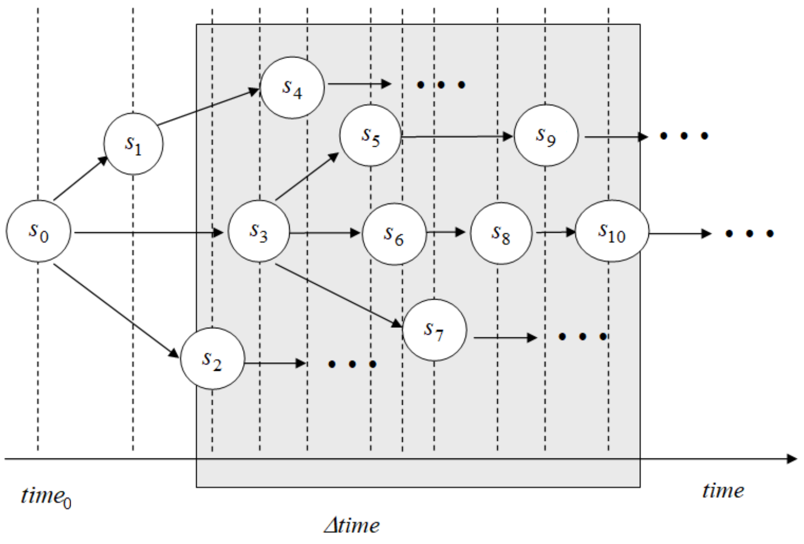


Fig. 4.3. Situation tree structure.

In the case of setting a restriction on the scope of the temporal operator, the absolute system time is always assumed, which is counted from the moment the control program starts.

For example, the expression $G_{<50}p$ means that there are always up to 50 units of time, i.e., at $time < 50$, the property p must be fulfilled. An interval expression can also be defined as a closed or open interval. For example, $F_{[5,10)}q$ means that sometimes in the interval $5 \leq time < 10$0a a condition q must be met.

Unlike temporal operators, the application of interval expressions to propositional variables always implies a relative timing and denotes the duration of the interval, during which a given variable continuously retains its true value. For example, $p_{>5}$ means that the time interval is considered when the propositional variable p remains *true* for more than 5 time units. Note that temporal operators and variables without interval expressions are treated as constrained operators.

Semantics of DCTL

Let us introduce a logical model corresponding to the temporal structure of CEN.

Statement 4.1. Evaluation of the CEN properties should be carried out in situations corresponding to the basic states of the network.

The confirmation of this statement is based on three fundamental principles that define three options for accounting for the values of CEN variables and tag attributes. The first of them is related to taking into account changes in the values of the input signals. Due to the principle of *periodicity* or *frequency*, the values of the input signals are fixed at the start of the cycle and do not change during the entire execution cycle.

The second option is related to the values of the output signals. Due to the principle of *direct action*, all data operations in CEN can be performed only on network transitions when calling transformation functions. It means that a change in the output signals in the future cannot occur otherwise than as a result of a change in marking, the final formation of which ends at the end of the execution cycle.

The third option is due to the dynamics of changes in the values of the attributes of tokens. By virtue of the principle of *determinism*, for any particular set of values of the input signals, a computation trace will always be executed, leading to a single final marking corresponding to the basic state of the network.

Let us consider a separate loop of network execution. We will call s -state a pair (s, ξ), where $s \in S^o, S^o \subseteq S$ is the situation corresponding to the basic state of the network, and ξ is the vector of additional coordinates that fixes the delay time on network transitions in the situation s. After the expiration of the delay time at any of the transitions marked in ξ, regardless of the values of the input signals, a situation change will occur, which will lead to a new s -state $(s^{'}, \xi^{'}), s^{'} \in S^o$.

The current situation can be inherited not by a particular situation, but by a whole variety of situations. This may be due to non-determinism of the control object behavior. In addition, the development of the situation may be affected by the receipt of tokens from related aggregates. Therefore, in the general case, for each specific situation s, a whole set of subsequent situations can be determined, which we denote $succ(s) = \{s_i^{'} | s \in S^o, i = 1,2, \ldots, k\}$. Each situation $s^{'} \in succ(s)$ can also have similar evolution. As a result, a tree of situations will be formed. This fact can be illustrated using the situation tree structure shown in Fig. 4.4.

For a formal definition of a situation tree, we introduce two definitions.

Definition 4.7. Let us call s -way an infinite sequence of situations and times $\delta \in \Re^+$ of the form $s_1 \xrightarrow{\delta_1} s_2 \cdots s_n \xrightarrow{\delta_n} \cdots$ such that $s_j \in succ(s_i), i = j + 1$.

Further we will consider the s -path as a mapping $\pi: \Re^+ \rightarrow S$ that satisfies the condition $\pi(0) = s$. Then $\pi(time)$ is the path from s in the moment $time$.

Definition 4.8. We will call a prefix of π, which we denote $\pi_\tau^{'}$, a subset of π defining mapping from a domain $[time - \tau, time)$ to .

Definition 4.9. Let us introduce the operation of concatenation of path segments, which we define as follows:

$$\begin{cases} \pi_\tau^{'}(time^{'}), \text{ if } time - \tau < time^{'} < time \\ \forall time^{'} \in \Re^+ : \pi_\tau^{'} \bullet \pi(time^{'}) = . \\ \pi(time^{'} - time), \text{ if } time^{'} > time \end{cases} \tag{4.20}$$

Definition 4.10. Let us call s -tree in CEN the set of s -paths starting at s. If $tr \subseteq S^o \times 2^{\Re^+ \times S^o}$ then $tr(s) = \{\pi | \pi(0) = s\}$ is the s -tree starting at s.

Since $tr(s)$ is a set of s -paths, then $\pi_\tau^{'} \bullet tr(s) = \{\pi_\tau^{'} \bullet \pi | \pi \in tr(s)\}$.

We will also assume that the relation tr has the property of closure:

$$\forall s \in S^{'}. \forall \pi \in tr(s). \forall time \in \Re^+ [\pi_\tau^{'} \bullet tr(\pi(time) \subseteq tr(s))]. \tag{4.21}$$

From (4.21) it follows that the behavior of the network does not depend on the past and depends only on the current situation. This fully complies with the CEN principle of no aftereffect. The property of closure asserts that the relation of total attainability is established on the tree of situations.

Statement 4.2. CEN generates Kripke's temporal structure

$$M =< AP, S^o, tr, M_0, I >, \tag{4.22}$$

where AP – a set of propositional variables that are elementary statements on a set of network variables V and a set of values of attributes of tokens $\{dm_i\}, i = \overline{1, |P|}$ located in places;

S^o – a set of situations corresponding to the main states of the network $S^o \subseteq S$;

$tr \subseteq S^o \times 2^{\Re^+ \times S^o}$ – a tree of situations with the property of closure;

s_0 – an initial situation, $s_0 \in S^o$;

$I: S^o \to 2^{AP}$ – an interpretation function that sets the values of propositional variables for each situation corresponding to the CEN basic state.

We will confirm this statement by establishing correspondence between the components of CEN and the elements of the Kripke structure given by Eq. (4.22). We will assume that predicates constructed from network variables, attributes of tokens, functional symbols, signs of arithmetic operations and binary relations $<, >, \leq, \geq, =$, as well as brackets and constants correspond to propositional variables. For example, a variable p can mean a predicate $defl < 0.5$, where $defl$ is a network variable indicating the amount of deviation. A set of interpretations S^o is determined on a set of situations corresponding to the main states of the network. This means that the evaluation of the values of the propositional variables will be performed at the end of each execution cycle, and we are only interested in the final markings of each execution cycle. Intermediate markings that can be achieved within the run cycle will not be counted. The relation tr means that every situation $s \in S^o$ has at least one follower $s^{'}$. The interpretation function $I(s)$ is determined for each situation by calculating predicates corresponding to propositional variables.

Based on these calculations, the values of subformulas and formulas in general are determined as it is shown below.

For a structure M, a state $s \in S^o$ and a formula φ, the satisfiability relation $M, s| = \varphi$ is defined as follows:

$s| = p \Leftrightarrow p = true.$

“In the state s, the predicate p is true”;

$s| = \neg(\neg p)$

“The negation of a false predicate is true”;

$s| = \varphi_1 \rightarrow \varphi_2 \Leftrightarrow s| \neq \varphi_1 \vee s| = \varphi_2$

“If the formula φ_1 is not satisfied or the formula φ_2 is fulfilled, it means that the formula φ_2 follows the formula φ_1”;

$s| = p_{\sim c} \Leftrightarrow (s| = p) \wedge \forall(s^{'} \in \pi^{'}_{\sim c}): s^{'}| = p)$

“The predicate is true during the time $\sim c$ on the state s, if there is a prefix s with the duration $\sim c$ on which the predicate p is true”;

$s| = EX_{\sim c}\varphi \Leftrightarrow \exists(\pi \in tr(s)).\, s^{'} \in \pi.\, s^{'} \in succ(s): s^{'}| = \varphi$

“The formula is true if on the time interval in the s-tree there is a path on which the formula is executed in the following situation”;

$s| = AX_{\sim c}\varphi \Leftrightarrow \forall(\pi \in tr(s)).\, s^{'} \in \pi.\, s^{'} \in succ(s): s^{'}| = \varphi$

“The formula is true if on the time interval $\sim c$ in the s-tree on all paths the formula φ is executed in the following s situation”;

$s \models E(\phi_1 U_{\sim c} \phi_2) \Leftrightarrow \exists(\pi \in tr(s).((\exists(time \sim c): \pi(time) \models \phi_2) \wedge (\forall(0 < time^{'} < time): \pi(time^{'}) \models \phi_1))$

“The formula is true if in the time interval $\sim c$ in the s-tree there is a path along which the formula φ_1 is executed until the formula φ_2 becomes true”;

$$s \models A(\phi_1 U_{\sim c} \phi_2) \Leftrightarrow \forall(\pi \in tr(s)).((\exists(time \sim c) : \pi(time) \models \phi_2) \wedge (\forall(0 < time' < time) : \pi(time') \models \phi_1))$$

"The formula $\sim c$ is true if the formula φ_1 is executed on all paths in the s-tree on all paths until the formula φ_2 becomes true";

$$s| = EF_{\sim c}\varphi \Leftrightarrow \exists(\pi \in tr(s)).(\exists(time{\sim}c) : \pi(time)| = \varphi)$$

"The formula is true if there is a path in the s-tree in the time interval $\sim c$ on which the formula φ is ever executed";

$$s| = EG_{\sim c}\varphi \Leftrightarrow \exists(\pi \in tr(s)).(\forall(time{\sim}c) : \pi(time)| = \varphi)$$

"The formula is true if there is a path in the s-tree in the time interval $\sim c$ on which the formula φ is always executed";

$$s| = AF_{\sim c}\varphi \Leftrightarrow \forall(\pi \in tr(s)).(\exists(time{\sim}c) : \pi(time)| = \varphi)$$

"The formula φ is true if the formula is ever fulfilled on the time interval $\sim c$ in the

Table 3.2 Determination of CA Properties by DCTL Formulas

Property description	*DCTL formula*
Safety	
In any situation, always after 12 minutes from the start of CA, with the pump running, the pressure in the chamber should not exceed 0.005 atm (invariance)	$AG_{>12}(pump \to (pres \leq 0.005))$
During the daytime in all situations, job A will be performed until job B starts (mutual exclusion)	$A(jobAU_{(9,17)}jobB)$
Always at least one engine must be running for a period of time up to 40 seconds after the start of movement (no dead ends)	$AG(go_{<40} \to ((drive1 \vee drive2)))$
Liveness	
Always from 8 am to 8 pm in the event of a conveyor idle for more than 10 minutes, the foreman must arrive at the site (guaranteed response)	$AG_{[8,20]}(stop_{>0.2} \to AF(master))$
A situation should arise when, in response to a request lasting from 5 to 10 seconds, the answer will be received at the next connection (total correctness)	$EF(request_{(5,10]} \to X(response))$

s-tree on all paths";

$s| = AG_{\sim c}\varphi \Leftrightarrow \forall(\pi \in tr(s)).(\forall(time \sim c): \pi(time)| = \varphi)$

"The formula φ is true if the formula is always fulfilled on the time interval $\sim c$ in the s-tree on all paths".

Let us note a number of equivalences inherent in DCTL formulas:

- $EF_{\sim c}\varphi \equiv E(trueU_{\sim c}\varphi)$;
- $AF_{\sim c}\varphi \equiv A(trueU_{\sim c}\varphi)$;
- $EG_{\sim c}\varphi \equiv \neg AF_{\sim c}\neg\varphi)$;
- $AG_{\sim c}\varphi \equiv \neg EF_{\sim c}\neg\varphi)$.

In addition to using interval constraints for predicates in DCTL, compared to TCTL, the operator X ("next") is returned, which is determined after the current situation. It is convenient for specifying properties that take into account the dynamics of changes in a certain parameter at each execution cycle.

Examples of the description of CA properties using DCTL formulas are given in Table 4.2.

4.3.2. Predictive Model

The interpretation of the CEN behavior in the form of PLMP allows determining the Kripke structure to substantiate the semantics of the temporal logic used, as well as provides the construction of a predictive model for analyzing the dynamic properties of the control process in the future. The forecast should consist in establishing a correspondence between the state of the CA and the state of the CO. This correspondence will be determined using the DCTL formula. We will assume that the state of the CA corresponds to the state of the CO if the value of the predicate written in the form of the DCTL formula is true. In this case, the situation arising in the CCS will be considered acceptable. Otherwise, an exception must be recorded that initiates changes to the CO.

Definition 4.10. The predictive model (PM) is called the six

$$PM =< CEN, \Phi, f, g, h, s_0 >, \tag{4.23}$$

where CEN – a given formal definition of CA in the form of a control E-network;

Φ – a set of DCTL logic formulas that specify the dynamic properties of the CA;

$f\colon V_O \to V_I$ – mapping of the output signals of the network to its inputs;

$g\colon DI \to \Re^+$ – mapping of discrete inputs of the network at many points in time;

$h\colon AI \to F$ – mapping that sets the laws of their change in time for analog input signals;

s_0 – the initial situation from which the prediction is based.

Let us consider each of the PM components in more detail.

CEN, which is the CA model, specifies all possible options for its implementation. Thus, CEN provides a simulation model of the control process with which computational experiments can be carried out. No additional steps are required to simulate CA. It is enough only to trace the movement of CEN in the space of ground states determined by the PLMP.

The initial situation s_0 for the prediction is determined by the marking of the network places and the set values of the input signals. Further, the process can develop from one basic state of the network to another. Moments of changing the main states will be determined by the specified transition delay functions. Following the PLMP scheme, the future development of the control process, up to the discrete model error, can be represented as a set of pairs $\{z_i(time) = (T_i, \xi_i(time))\}, i = 1,2,\ldots$ where T_i is a set of transitions delayed in the *i*-th basic state of the network, and $\xi_i(time) = (\xi_{i1}(time), \xi_{i2}(time), \ldots, \xi_{i|T_i|}(time))$ is the vector of additional coordinates corresponding to this ground state. Then, the predicted time of the main state change will be calculated by the formula

$$time_{i+1} = time_i + \min_j(\xi_{ij}), \tag{4.24}$$

where $time_i$ is the time to reach the *i*-th basic state of the network.

Taking into account the time delays at network transitions by ranking them within the framework of PLMP is only part of the simulated network dynamics determined by its internal conditions. Another part of this process is associated with a change in operating conditions external to the CEN – a change in the input signals of the network. To simulate external conditions, we will use the CO model, which we will build on the basis of the given relations f, g, h.

This model should provide for the fixation of two variants of events related to the operation of the CO. The first option concerns the generation of discrete input signals, the second – the calculation of analog (continuous) values. Let us first consider a variant of discrete signals. We will assume that in a general case, each CEN output $v' \in V_O$ can be assigned not one input signal, but a whole set of discrete input signals $\{v'' | v'' \in DI\}$, such that $f(v') \subseteq DI, DI \in V_I$. It may be $f(v') = \emptyset$.

To simulate each j-th element of the set $f(v^{'})$, we will use the unit function

$$DI^{j}(time-\lambda)=\begin{cases}0, \text{ for } time<\lambda \\ . \\ 1, \text{ for} time\geq\lambda\end{cases} \tag{4.25}$$

In the case when an inverse change of the input value is required, the operator is used

$$\overline{DI^{j}}(time-\lambda)=1-DI^{j}(time-\lambda). \tag{4.26}$$

The value λ is defined by mapping g as a function $g(DI)$. The prediction for analog (continuous) inputs is based on the values specified by the mapping h of the laws of change corresponding to these inputs. Continuous input signal models can be represented by difference or differential equations. The state of such models changes continuously over time. In a general case, the analyzed i-th continuous input corresponds to the equation of the form

$$AI^{i}=F_{i}(time), \tag{4.27}$$

where $F_{i}(time)=h(v^{''}), v^{''}\in V_{I}, F_{i}(time)\in F$.

When there is a linear dependence of the input quantity on state variables, a difference equation is as follows:

$$AI^{i}_{time^{''}}=AI^{i}_{time'}+(time^{''}-time^{'})\cdot F^{*}_{i}, \tag{4.28}$$

where $time^{''}$ is the predictive time point;

$time^{'}$ – current moment in time;

$AI^{i}_{time^{''}}$ – the value of the entry AI^{i} at the time $time^{''}$;

F^{*}_{i} – a variable denoting the rate of change of the i-th input signal (can be specified as a function of other variables in the model).

An alternative way of AI^{i} determination is a definition of the derivative of a state variable. In this case, the derivative is integrated to obtain values AI^{i} at each step:

$$AI^{i}_{time^{''}}=AI^{i}_{time^{''}}+\int_{time^{'}}^{time^{''}}D^{i}dtime, \tag{4.29}$$

where $D^{i}=a\cdot AI^{i}+b$, which corresponds to the differential equation:

$$\frac{d(AI^i)}{dt} = a \cdot AI^i + b. \quad (4.30)$$

Usually, in simulation systems the values of state variables are calculated as a result of solving the corresponding equations at a separate point in time corresponding to the simulation step, with the obtained values being saved for later use. For a new point in time, information about the values of the state variables obtained in the previous step is used. The required variable is calculated step by step, which is then displayed as a simulation result.

In our case, when analyzing the input signals, it is advisable to solve the inverse problem and determine the moment in time when the state variable $AI^i_{time''}$ reaches a given threshold value. Then you can avoid additional performance losses associated with changing the model time with a small constant step, which is critical for real-time systems. For discrete signals, this will be a simple calculation of function $g(DI)$.

In the case of analog signals, there are two options:

1. Solution of the difference equation (4.28) with respect to time

$$\Delta time = time'' - time' = \frac{AI^i_{time''} - AI^i_{time'}}{F^*_i}. \quad (4.31)$$

2. Solution of Eq. (3.30).

From (3.30) we obtain

$$dtime = \frac{d(AI^i)}{a \cdot AI^i + b}. \quad (4.32)$$

After integrating both sides of Eq. (4.32), we will have

$$\Delta time = \frac{1}{a} \cdot ln|a \cdot AI^i + b| + C. \quad (4.33)$$

The integration constant C must be recalculated after each CEN cycle based on the measured value of the input AI^i. The first measured value should be taken as the initial condition for the first CEN execution cycle at $\Delta time = 0$.

We will define the dynamics of the states of the CO as the PLMP proceeding in parallel with the process of CEN functioning. Let us denote this process $v''(time)$ and give its formal definition for $|f(v')| > 0$:

$$v''(time) = (f(v'), \zeta_{v''}(time)) \quad (4.34)$$

where $f(v')$ is the set of input signals generated by the output v';

$\zeta_{v''}(time)$ – a vector of additional coordinates (delay times for inputs $f(v')$);

$|f(v')|$ – the number of inputs corresponding to the output v'.

As with the CEN process, the rate of change of $\zeta_{v''_i}(time)$ is minus one, i.e.,

$$\frac{d\zeta_{v''_i}(time)}{dtime} = -1. \tag{4.35}$$

The union of the sets $\{z_i(time)\}$ and $\{v''_j(time)\}$ forms a generalized PLMP describing the dynamics of the control process in the space of situations. The predicted time for changing situations will be calculated by the formula

$$time_{l+1} = time_l + \min_{i,j}(\xi_{lj}, \zeta_{li}), \tag{4.36}$$

where $time_l$ is the time of the l-th step of the prediction.

Expression (4.36) means that the advance of time at each step of the forecast is carried out to the nearest event in the models of the CEN and CO processes. After each such step, the time values of the additional coordinates are recalculated. It should be noted that in reality the moments of setting the input signals, calculated by the predictive model, may not exactly coincide with the beginning of the next real execution cycle. This fact will determine the accuracy of the forecast, which is caused by discretization of the continuous time model.

The set of formulas Φ defines the scope of the predictive definition. Based on the preliminary syntactic analysis of these formulas, sets of output and input signals are formed, which must be controlled during the forecast. In addition, information is extracted from these formulas regarding the threshold values of the input analog signals and the duration of the time intervals to be monitored. The mappings f, g, h are set in the form of correspondence tables. The mechanism for verifying the dynamic properties of CA, given in the form of DCTL formulas, is discussed in detail in the next subsection.

4.3.3. Dynamic Model Checking of DCTL Formulas on CEN

Following the verification task, we will use the implementation model presented in the form of CEN to check the properties of the CA.

If we do not consider the development of the control process in the future but restrict ourselves only to monitoring the given specifications in real time, the whole solution may consist in calculating the values of the corresponding DCTL formulas for the current situation. In this case, the verification task is reduced to checking the formulas on the state model determined by the change of markings at a time.

This approach, although it allows for control over the implementation of CA, is devoid of any predictive capability. To provide this opportunity, we will use the Receding Horizon Strategy, which we will project onto the predictive model

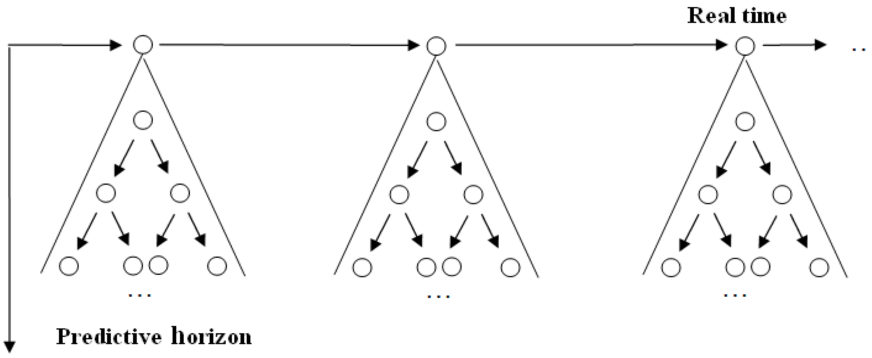


Fig. 4.4. Predictive model implementation diagram.

described above. We will assume that the predictive time horizon is determined by the PLMP obtained as a result of combining the list of events associated with the planned triggering of delayed transitions of CEN and events occurring in the CO. In this case, the total distance of the horizon at each cycle of the network execution will be determined as follows:

$$L_l = \max_{i,j}(\xi_{lj}, \zeta_{li}), \quad (4.37)$$

where l – the execution cycle number;

ξ_{lj} –j -th delay vector component CEN of CA in the l-th cycle;

ζ_{li} – i -th delay vector component of the input signals in the l-th cycle.

From Eq. (4.37) it follows that the predictive interval, determined by the horizon remoteness, can change from cycle to cycle, which actually leads to a floating predictive horizon as shown in Fig. 4.4.

Within the horizon set at each CEN execution cycle, it is possible to compute formula values on states that include temporal operators, including those with interval expressions. In this case, one counter is sufficient, which counts the real operating time of the CA. However, to control the intervals specified for the propositional variables, it is necessary to provide additional counters that would count the time the predicate, indicated by the propositional variable, remains in the true state.

Let us denote $CP \subseteq AP$ to be the set of propositional variables with interval expressions that are present in DCTL formulas for a particular CEN. Each variable $q \in CP$ will be assigned a time counter $cl \in C, |C| = |CP|$, which starts when the variable becomes true and resets to zero when the variable becomes false. When checking formulas on states that contain propositional variables with interval expressions, the values of the counters $cl \in C$ must be checked to see if they satisfy the given constraints.

More complex than checking formulas on states is checking formulas on trees. The set of situations that inherit the current situation s will be formed on the set of possible options for marking input places and values of input signals. Among the latter, let us single out those signals that are not considered in the predictive model. Let us designate them as V_I^-. These will be signals whose values cannot be tracked by CEN, since they are determined by non-deterministic external conditions of the control process. Such signals can arise, for example, as a result of the actions of an operator, dispatcher, etc. We will assume that all these signals are discrete. The influence of analog signals on the development of the control process can be set by giving them the status of discrete signals. For input places P_{in}, we will consider only the fact of marking these places without taking into account the values of the parameters, assuming that the parameters do not affect the calculation of the decision procedures for conditional transitions.

Thus, all the many options that can develop in any situation will be equal $2^{|V_I^-|+|P_{in}|}$. Taking into account the remoteness of the predictive horizon in the l-th cycle of execution, the total number of options will be $2^{(|V_I^-|+|P_{in}|)\cdot L_l}$.

The input signals that are included in the predictive model (denote them V_I^+) are taken into account when calculating the DCTL formulas according to their values, which are set at the time of verification. In this case, the analog signals are recalculated for each execution cycle according to the corresponding equations, and we can assume that their influence exactly corresponds to the condition of the absence of aftereffect. This may not be the case for discrete signals.

For example, let us consider a case where the delay λ until the input signal is set to the desired value is longer than the duration of a cycle. Such a case is shown in Fig. 4.6.

If the output signal for setting the input DI^i to the “1” state arrived at the moment of time $time_{l+0}$, then the corresponding reaction of the CO should be detected after a time interval equal to λ. At each cycle, the predictive model should take into account

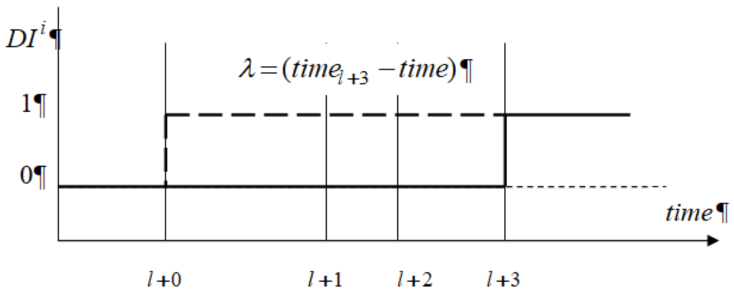


Fig. 4.6. Timing diagram of the prediction for a discrete signal.

Table 4.3 Propositional Variables

Variable	Predicate
q_1	$sensor1 = 0$
q_2	$ready = 1$
q_3	$sensor2 > 5.1$
q_4	$alarm = 3$

the fact of DI^i signal generation, but not after interval λ, but after an interval equal to

$$\lambda' = (time_{l+j} - time_{l+0}), \tag{4.38}$$

where j– the number of the cycles following the initialization cycle of the output signal.

Let us look at the procedure for calculating the value of the DCTL formula for a particular situation. If we assume that the current situation satisfies the specified interval constraints imposed on the operators, then the formula under the temporal quantifier is subject to verification. To calculate it, we will use BDDs; however, each binary propositional variable will be compared with predicates given on the variables CEN. This correspondence can be displayed in a separate table. As example, for formula $\varphi = AG((q_1 \wedge q_2) \rightarrow X(q_3 \rightarrow q_4))$ see Table 4.3.

A decision diagram based on Table 4.3 can be displayed in the form shown in Fig. 4.7.

$$\begin{tikzcd}
 & & & \phi = AG(\phi_1^3)\P \arrow[d] & & & \\
 & & & \phi_1^3 = ((q_1 \wedge q_2) \rightarrow X(q_3 \rightarrow q_4)) \arrow[dll] \arrow[drr] & & & \\
 & \phi_1^1 = q_1 \wedge q_2 \arrow[dl] \arrow[dr] & & & & \phi_1^2 = X(q_3 \rightarrow q_4)\P \arrow[d] & \\
\phi_1^0 = q_1 & & \phi_2^0 = q_2 & & & \phi_2^1 = (q_3 \rightarrow q_4) \arrow[dl] \arrow[dr] & \\
 & & & & \phi_3^0 = q_3 & & \phi_4^0 = q_4
\end{tikzcd}$$

Fig. 4.7. Binary decision diagram.

Following this diagram, the DCTL formula is calculated in the following order:

- first, all formulas of depth 0 are calculated: $\varphi_j^0, j = \overline{1,4}$ – predicates denoted by propositional variables;
- then subformulas of depth 1 – φ_1^1 and φ_2^1 are calculated;
- then the sub-formula of depth 2 – φ_1^2 is calculated;
- then the sub-formula of depth 3 – φ_1^3 is calculated;
- and finally the formula of depth 4 is calculated – the original formula φ.

Since the formula φ_1^2 contains a temporal operator, its calculation should be postponed until the next cycle. This also entails a delay in the calculation of the initial formula, which, due to the presence of the quantifier, should generally take the final value only after the results of calculations on all paths of the situation tree during prediction. When calculating each of the propositional variables for which interval constraints are indicated, the conditions for their fulfillment according to the readings of the counters must be checked.

The preliminary clarifications made regarding the specifics of calculations during the implementation of the proposed predictive method allow us to go directly to the description of the algorithm for checking DCTL formulas. This algorithm includes two blocks of basic operations: an initialization block and a computation block. A detailed description of the dynamic model checking algorithm is given below.

The initialization block is executed before the start of the CA and includes the following steps:

1. Plotting BDD for each given DCTL formula. For each formula $\varphi \in \Phi, i = 1, \ldots, |\Phi|$, define a set of formulas of depth 0 (propositional variables) $\Phi^0 = \{\varphi 0_j | \varphi 0_j = q_j\}$, a set of formulas of depth 1 $\Phi^1 = \{\varphi 1_j\}$, etc. We will assume that the formula $\varphi = \{\Phi^i\}$, where Φ^i is the set of i -depth formulas included in the formula φ.

2. Allocation of a subset of input signals controlled by the predictive model. Using the predictive model, select a subset of the input variables $V_I^+ = \cup f(v'), v' \in V_o, f(v') \neq \emptyset$.

3. Calculation of threshold values for input signals. For each analog input signal $v \in AI \cap V_I^+$ present in the table of propositional variables, a threshold value $\overline{v}$ is assigned equal to the constant written in the predicate corresponding to this signal.

4. Creation of CEN model of CO. Initialize the creation of a CEN of CO by setting the number of X-type transition output places and Y-type transition input places to $|V_I^+|$. In the transition-queue T1, using the token generator, $|V_I^+|$ tokens are placed

that simulate the input signals. For all transitions of type T, the value of the activation function $\alpha(t) = 0$ is set.

5. Allocation of a subset of input signals that form computation trees $V_I^- = V_I \backslash V_I^+$. This set, together with a set of input places P_{in}, will determine the options for the development of situations when constructing situation trees, i.e., form a set $V_I^- \cup P_{in}$.

The block of calculations is executed in each cycle of the CA execution and includes the following steps:

1. Fix the system execution cycle time. Store the system time of the current run cycle in a variable *TIME*.

2. Form a set of verifiable formulas. Create set of formulas $\Phi^{'} = \Phi$.

3. Choose a postponed situation. Extract place markings M_s from a variety of pending situations $S_H = S_H \backslash M_s$. If $S_H = \emptyset$, then go to item 12.

4. Build an s-tree for the current situation. Form a set of binary vectors $V^s = \{V_i^s\}, i = 1, \ldots, 2^{|V_I^-|+|P_{in}|}$ of lengths $|V_I^- \cup P_{in}|$ that differ in at least one component.

5. Perform the cycle of testing – s-paths. If $V^s \neq \emptyset$, then select an element, remove an element from:, otherwise go to Step 3.

6. Check the compliance of the current system time with the interval constraints specified for propositional variables. If $\exists q \in \Phi^0 . cl \notin (\sim c)$, where cl– the value of the time counter for variable q; $(\sim c)$ – an interval expression that specifies the domain of definition of a propositional variable, then go to Step 5.

7. Check whether the current system time matches the interval limits set for temporal operators. If $\exists \varphi \in \Phi^i . TIME \notin (\sim c)$, where $(\sim c)$ – an interval expression specifying the domain of definition of the temporal operator, then go to Step 5.

8. Activate the CEN model of CO. For CEN transitions $T_{5+j}, j = \overline{1, K}$ of the CO model that are not in delay, the delay time is assigned according to the following rule:

- for discrete signals – $\forall DI^i \in V_I^+ . \tau(t) = g(DI^i)$;
- for analog signals – $\forall AI^j \in V_I^+ . \tau(t) = \Delta time$, where $\Delta time$ is calculated according to Eqs. (4.31) and (4.33).

For all transitions of type T, the value of the activation function is set in $\alpha(t) = 1$.

9. Calculate the values of formulas. For each formula $\varphi \in \Phi^{'}$, pass through its BDD

and sequentially calculate the values of all subformulas and the formula as a whole, starting with subformulas of depth 0. In this case:

- if any subformula of the formula φ contains a temporal operator G and its value is equal to "false", then replace this subformula with "false";

- if any subformula of the formula φ contains a temporal operator F and its value is "true", then replace this subformula with "true";

- if any subformula of a formula φ contains a temporal operator X, then remember the values of all subformulas, calculate the value of the formula taking into account the values of the previous step;

- if any subformula of the formula φ contains a temporal operator U, then save all the values of the formula φ_1 obtained at each step of prediction, calculate the value of the subformula when the value of the formula φ_2 is "true", and replace the subformula with the obtained value;

- if none of the subformulas of the formula φ contains temporal operators, then assign the calculated value $\Phi^{'} = \Phi^{'}\backslash\varphi$ to the formula, otherwise assign the formula a value calculated without taking into account the temporal operators;

- if all subformulas of the formula φ containing the quantifier E have received the value "true", then assign this value to the formula, $\Phi^{'} = \Phi^{'}\backslash\varphi$;

- if any subformula of the formula containing the quantifier A received the value "false", then assign this value to the formula, $\Phi^{'} = \Phi^{'}\backslash\varphi$;

10. If $\Phi^{'} = \emptyset$, then go to item 12.

11. Pass through the forecast horizon. If $\xi = \emptyset \wedge \zeta = \emptyset$, there are no delayed transitions to either CEN of CA or CEN of CO, then go to Step 5, otherwise

- put the current marking M_s in a lot of pending situations S_H;

- calculate the new value of the model time according to the formula

$TIME = TIME + \min_{i,j}(\xi_j, \zeta_i)$,

where ξ_j – the delay time of the j-th transition to the CEN of CA;

ζ_i – the delay time of the i-th transition to the CEN of CO;

- make calculations for CEN of CA and CEN of CO;

• go to item 6.

12. Complete the algorithm. Assign the calculated values to all formulas.

4.4. Summary

The control algorithms specified by the control E-networks can be considered reactive systems for which there are known classifications of dynamic properties based on safety and survivability requirements. The most appropriate means of specifying these properties are temporal logics, in particular, Timed Computation Tree Logic (TCTL), which, nevertheless, does not allow taking into account interval dependencies between events occurring in the control system.

To assess the dynamic properties of control systems in real time, a set of tasks must be solved related to clarifying the temporal properties of control E-networks, developing a dialect of temporal logic focused on control processes, and building a predictive model that takes into account the specifics of interaction between the control system and the external environment.

The temporal model of the control E-network includes a time model, a state model and a computation model that determine the nature of the network behavior in time. As a time model, a discrete-event model is adopted, which is characterized by a change in the basic states of the network at times determined by the duration of the CA execution cycle.

To ensure the determinism of the CA behavior in emerging situations, it is proposed to use the dynamic synchronization procedure, which is built on the observance of the introduced set of axioms that specify the cause-and-effect relationships between network transitions. The reliability of the result obtained is confirmed by the proved theorem on the conditions for observing the property of deterministic behavior of the control E-network.

In order to specify the complex dynamic properties of CA, a new kind of temporal logic is proposed – the interval logic of the DCTL computation tree, which allows taking into account the temporal domain of definition of temporal operators and individual propositional variables using interval expressions. The semantics of DCTL logic formulas can be defined both on individual situations and on a tree of situations, taking into account the possibility of non-deterministic behavior of the CO.

The predictive model used to analyze the dynamic properties of the ontrol system, specified using DCTL logic formulas, implements the Receding Horizon Strategy and makes it possible to take into account the hybrid nature of the control system behavior, modeling it by means of an automatically created control E-network. The coordinated execution of the CA model and the CO model is the essence of the used dynamic verification mechanism, which applies a binary decision tree to analyze formulas on predictable situations.

The developed algorithm for checking DCTL formulas on control E-networks provides dynamic verification of the control process in real time, which creates conditions for a timely response to emerging situations in order to prevent undesirable developments by dynamically changing the control system.

Chapter 5. Recovery Models and their Construction

Like predictive models, recovery models must provide the control system with missing information that cannot be obtained from the sensors through the input data transmission channels. In this chapter, such a problem is solved by using models created by means of computer graphics and computer vision, which are directly embedded in the control loop. Computer recovery models make it possible to fill in the missing information regarding the spatial location of the CO at the macro and micro levels and thereby create conditions for the implementation of methods of adaptive and multi-agent control based on model data. Recovery models play an especially important role when the operator involved in decision making is included in the control loop.

5.1. Cases of Computer Recovery Models

The role played by the recovery models in the CS of IMS is to implement a closed control loop even in cases where the real capabilities of sensor devices do not allow it. In such cases, the CO is replaced by a model, which should provide the CS with all the missing information required by the conditions of the CS functioning. Most often, such situations arise in control tasks with space-temporal information, which concerns, for example, the relative position of the CO and its environment, the location of the tool performing a complex technological operation on the surface of the product, the spatial direction of energy fields, etc.

5.1.1. Man in the Control Loop

Undoubtedly, the most popular recovery models are the ones that involve the active participation of the operator in the management of the system, as is the case, for example, at nuclear power plants, when using deep-sea manipulation robots, in the cockpits of airliners. Recovery models should be able to simulate the environment and provide the operator with all the missing information through detailed real-time reproduction of the state of spatial objects and its visualization. What cannot be seen in real life must be available on the recovery model used. This use of recovery models can be characterized by the well-known term “Man-in-the-Loop” (MIL) – “operator in the control loop” (Pollini and Innocenti, 2000).

An example of building a control system using recovery models according to the MIL version is shown in Fig. 5.1. The main distinguishing feature of a control system,

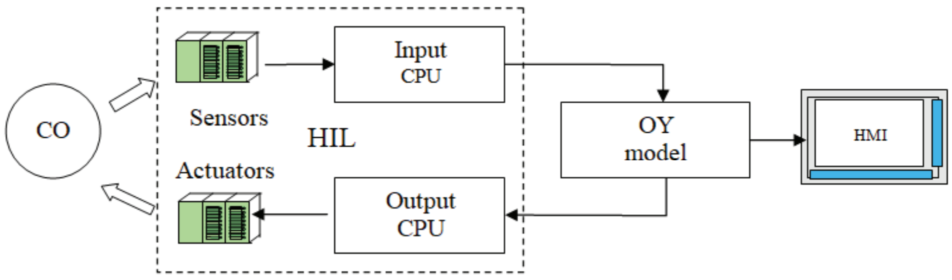


Fig. 5.1. MIL case of recovery model usage.

which contains an operator, is the presence of a display (monitor) with a Human Machine Interface (HMI).

The input, or communication, processor processes information from the interface modules, which gives a partial view of the state of the CO. It can be data on the spatial location of the CO, for example, the values of the current coordinates, or indications of navigation devices (direction of movement, speed, etc.). Information about the state of the environment can also be partially obtained: temperature, air pressure, and vacuum level. However, all this may not be enough for the operator to make a decision, since all the information listed does not create the effect of presence, which is achieved only with direct observation of the process. But even if such observation is feasible, it may also be limited by a view angle, image scale, or even hidden by nearby objects.

The visual computer model, which is displayed on the monitor screen, is able to reproduce by means of graphical imitation all the details of the CO state in real time and thereby compensate for the arising limitations of the observation system. Typically, such a model is a three-dimensional representation of an op-amp immersed in a synthetically generated environment. It allows the operator to observe the object in all positions of interest using the controls.

Such a representation of CO in combination with a synthetic environment most fully corresponds to the modern concept of Virtual Reality (VR) (Delaney, 2017). Allowing with the same success for the simulation of quite tangible, as well as more abstract objects (for example, magnetic fields or turbulent flows), virtual reality technology helps reproduce the entire production process from the development of the product concept to the stage of its operation. At the same time, virtual reality provides, as a rule, the creation of more complex models than when using other methods. For example, when working with a virtual model, the motor housing can be made transparent so that its internal structure is visible for observation.

It should be noted that virtual reality as recovery models can be used at various stages of the product life cycle:

- creating products, when the simultaneous work of several departments at different stages of design is required;
- forming tasks for technological lines without use of real product samples;
- testing the created control programs;
- evaluating the developed plans, etc.

In all cases, the use of virtual models provides not only high quality control of the design process, but also significant time and cost savings, eliminating the need to create physical prototypes.

Another important application of the MIL control option is personnel training. Simulation of real operating conditions of systems can be used for training pilots and operators of complex industrial installations. The virtual object will allow you to work out all control operations, including even those that arise in the event of breakdowns, equipment failure or due to extreme external conditions of operation. It also saves time, money, and at the same time prepares you to act in the most difficult situations.

Finally, the most important aspect of using virtual recovery models is the implementation of remote object management. Surveillance cameras that are actually used can be replaced by virtual computer models that change synchronously with the actions performed. In this case, the location of the observation means can be chosen by the operator independently. The main requirement that such recovery models must meet is high modeling accuracy.

5.1.2. Hardware in the Control Loop

The need for recovery models may not be limited to the operator. Reconstruction may be subject to the conditions of real processes when they are evaluated by hardware solely from the obtained images. This is often the case, for example, in automatic detection and tracking systems.

For example, the recognition of cartographic information in missile homing systems, the detection and classification of target marks on the radar screens, the training of the tool paths along the marked lines on the surface of the products and the tracking of the specified trajectories of movement are the tasks that are solved by hardware automatically directly from the digitized images of objects obtained.

These images can either be displayed on monitors for observation by operators, or hidden for observation, remaining in the computer memory. The important thing

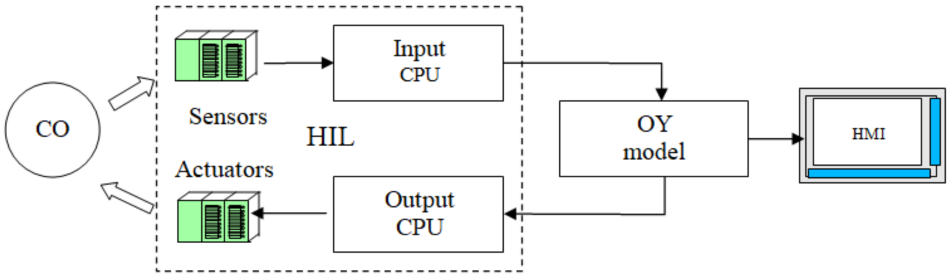


Fig. 5.2. HIL case recovery model usage.

is that these models are an integral part of the control process, closing the feedback loop. Such use of recovery models will be called "Hardware-in-the-Loop" (HIL).

Figure 5.2 shows a recovery model usage diagram for HIL case recovery model usage.

A distinctive feature of this scheme is that the HMI no longer plays a decisive role and is only an auxiliary element in the CS structure. All actions for analyzing the situation and making decisions are carried out automatically by hardware, which can combine sensors (observation equipment) and actuators, together with controllers that ensure their work. This means that it is not the creation of virtual reality pictures that comes to the fore in the construction of reconstruction models, but the acquisition of a high-precision image of the CO and its recognition.

The key point for the successful functioning of a CCS that implements the use of recovery models according to the HIL option is the high speed of input/output channels and high-speed data exchange between hardware and software. Computations related to the recognition and classification of digitized images should be completed no later than it is provided by the cycle of functioning of the hardware complex. Real-time operation in sync with the control program is a fundamental requirement for HIL recovery models. This can be achieved using highly efficient digital image processing and recognition algorithms. No less important is the speed of data transmission channels.

5.1.3. Recovery Model Design

An analysis of the use cases of recovery models in the control loop shows that their implementation is associated with the creation of virtual reality display models and image recognition models. In the MIL variant, display models prevail, while in the HIL variant, the main burden falls on the recognition models.

The main requirement for display models is the maximum compliance with the situation arising in the control process. In addition, the formation of virtual mappings should occur in real time, ensuring that the model is synchronized with the position of the CO. In addition to display models, a requirement for extensibility can be put forward – the possibility of complicating the virtual world model with new objects and properties.

Recognition models, in turn, should provide automatic recognition of the current state of the CO based on its images obtained using observation tools. Since the problem of recognition is solved automatically, the main requirements for it are efficiency and flexibility. Efficiency means that the time of recognition (acquisition, processing and analysis of the image) should not increase the duration of the control cycle, and flexibility implies the ability of the system to recognize arbitrary objects both with and without preliminary training.

The listed requirements make it possible to formulate a list of tasks that must be solved when building recovery models. Mapping models in the structure of reconstruction models should perform the following functions (Dorf and Bishop, 1998):

1. To simulate virtual three-dimensional representations of the CO state and its position in the surrounding space. At the same time, the required accuracy in the presentation of image details that are important from the point of view of the control process must be ensured, and the use of color gamut to convey the internal state of the displayed objects.

2. To provide an opportunity for the operator to actively influence the resulting virtual display by changing the viewing angle for inspecting any point of the CO and objects of the external environment, as well as by changing the image scale.

3. To scan the states of the control process, based on the data received from the sensors. In fact, it is necessary to give the operator the possibility of situational control of the object, based on the provided virtual display.

The functions of recognition models used when working with images differ from those inherent in the classical formulation of the mathematical theory of recognition (Silbert and Hawkins, 2016). In a pure form, the theory of recognition considers situations that are characterized by the absence of order relations specified on the set of object attributes. It is assumed that each object is identified with some point of the multidimensional attribute space, and the class of objects is represented by a compact set of such points. The task of recognition is to assign an object to one or another class, and knowledge of the formal description of the object itself is not required.

In contrast to this approach, in image recognition, mathematical problems arise associated with the formal description of an image as an object of analysis (Schyn et al., 2003). In this case, information should be used that reflects the mechanism of image formation – both the image as a whole and the objects represented on it.

The structure of the image allows determining what objects can be distinguished in the image, how elementary they can be, and in what relationships they are located. Moreover, image analysis is only part of the overall recognition task in this case. Based on the information contained in the image, a model should be obtained that gives an idea of the shape of the object. In the future, this model can be used in CS when implementing one or another control method.

Thus, the recognition models used in the control loop should provide:

- construction of an image model as an object of mathematical analysis;
- formal description of the structure of the recognition object;
- conversion of image models to a form convenient for recognition;
- restoring the shape of an object using image models;
- training on reference images.

It should be noted that the considered recovery models can closely interact with each other in the process of solving control problems. The expediency of the connection between them follows from considerations of the maximum information content of the display subsystem, on the one hand, and the high efficiency of the recognition subsystem, on the other. In particular, the use of these models makes it possible to implement methods of adaptive and multi-agent control of the IMS, which are based on the acquisition and analysis of visual information. Moreover, what is especially important, the application of these methods is possible at all levels of control, as well as with the involvement of predictive models.

5.2. Virtual Reality Models

5.2.1. Display Subsystem Structure

The display subsystem is intended for virtual display of objects. Moreover, both real objects and models of objects of the virtual world are subject to display. The block diagram of the display subsystem is shown in Fig. 5.3.

The main components of the display system are:

- monitor – a set of hardware for the presentation of virtual reality;
- frame generator – algorithms for creating virtual display frames;
- images of objects in virtual space;

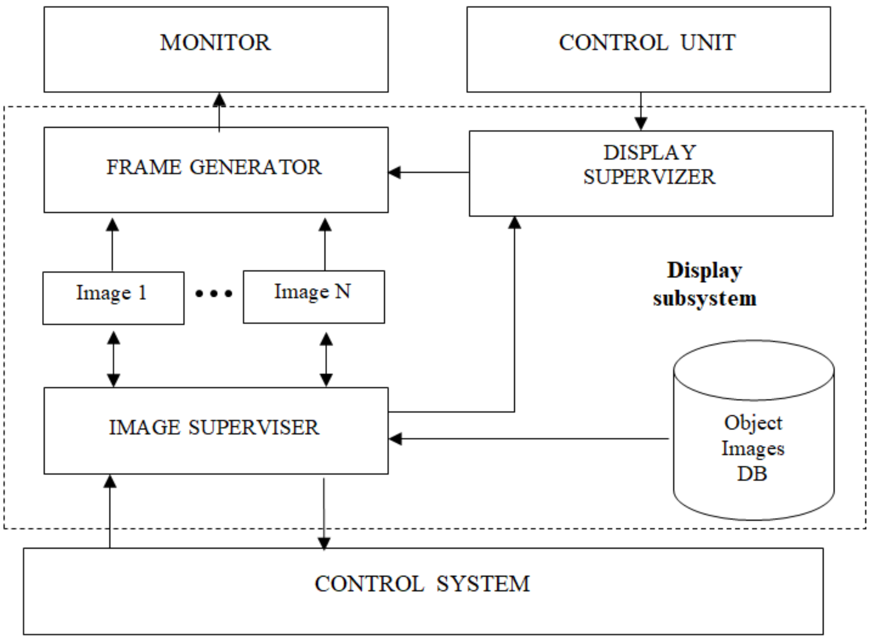


Fig. 5.3. Block diagram of the display subsystem.

- image supervisor – an algorithm for managing the interaction between object images;
- control unit – a set of hardware for receiving control commands from the operator (sensors, drives, consoles, etc.);
- display supervisor – an algorithm for controlling the viewing angle in the virtual world;
- database (DB) of object images containing three-dimensional models of objects from which frames are formed.

The display subsystem has three main elements that perform control functions: frame generator, image supervisor, display supervisor. Let us consider them in detail.

5.2.2. Frame Generator

The block diagram of the frame generator is shown in Fig. 5.4. The main components of the frame generator are the following:

- OpenGL Graphics API – libraries of graphic functions of the OpenGL standard, which transform the commands of the standard into the form required for the

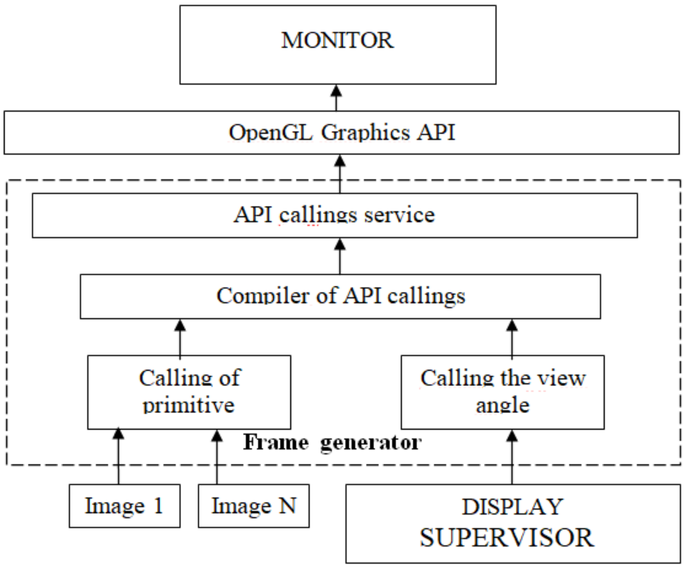


Fig. 5.4. Block diagram of the frame generator.

complex of display hardware;

- API callings service – an extension of the implementation language that allows using the OpenGL standard;

- Compiler of API callings – a language extension that converts higher-level calls into commands of the OpenGL standard;

- Calling of primitive – language words that call the image of an object consisting of primitives;

- Calling the view angle – language words that control the view angle.

5.2.3. Image Supervisor

The structure of the image supervisor is shown in Fig. 5.5. The main components of the image supervisor are the following:

- Image access API – provides access to the attributes of images;

- Display rule analyzer – analyzes the rules of behavior of objects and executes a request to the image attributes to change them in accordance with the analysis results;

- Object behavior rules – a set of rules that determine the features of the behavior

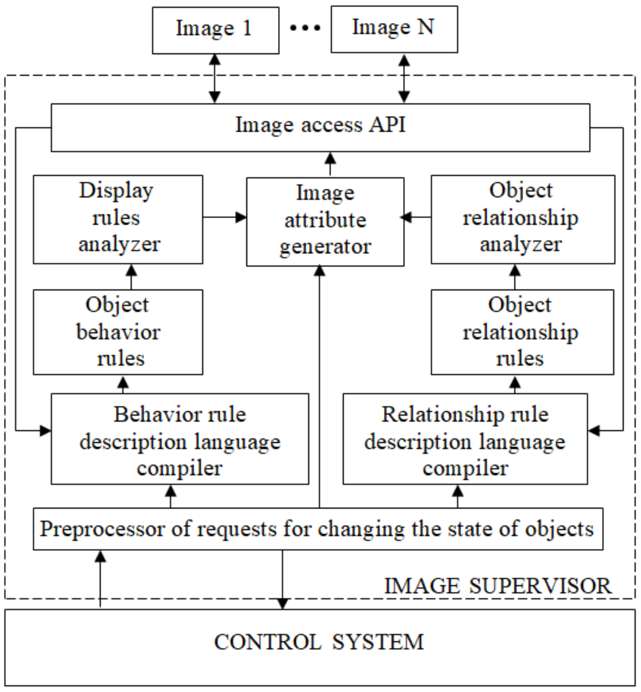


Image 1
Image N
Image access API
Display rules analyzer
Image attribute generator
Object relationship analyzer
Object behavior rules
Object relationship rules
Behavior rule description language compiler
Relationship rule description language compiler
Preprocessor of requests for changing the state of objects
IMAGE SUPERVISOR
CONTROL SYSTEM

Fig. 5.5. Block diagram of the image supervisor.

of objects (an object can be transparent for other objects, hidden, etc.);

- Behavior rule t description language compiler – compiles the rules for mapping objects written in the object description language into expressions in the programming language;

- Image attribute generator – executes a request via the image access API to change the attributes for the corresponding images;

- Object relationship analyzer – analyzes the rules of relations between objects and makes a request through the API access to the attributes of images to change them in accordance with the results of the analysis;

- Object relationship rules – to define relations between objects. For example, when an object rotates around its axis, then all objects associated with it can rotate. At the same time, the movement of an object does not affect objects that are not associated with it;

- Relationship rule description language compiler – compiles the relations between objects written in the language of the rules of relations into the rules of relations for the programming language;

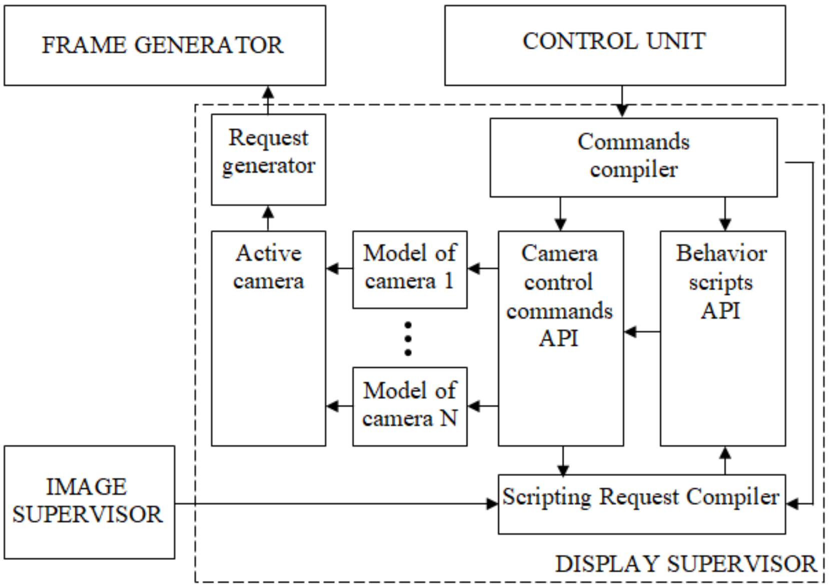


Fig. 5.6. Block diagram of the display supervisor.

- Preprocessor of requests for changing the state of objects – state change requests are divided into three categories: requests to change object display rules, requests to change object attributes, and requests to change relations between objects.

5.2.4. Display Supervisor

The structure of the display supervisor is shown in Fig. 5.6.

The main components of the display supervisor are the following:

- Request generator – creates a request to change the position of the surveillance camera;

- Active camera – model of the active camera from which the observation is carried out at the current time;

- Model of camera 1, .., Model of camera N – a stack of cameras that are needed to quickly switch from one place of virtual space to another while maintaining the position at the current point;

- Commands compiler – converts commands from the control unit hardware complex into camera model commands;

- Camera control commands API – language extension for managing a stack of cameras;

- Behavior scripts API – language extension for script execution. A script is a sequence of commands, the execution of which is similar to a certain sequence of control actions, for example, a script of a standard traversal of some part of the virtual world;

- Scripting request compiler – scripting requests in the virtual world are compiled into scripts in a programming language.

5.2.5. Technics for Constructing Display Models

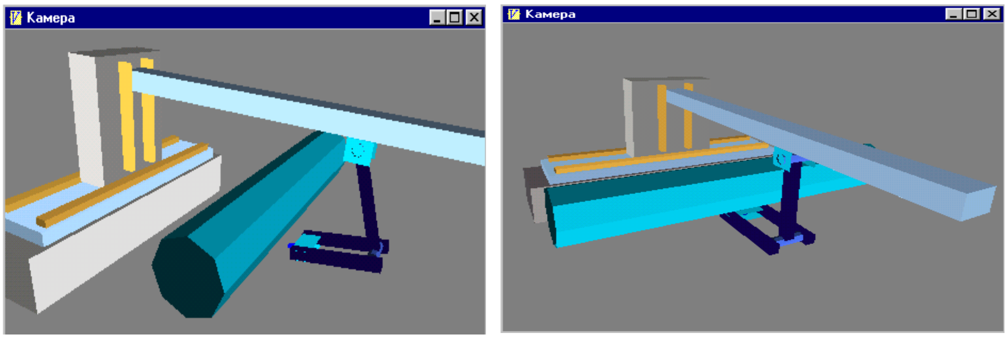
Three-dimensional images for the display subsystem can be developed in two main ways:

- created from a set of standard objects, which are a straight parallelepiped, cube, sphere, cone, pyramid, cylinder, etc.

- generated according to the given descriptions using any graphics software system that supports standard 3D formats: Autodesk 3dsMax, 3D Bruce Models, NuGraf Rendering System, OpenGL, etc.

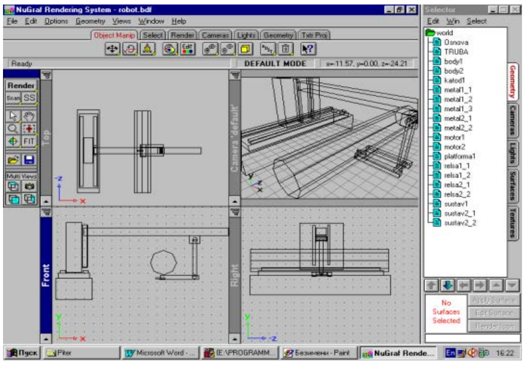
A view of a welding robot model built from standard objects created using the OpenGL library is shown in Fig. 5.7.

The basic objects (primitives) in this case are a cylinder and a straight parallelepiped. The number and position of the primitives used are specified using parameters, and the generated object is represented as a tree of inherited primitives.



a) view angle 1 b) view angle 2

Fig. 5.7. Welding robot display model view.


a) image design

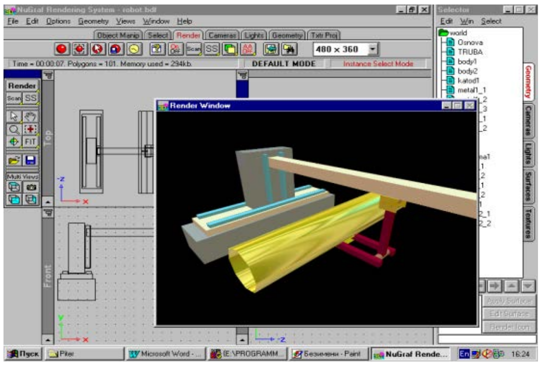

b) image representation

Fig. 5.8. Welding robot model development process.

The process of developing such a model in the NuGraf Rendering System is shown in Fig. 5.8.

The image database can be expanded by introducing new objects. Figure 5.9 shows the process of developing scene objects for a welding chamber using Autodesk 3dsMax.

5.3. Image Recognition Models

An image is usually understood as one or more projections of a spatial object obtained using an observation system. Therefore, the task of automatic recognition deals with flat images with all their inherent processing features. This applies to both the procedure for obtaining an image and its preliminary filtering, and recognition of the contours of objects. The end result of recognition should be the restoration of information about the state of the object, which is necessary for making a managerial decision and, above all, the shape of the object.

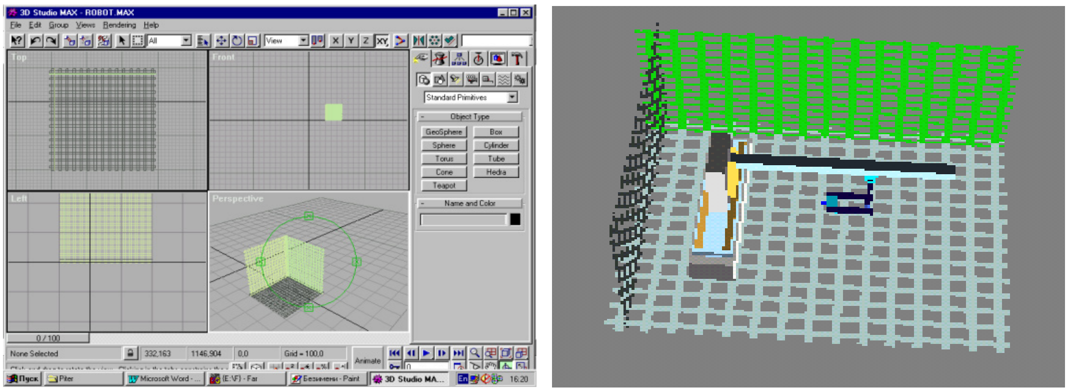


Fig. 5.9. Expansion of the virtual world with new models.

5.3.1. Recognition Subsystem Structure

Consideration of the image recognition models used in the control loop will be carried out on the example of the CCS for electron beam welding machines.

The problem that arises during the welding process is the precise hitting of the electron beam in the middle of the joint between the edges of the products being welded. Considering that the joint can have a width of tenths of a millimeter, it is impossible to do this process without using special observation means. In addition, welding must be carried out inside a vacuum chamber, which isolates the operator from direct observation of the welding process. It is possible to ensure contact with the joint in such conditions only by obtaining and processing enlarged images of the joint. Moreover, it must be done in the process of moving the electron beam over the surface of the product synchronously with the movement and taking into account the interference caused by the influence of strong electric and magnetic fields. The use of television cameras in this case is limited only at the stage of preparation for welding and is excluded at the time of welding.

Thus, the task of image recognition in the welding process includes:

- obtaining an image of the product surface at the welding site;
- filtering of the received images;
- recognition of the joint area in the resulting image;
- accurate detection of the middle of the joint;
- restoration of the joint trajectory.

The functional structure of the recognition subsystem, which provides the solution of the recognition problem in this setting, is shown in Fig. 5.10.

This structure reflects the sequence of functional tasks solved in the process of image recognition. It should be borne in mind that there are no universal models of

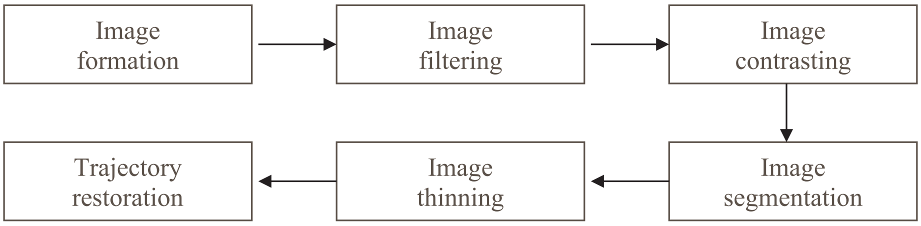


Fig. 5.10. Functional structure of the recognition subsystem.

images that have the necessary degree of constructivism for all recognition problems. In this regard, at each of the stages, an image model of a certain type is formed, which is then converted into a model of the next level. We can say that a hierarchy of image models is formed with its inherent transformation procedures. If we denote an image I, and the transformation applied to it T, then the whole recognition process will consist in defining on the equivalence classes $\{I\}$ a set of algebraic transformation systems $\{T\}$ such that, $T: I \to I^*$ where $I, I^* \in \{I\}, I \neq I^*$.

The advantages of the proposed approach are that the decomposition of the general recognition problem into a sequence of stages makes it possible to reduce the complexity of the solution due to the sequential use of various image models and their mutual transformations. It should be borne in mind that all stages of recognition should be performed at the rate of development of the process synchronously with the movements of the electron beam. Let us consider these models in more detail.

5.3.2. Sampling and Quantization Models

To process an image on a computer, it must be converted to a finite set of numbers. However, the image is primarily some kind of signal that conveys information to the surveillance system. Therefore, the task of forming an image is a task of signal processing for the purpose of its sampling.

Among discrete signaling models, the most widespread models are the ones that are focused on the positional representation of image elements. In them, the image is a matrix $\|s_{ij}\|, i, j = \overline{1, N}$, each element (pixel) of which contains some characteristics of the original image. The main advantage of the matrix representation of the image is to preserve the structure of the original image. Such representation of video data is also called direct, and it allows for easy implementation of their digital processing (Gonzalez and Woods, 2018).

When processing signal information, two basic procedures are used: sampling and quantization. The goal of sampling is to build an image matrix for some limited space. In the course of further processing, only the signal values recorded in the elements of the image matrix will be used. The classical way of discretizing continuous signals, which are real images, is the application of the Nyquist–Shannon–Kotelnikov theorem. Despite the fact that this theorem admits a trivial generalization to the two-dimensional case, it is of little interest in computer observation systems. When working with images, the sampling rate should be significantly higher than that determined by the results of spectral analysis.

Quantization consists in mapping the values of brightness (for black and white images) and color (for color images) into integers. It is correct to assume that 8 bits (256 levels of brightness) are required to represent most images, and in many practical tasks, in particular, for welding robots, 4 bits (16 levels) are sufficient. But fewer bits guarantee less processing time, which is critical in control tasks.

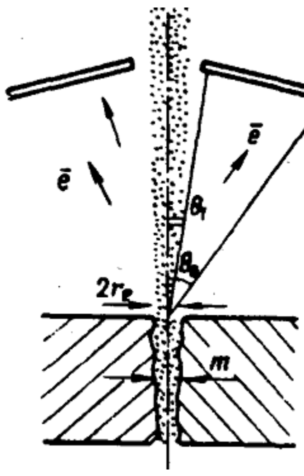


Fig. 5.11. The principle of operation of the joint recognition equipment.

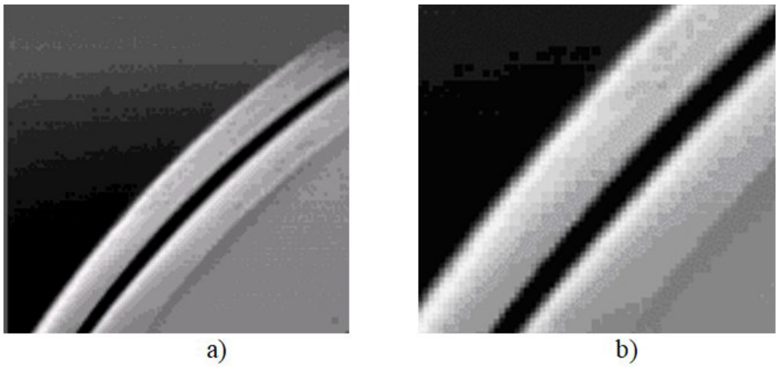
Currently, spatial sampling of images is carried out, as a rule, using hardware by means of electronic scanning. Moreover, in these cases, sequential processing is sufficient. As a result, the two-dimensional function of brightness $B(x, y)$, describing a flat image, is converted into a one-dimensional function of time $B(t) = B[x(t), y(t)]$ using line-frame scanning. Such systems are usually called raster systems. With the cyclical operation of the observation equipment, all points of one frame have the same reference time.

For example, in an EBW machine, a special device is used as such observation equipment. This device has a number of advantages over other surveillance means, including television (Nazarenko, 1993). Its principle of operation (Fig. 5.11) is based on the measurement of the energy of secondary electrons, which, after being reflected from the joint, are captured by a special sensor installed at the end of the electron gun.

As a result of these measurements, a signal is generated, and then it is processed

Table 5.1 Parameters of the generated Joint Image Frame

Frame parameter	Value
Image matrix size, pixels	128×128
Scanning area size, mm	20×20
Pixel size, mm	0.16
Pixel quantization size, bit	4
Frame scan time, msec	4



without increasing in ZOOM mode.

Fig. 5.12. View of the joint image on the monitor screen:

by a special electronic board. The board converts in real time the signals of the secondary electron sensor into digital codes, which are sent to the computer in the form of separate frames. The computer program, based on the received data, generates an image frame, which has the parameters shown in Table. 5.1.

An image frame is saved in the computer memory for further processing. It can also be displayed on the monitor screen for operator observation of the joint. Although this operation is not required when using the HIL variant, it is quite convenient because it allows you to trace the entire recognition process. In some cases, as, for example, when teaching the paths of joints, it is mandatory.

The view of the joint image formed using the sampling and quantization models is shown in Fig. 5.12.

The joint on the monitor screen looks like a narrow dark strip, the smooth surface of the products being welded – a light area on both sides of the joint. Both the joint and the surface of the products have a fairly uniform texture. In the computer memory, information about the image frame is presented as a two-dimensional array (matrix). The number in each cell of the matrix corresponds to the brightness of the dot, which depends on the value of the flux of secondary electrons captured by the sensor. There four bits are used to define color results in 16 shades of gray with brightness ranging from 0 (for the darkest pixel) to 1:5 (for the lightest pixel).

For a more visual representation of the image, each pixel of the matrix is represented by four pixels on the monitor screen, which at a set resolution of 640×480 gives an image magnification of about 1: 5 (Fig. 5.12 a). When the ZOOM option is enabled, the size of the scanned area is halved, which corresponds to a representation scale 1:10 (Fig. 5.12 b).

It should be noted that technical systems do not have the task of lowering the image contrast by using histogram equalization procedures. This circumstance is a positive factor, since no additional time is required for the primary image processing.

5.3.3. Filtering Models

The need to filter interference in technical surveillance systems is primarily associated with the effect of various physical fields caused by the operation of equipment and nearby mechanisms. Therefore, any technique for accounting for interference should be adequate to the real situation and the adopted recognition concept, since the general procedure for eliminating interference by introducing a certain random scatter of feature values in some cases leads to the appearance of additional errors.

The simplest technique is linear space-invariant filters (threshold, moving average, recursive, etc.), which are widely used in time signal processing. However, the relationship between noise removal and blurring of the edges of an image means that line filters should be used with caution when processing images. In particular, they are not suitable for our task, which requires a clear image of the joint against the background of product surfaces. In technical systems, it is more often required not to smooth out the contours of areas, but to eliminate the gaps in image areas and changes in texture that occur due to interference.

Filters, the use of which precisely removes noise from areas located inside the image regions, without causing blurring of its edges, are much more complicated than a simple linear transformation. They form a class of non-linear filters. Among the latter, the most widely used for spatial image filtering is the method of anisotropic filtering (Olano, 2001). A discrete interpretation of this method leads to the relation

$$\tilde{a}_{ij} = \Lambda\left[\sum_{v=-N_a/2}^{N_a/2}\sum_{\xi=-N_a/2}^{N_a/2} a_{i+v,j+\xi}w_{v\xi} - \eta\right], \tag{5.1}$$

where $\tilde{a}_{ij}$ – an element of the filtered image matrix located at the intersection of the i-th row and the j-th column;

$a_{i+v,j+\xi}$ – an element of the image matrix distorted by noise, which is located at the intersection of the $(i+v)$ -th row and the $(j+\xi)$ -th column;

$w_{v\xi}$ – an element of the aperture (additional matrix in size $N_a \times N_a$) located at the intersection of the v-th row and the ξ-th column;

η – a filtering threshold (constant);

Λ – a threshold function.

For complete filtering, the image matrix is symmetrically supplemented with elements equal to zero so that the resulting size is equal to the $(N+N_a)\times(N+N_a)$

Table 5.2 Aperture Size 5×5

w_3	w_3	w_3	w_3	w_3
w_3	w_2	w_2	w_2	w_3
w_3	w_2	w_1	w_2	w_3
w_3	w_2	w_2	w_2	w_3
w_3	w_3	w_3	w_3	w_3

elements, where N is the size of the image in pixels. Note that for $w_{v\xi} = const$, the averaging algorithm takes place. In addition, when filtering a quantized image, it is necessary to introduce a set of thresholds, the power of which is equal to the number of brightness gradations (in our case, these are 16 values). The element $\tilde{a}_{ij}$ is assigned a value corresponding to the maximum threshold that exceeds the sum on the right side of Eq. (5.1).

The filtering quality increases with increasing aperture size $N_a \times N_a$. However, at the same time, the time spent on calculations increases proportionally N_a^2. In practice, it is enough to have $N_a = 5$, which gives good quality with a short filtration time. The aperture of this size corresponds to Table 5.2.

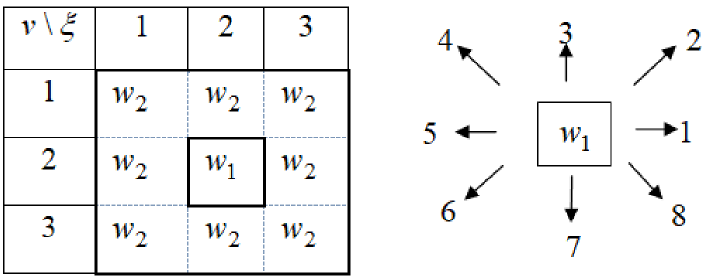
Elements $w_{v\xi}$ are usually selected based on a normal uncorrelated bivariate distribution, the maximum of which coincides with the center of the aperture. The smaller the standard deviation of the distribution σ_a, the more weight is given to the central element if the next normalization condition is met:

$$\sum_{v=-N_a/2}^{N_a/2} \sum_{\xi=-N_a/2}^{N_a/2} a_{i+v,j+\xi} w_{v\xi} = 1. \tag{5.2}$$

Practice shows that narrow apertures ($\sigma_a < 1$) are more effective at a low noise level, while wide ($\sigma_a \geq 1$) – at a large one. In this case, the values range from 0.4 to 0.02.

Anisotropic filtering works successfully in many situations; however, in the above application, the structure of the image is not taken into account, while such information can be useful in solving a specific filtering problem. In particular, in EBW it is a priori known that the joint has sufficiently smooth edges and, therefore, its image should not have large gaps and thinning. Based on this information, you can improve the anisotropic filtering method using the idea of constructing a composite filter. This filter calculates the gradient of the original image.

One could first simply apply a linear directional filter to the image and then perform



$v \setminus \xi$	1	2	3
1	w_2	w_2	w_2
2	w_2	w_1	w_2
3	w_2	w_2	w_2

Fig. 5.13. Coordinates and directions of aperture.

anisotropic filtering. In this case, we will have an obvious loss of performance. It will be more efficient to use a linear directional transform together with the aperture at a single filtering step. In this case, a narrow aperture with $\sigma_a > 0.3$ can be chosen, since when $\sigma_a \leq 0.3$ the weight of the central element w_1 is 1, the other elements with $w_2 = w_3 = 0$ are not filtered. It can be accepted $\sigma_a = 0.5$, since for it there are already known from practice values $w_1 = 0.44$, $w_2 = 0.07$, $w_3 = 0.00$. As you can see, in this case, the aperture degenerates into a matrix 3×3, and eight predefined directions from the central element can be set for it, which can be investigated when calculating the image gradient. Taking into account these directions, each pixel of the image corresponding to the aperture will, in addition to coordinates, be characterized by an angle of location relative to the central element of the aperture:

$$a_{i+v,j+\xi} \Rightarrow a(i,j,\alpha), \tag{5.3}$$

where $\alpha = 1,..,7$ is the direction of the aperture element specified by coordinates. The coordinates and directions will be counted according to Fig. 5.13.

The gradient of the image point coinciding with the center of the aperture is defined as a vector directed towards the maximum decrease in brightness, assuming that the joint is always represented by a dark stripe. The gradient value is calculated by the formula

$$g(x,y,\beta) = \max_{\alpha}\left[\frac{\sum_{k=1}^{L} B_\alpha(x_k,y_k)}{L}\right], \tag{5.4}$$

where $B_\alpha(x_k, y_k)$ – the brightness value of the k-th pixel, counting from the central element of the aperture in the direction α;

L – the specified length of the direction vector;

β – the direction of the gradient.

Based on Eqs. (4.3) and (4.4), the final formula can be obtained for calculating the brightness levels of image pixels during filtering

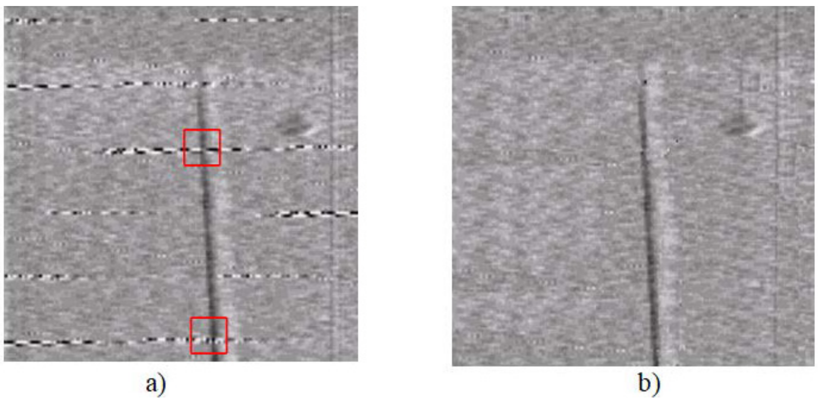


Fig. 5.14. View of the joint image before (a) and after (b) filtration.

$$\tilde{a}_{ij} = \Lambda\left[\sum_{v=1}^{N_a}\sum_{\xi=1}^{N_a} a_{i+v,j+\xi} w_{v\xi} - \eta\right], \tag{5.5}$$

where $a_{i+v,j+\xi} = a(i,j,\alpha)$ when $\alpha \neq \beta$ and $a_{i+v,j+\xi} = g(i,j,\beta)$ when $\alpha = \beta$; Λ – the threshold function.

Figure 5.14 shows the images of the joint before (Fig. 5.14 a) and after (Fig. 5.14 b) the application of the anisotropic filtering method together with a directional filter.

Thus, using the applied filtering model, it is possible to restore the areas of the joint image distorted by interference (indicated by red squares) and clear the surface texture of the sample.

5.3.4. Contrasting Models

Many manufacturing tasks require precise definition of the boundaries of objects in order to isolate their shape or find the middle pixels. The anisotropic filter, although it does not introduce additional blurring of the edges, does not solve the contrast problem. Figure 5.15 shows an example of the various contrast options that may arise in practice.

To increase the contrast, special algorithms are used that perform image reconstruction functions. These algorithms usually implement the idea of linear

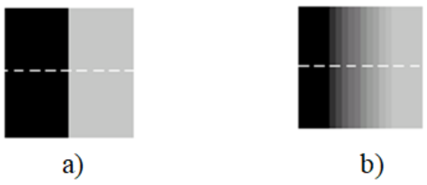


Fig. 5.15. Examples of image contrast: a) perfect border; b) blurred border.

filtering, when the original image $f(x, y)$ is converted to an image $g(x, y)$ as a result of applying a linear transformation

$$g(x,y) = \sum_i \sum_j h(i,j)\, f(x+i, y+j). \tag{5.6}$$

Known methods for determining boundaries mainly used as linear transformations the convolution of image points with template. For example, for convolutions along the x-axis and y-axis, respectively, the Sobel method (Parker, 1999) uses templates of the form that is shown in Fig. 5.16.

Eight templates with size 3×3 are used in the method of contrasting with directional gradient masks (the number of templates corresponds to the number of main directions: north, northeast, etc.). The course name indicates the direction of the slope of the brightness difference at which the template gives the maximum response. It should be noted that in all methods using templates, the latter have zero total weight; therefore, in image regions with constant brightness, they give zero response. The disadvantage of the listed methods can be considered that they all use static templates that do not take into account the real values of the brightness of the image pixels and their relative position.

With the existing information regarding the structure of the analyzed image, the construction of templates can be improved by making this process dynamic. Unlike simple filtering, we need to reorganize the original image to highlight the difference at the levels of the seam image and the background area. At the same time, we will adhere to the following assumptions:

1. The joint is always displayed as a dark bar and has the lowest brightness level.

2. The brightness of the joint points should not change, which means, taking into account clause 1, there should be no correction towards decreasing brightness for any points.

3. The diagonal of the template is oriented in the direction of the gradient of the image point corresponding to the center of the template.

$$h_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \; ; \qquad h_y = \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Fig. 5.16. The forms of image templates.

Under the assumptions made, the algorithm for calculating the template for all image points will include the following steps:

1. Select an element of the image in size 3×3 (each side of the original image should be initially increased by 1 pixel).

2. Align the template with the image point a_0 and set the central element of the template to 1.

3. Calculate the gradient of the point a_0.

4. Divide the image element by the diagonal into two disjoint subsets A_1 and A_2; set all elements of the template located on the diagonal to 1.

5. Calculate the average brightness of the pixels for each formed subset $A_i = \frac{\sum_{j=1}^{3} a_{ij}}{3}, i = 1,2$.

6. Compare the brightness of the subsets with the brightness of the average pixel: if $(a_0 \geq A_i) \wedge (a_0 < A_j), i \neq j$, then all elements of the template that cover $a \in A_i$, assign the value 1, but the elements $a \in A_j$ – the value 2, otherwise all elements of the template get the value 1.

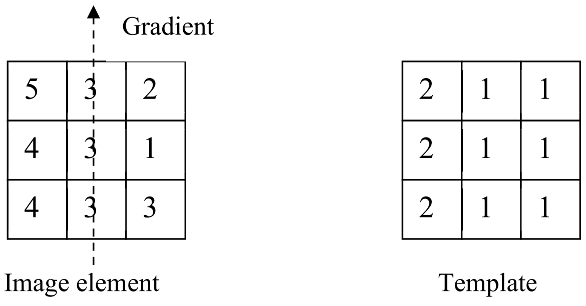


Fig. 5.17. Dynamically generated template.

7. Apply transformation to the image element 5.6.

An example of a template generated by this algorithm is shown in Fig. 5.17.

Figure 5.18 shows a view of the joint image obtained before and after applying the contrasting method described above.

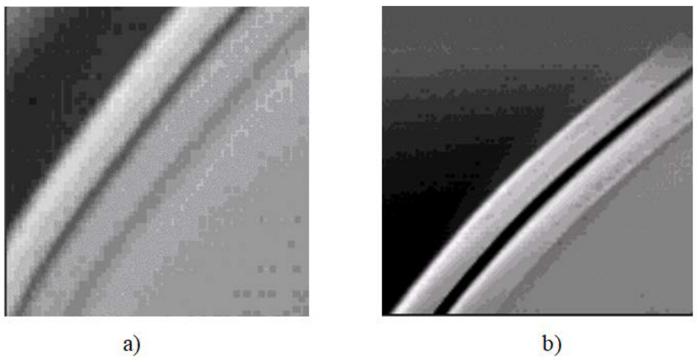


Fig. 5.18. View of the joint image before (a) and after (b) contrasting.

5.3.5. Segmentation Models

Segmentation allows you to select homogeneous areas of images, such as a joint. The main methods traditionally used in segmentation are thresholding, boundary detection, and augmentation of areas.

The first two of the listed methods are based on determining the difference in the brightness values of pixels, while the segmentation method by augmentation involves finding groups of pixels with similar brightness values. In its simplest form, it also involves selecting a pixel and examining adjacent pixels to check the proximity of brightness values. If the brightness values are close, then the corresponding pixels are included in the same group to form a region. In this case, the area is formed by splicing individual pixels.

However, it is more efficient to use in segmentation not separate points, but entire areas. There is, for example, a group of methods related to digital morphology (Latecki and Gross, 1995). The concept of digital morphology is that pixels are assembled into groups that have a given structure. These groups of pixels are called shapes, or building blocks. The main morphological operations that are applied to already selected pixels are deleting and adding shapes. As a rule, binary templates are used, consisting of two types of pixels: white and black, which are designated 1 and 0, respectively.

Formally, the operation of adding a template is defined as

$$S + S_1 = \{c | c = s + s_1, s \in S, s_1 \in S_1\}, \tag{5.7}$$

where S – the image to be segmented;

S_1 – a figure template.

Sketchily, the operation of adding shapes looks as shown in Fig. 5.19.

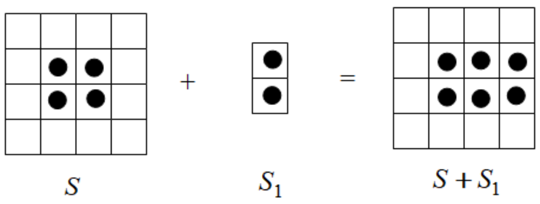


Fig. 5.19. Scheme for adding a template.

For gradient images, it is possible to propose, instead of binary templates, the calculation of the average brightness value for all pixels of the template. The same condition can serve as a constructive rule for constructing the template itself: those pixels that do not worsen the uniformity (average brightness) of the area will be enrolled in the template area. As a result, templates of a certain shape will be generated dynamically in the course of segmentation.

By combining the augmentation method with digital morphology, it is possible to obtain a more efficient algorithm for selecting areas to be recognized, since whole groups of pixels will be added at once. This algorithm will include the following steps:

1. On the original image, determine the first element belonging to the seam area (perform training) with brightness a_0.

2. Determine the width of the joint W (the diameter of the smallest circle, homogeneous to the first element).

3. Determine the direction of the joint by circular scanning the image with a vector of a given length. The direction of the joint will correspond to the direction of the vector of image pixels, which has the lowest brightness (darkest); assign $k = 0$.

4. Having chosen as the base a_0, construct a rectangular area R with a width equal to W, and a length equal to $W + k$, located perpendicular to the calculated direction of the joint.

5. Check the homogeneity of the resulting area. If the average brightness of the area $B_k = \frac{\sum_{p \in R} a_p}{|R|}$ has not improved, i.e., $(B_k - a_0) < \delta$, then assign $k = k + 1$ and go to item 4, otherwise go to item 6.

6. If $k = 0$, then go to item 7, otherwise assign $a_0 = a_{k-1}$, where a_{k-1} is the median element of the side of the rectangle opposite to the base, which was obtained in the previous step; go to Step 3.

7. Finish selection of the joint area in the image frame.

Figure 5.20 shows a view of the segmentation model built using the augmentation method with digital morphology for joint recognition.

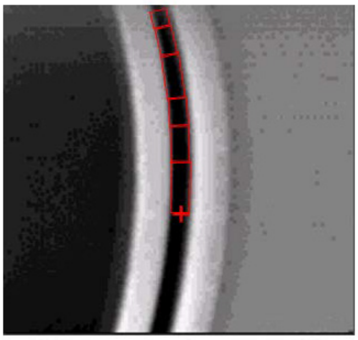


Fig. 5.20. Joint image segmentation model.

5.3.6. Thinning Models

Skeleton models are used to restore the geometric properties of objects to be recognized, in particular the path of the joint.

At this stage, the original regions obtained as a result of segmentation are converted into lines one pixel wide. This approach is most widespread in the problems of character recognition, but it can be successfully applied in technical vision systems.

The basic technique in skeletal models is thinning. It is an iterative procedure that eventually extracts the skeleton of an object. At each iteration, a border pixel that has at least one pixel adjacent to the background is removed if it does not violate the object topology. There are two main requirements that a thinning model must satisfy:

- if the object is connected, then the result must also be connected;
- the line obtained after thinning should pass in the middle of the area subject to thinning.

The thinning algorithms based on the use of templates have become most popular in recent years. The principle of their work is to iteratively align the template with the image and remove the middle pixels. The original image is gradually thinned until the last layer of boundary pixels is reached.

An example of this kind of algorithms is the Stantiford's algorithm (Parker, 1999). It uses a set of four templates of the size 3 × 3 shown in Fig. 5.21.

Note that when checking images, only template pixels marked with circles are considered. The rest of the pixels are not counted. In addition, the Stantiford's algorithm considers binary images whose pixels are marked with either 0 (white) or 1 (black). Application of templates $T_1 - T_4$ is based on two definitions.

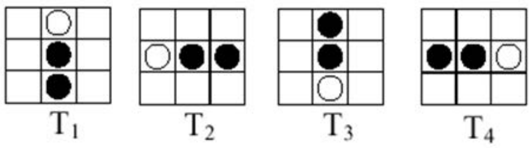


Fig. 5.21. Patterns for removing pixels by the Stantiford's algorithm.

Definition 5.1. An endpoint is a black pixel that has only one bordering black pixel out of eight adjacent pixels.

Definition 5.2. The indicator of the connectivity of an image element covered by a pattern is the number of connected components that are formed when the central pixel is removed.

In Fig. 5.22, image elements with different connectivity indicators are given as an example: C_n=0, b) C_n=1, c) C_n=2, d) C_n=3, e) C_n=4.

Within the template, pixels are numbered counterclockwise, starting from the pixel to the right of the center. The center pixel itself is numbered 0.

Let us apply the Stantiford's algorithm for thinning the image of the joint, slightly modifying it. In particular, we will calculate the connectivity index of the central pixel by the formula:

$$C_n = \sum_{i=1}^{8}(B_k + B_{k+1})(1 - B_k). \tag{5.8}$$

Further, let us perform the following steps:

1. Convert the image obtained after segmentation to a binary form, assigning all black pixels of the selected area to 1, and all other pixels to 0.

2. Find an element of the image A that satisfies the pattern T_1, sequentially passing through the image with this pattern along the top border of the image from left to right and from top to bottom.

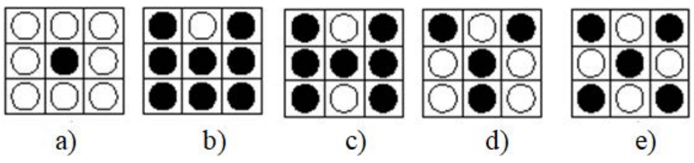


Fig. 5.22. Examples of picture elements with different connectivity indicators.

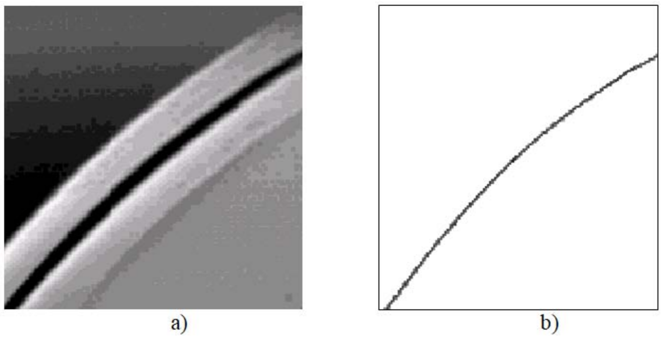


Fig. 5.23. View of the joint image before (a) and after (b) the thinning algorithm.

3. If the central pixel of the image element A is not the end point and has a connectivity index equal to 1, then this pixel is marked for deletion.

4. Repeat Steps 2 and 3 for all image points that match the pattern T_1.

5. Delete all points marked for deletion.

6. Repeat Steps 2–4 for all templates T_2–T_4. The pattern T_2 passes, starting from the left edge of the image, sequentially moving from the left edge of the image from bottom to top and from left to right. The pattern T_3 passes from the bottom edge of the image, moving sequentially from right to left and from bottom to top. The pattern T_4 is traversed starting from the right border of the image, moving sequentially from top to bottom and from right to left.

7. Repeat Steps 2–6 until at least one pixel is removed.

Figure 5.23 shows the result of this thinning algorithm.

Note that always after applying the thinning model, an image of the selection is one pixel wide.

5.3.7. Trajectory Restoration Models

Analysis of the majority of applied problems of image processing shows that most often a distinctive feature of an object is its shape.

The set of points obtained as a result of the primary image processing must be converted into a mathematical description of the object shape. The requirements for such a description depend on the recovery goals. In some cases, it is necessary to obtain a description of the outline of the selected area in order to then use it for classifying objects. In the problem we are considering, the trajectory of the joint of the welded products is subject to restoration, which should set the program for moving the electron beam.

There are several problems associated with the restoration of trajectories. Finding a trajectory passing through a given set of points is an interpolation problem. If it is necessary to draw a trajectory near a given set of points, then the approximation problem is solved. Finally, when it is necessary to reproduce the curve according to the obtained mathematical descriptions, the problem of computer graphics is solved.

From a mathematical point of view, interpolation problems are easier to solve. However, in many cases, an approximation is more appropriate. First, data distortion is often caused by the presence of noise, which should not be mistaken for actual deviations. Second, the tool movement program is usually specified as a set of standard CNC commands, including linear and circular interpolations; spline interpolations, like Bezier polynomials, do not always reproduce the trajectory correctly, since they allow for unwanted deviations in the spacing between points. The latter problem can be partially overcome by localizing and interactively defining glue points. However, for automatic systems of the HIL class, this solution is no longer trivial.

With the existing tolerances for accuracy, the trajectory can be successfully reconstructed by an automatic approximation. The choice of reference points in this case is replaced by measures of proximity, precisely determined mathematically. Note that the approximation by splines with variable knots generally cannot be solved by strictly mathematical means.

Let us consider an automatic approximation algorithm that provides trajectory reconstruction from the resulting image. The limitations of this algorithm are as follows:

- a sequence of points $P = ((x_1, y_1), (x_2, y_2), \dots, (x_N, y_N))$ obtained as a result of primary image processing at the stages from sampling to refinement is considered the initial data;

- only trajectories defined on the plane are considered;

- approximation is carried out using segments of straight lines and arcs of circles starting at fixed points, which are the gluing points of the local areas (trajectory segments) selected during the operation of the algorithm;

- the Least Squares Method (LSM) is used to calculate the parameters of the approximating functions.

The automatic approximation algorithm includes the following steps:

1. Select the path segment that includes the first three points in the sequence P. Assigned to $k = 1$ – a starting point number, $n = k + 1$ – a segment last point number.

2. $n = n + 1$, $LA = false$ – a sign of linear interpolation, $CA = false$ – a sign of

circular interpolation. If $n = N$, then go to Step 6. The possibility of approximating the selected trajectory segment consisting of $(n - k + 1)$ points with a straight line segment is checked. Straight line $\overline{y} = ax + b$ coefficients are calculated by the formulas:

$$a = \frac{s_{xy} - s_x \cdot s_y}{s_{x^2} - s_x \cdot s_x},\ b = \frac{s_{x^2} s_y - s_x s_{xy}}{s_{x^2} - s_x \cdot s_x}, \tag{5.9}$$

where $s_x = \sum_{i=k}^{n} x_i$, $s_{x^2} = \sum_{i=k}^{n} x^2{}_i$, $s_y = \sum_{i=k}^{n} y_i$, $s_{xy} = \sum_{i=k}^{n} x_i \cdot y_i$.

3. The error of linear approximation is calculated by formula:

$$E = \max_{i \in \{1,2,\dots,n\}} d_i^2, \tag{5.10}$$

where d_i – the distance from a point $(x_i, y_i), i = \overline{k, n}$ to the interpolating line.

If $E \le \delta_l$, where δ_l is the specified accuracy of the linear approximation, then the initial point of the approximating straight line $\overline{y}(x) = y(x)$ is fixed, the error of the corrected linear approximation is calculated by Eq. (4.10) and, if $E \le \delta_l$, then $LA = true$ and the transition is to item 2.

4. $n = n + 1$. If $n = N$, then go to Step 6. The possibility of approximating the selected trajectory segment consisting of $(n - k + 1)$ points by an arc of a circle $(x - a)^2 + (\overline{y} - b)^2 = R^2$ is checked. The unknown coefficients a, b, and c, which are used to calculate the radius of the circle $R = \sqrt{c + a^2 + b^2}$, are determined using the system of equations

$$\begin{cases} s_{y^2} \cdot b + s_{xy} \cdot a + s_y \cdot c = s_{y^3} + s_{x^2 y} \\ s_{xy} \cdot b + s_{x^2} \cdot a + s_x \cdot c = s_{xy^2} + s_{x^3}, \\ s_y \cdot b + s_x \cdot a + n \cdot c = s_{y^2} + s_{x^2} \end{cases} \tag{5.11}$$

where $s_x = \sum_{i=k}^{n} x_i$, $s_{x^2} = \sum_{i=k}^{n} x_i^2$, $s_{x^3} = \sum_{i=k}^{n} x_i^3$, $s_y = \sum_{i=k}^{n} y_i$, $s_{xy} = \sum_{i=k}^{n} x_i \cdot y_i$, $s_{x^2 y} = \sum_{i=k}^{n} x_i{}^2 \cdot y_i$, $s_{xy^2} = \sum_{i=k}^{n} x_i \cdot y_i^2$, $s_{y^3} = \sum_{i=k}^{n} y_i^3$, $s_{y^2} = \sum_{i=k}^{n} y_i^2$.

5. The maximum error of circular interpolation is calculated:

$$E = \max \left[\max_{k \le i \le n} d_i^2,\ \max_{1 \le j \le (n-k)} c_j^2 \right], \tag{5.12}$$

where d_i – the distance from the point $(x_i, y_i), i = \overline{k, n}$ to the interpolating circle;

c_j – the distance from the center of a line segment connecting two adjacent points of the segment to the approximating circle.

If $E \leq \delta_c$, where δ_c is the specified accuracy of circular interpolation, then the starting point of the approximating circle $\overline{y}(x) = y(x)$ is fixed, the error of the corrected circular approximation is calculated by Eq. (5.12) and, if $E \leq \delta_c$, then $CA = true$ the transition is to Step 4.

6. If $CA = true$, then the circular interpolation is set for the segment $(x_k, y_k), \ldots, (x_n, \overline{y}_n)$. If $LA = true$, the linear interpolation is set for this segment, otherwise the linear interpolation obtained for the segment at $n = n - 1$. If $CA = false$ and $LA = false$, the points of the segment are connected by straight line segments, forming a piecewise linear approximation.

7. A new segment $k = n, n = k + 1$ is selected. If $n = N$, then go to item 8, otherwise go to item 2.

8. Complete the algorithm.

5.4. Summary

The main purpose of the recovery models is to provide the CCS with the missing information that cannot be obtained from the sensors through the input data transmission channels. There are two main options for using recovery models (operator in the control loop and hardware in the control loop), which define two classes of computer recovery models: virtual reality display models and image recognition models.

Display models perform functions associated with displaying virtual three-dimensional representations of the CO position in the surrounding space with the ability to change the viewing angle and image scale. Taking into account the scanning of the CO state according to the data coming from the sensors in real time, it is possible for the operator to carry out situational control of the object, based on the provided virtual display.

The technique for constructing virtual representations is based on the use of a display subsystem that includes a frame generator, an image manager and a display scheme manager as structural components. The used images of objects can be created either on the basis of a set of typical elements, or formed using well-known graphic systems.

Models of image recognition imply the sequential application of methods of obtaining images, filtering the obtained images, contrasting and segmentation of images to select target areas, as well as their refinement to restore the resulting trajectories in the form of graphic models.

To speed up the processes of obtaining and processing images, as well as to ensure high quality recognition, a set of methods and algorithms has been used. It includes:

an improved method of anisotropic filtering, taking into account the structure of images, an improved contrasting method using dynamically created templates, and an improved segmentation method by building up with simultaneous application of digital morphology. These methods can be applied not only to the control loop of EBW machines, but also to other control problems using recovery models.

Chapter 6. Software and Hardware Tools for MOC

In the process of applying model-based control, it is necessary to solve two practical problems: the creation of CA models for their preliminary debugging and their integration into the control loop of technological process. The main difficulties arise in the development of technology that supports the use of implementation and predictive models, since recovery models are based on standard instrumental and operational environments.

6.1. Technology for Creation of Implementation Models

6.1.1. E-net Modeling System

The E-nets Modeling System (EMS) (Kazymyr et al., 2011) is a tool for creating software implementation models which is available at http://195.69.76.84:8080/ems-ui-vaadin-0.1-SNAPSHOT/. The main difference of EMS is that it combines the power of simulation with the ease of creating software models of CA. The JAVA language has been chosen as the basic programming language, which is not only an object-oriented language corresponding to the chosen formal approach, but also provides cross-platform program execution.

E-networks in combination with an aggregate approach are used as the formal apparatus involved in EMS. Thus, this system of simulation modeling fully meets the needs of building models for the implementation of CA and can be successfully used as a software environment for their development and preliminary research.

EMS functionality is focused on supporting the full life cycle of simulation models, including the development of conceptual, formalized and programmatic models. This modeling system allows you to create new models for the implementation of CA; modify existing models and perform statistical experiments with models at the design stage of the CA. In addition, EMS contains a graphical web-interface with specification language that enables models to be built by a user with no prior programming background.

The EMS architecture is shown in Fig. 6.1.

The main part of this architecture is a model. An aggregate is used to store the structure of the model. It can contain base elements of CEN, such as transitions, positions, variables including temporal formulas, as well as nested aggregates inside.

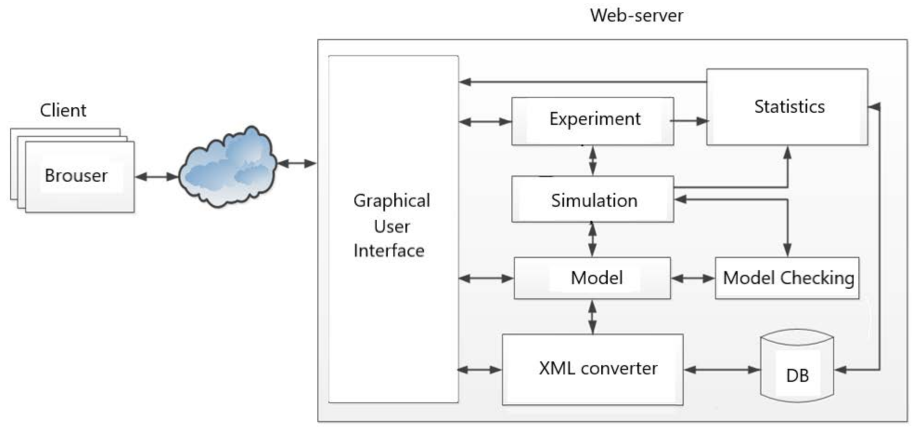


Fig. 6.1. EMS architecture.

The model contains a reference only to the root aggregate; the objects of the root aggregate are taken into account when the model is run. Thus, in essence, a model is also an aggregate and can be used precisely as an aggregate in other structurally more complex models. The main difference between the model and the aggregate is that the latter cannot be launched to carry out experiments and collect statistical data.

Variables defined by name, type, and value can be defined at both the root and nested aggregate levels. The scope of a variable is determined by the aggregate in which it is created and the child aggregates. The token is determined by the position at which it is set, which allows creating several tokens in the aggregate with a different set of attributes. Each attribute, like a variable, is characterized by a name, type, and value.

The main window of the EMS system (Fig. 6.2) is visually divided into several parts.

The elements of the main window of the system are the following:

- Model – a graphic editor of models;
- Components – a set of E-network components for building models;
- Aggregate – a library of aggregates (templates for creating models). This window displays previously created aggregates that the user can reuse in other models;
- Model components – display the internal structure of the model in the form of a hierarchy tree, which contains all the elements of the model at all its levels;
- Console – a system console, which displays errors and other messages related

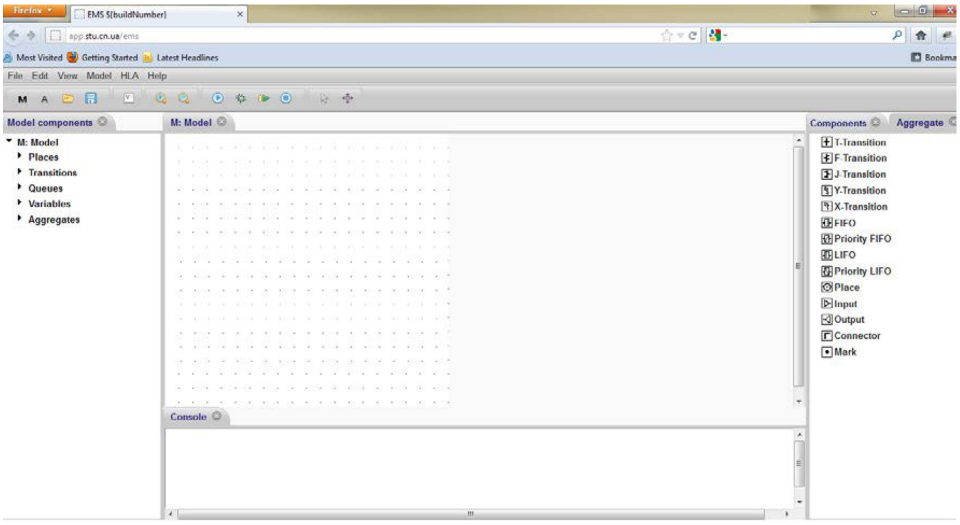


Fig. 6.2. General view of the main window of the EMS system.

to the modeling process. For example, when the system starts, the message "System started" is displayed in the console.

- Main menu of the system for setting parameters and performing simulation;
- Quick launch toolbar.

By default, all panels of the system are active; however, if necessary, they can be disabled/enabled using the main menu command View ⍰ Model components ⍰ Panel name.

The creation of models carried out in graphical mode on the basis of the developed components includes the following stages:

1. At the first stage of creating models, it is necessary to determine its conceptual scheme, i.e., to determine its structure. Since in the proposed approach the model is a set of pyramidal growing aggregates, it is, first of all, necessary to create the aggregates of the model. There are two ways to call the unit creation window (Fig. 6.3): using the File→New→Aggregate main menu item or the "A" button on the quick launch panel.

In the window that appears, the user enters the name of the unit and chooses the method of its creation: from a "blank slate" or on the basis of previously created units, in the latter case, the user selects the unit from the list and presses the "Choose" button.

2. At the second stage, it is necessary to create (edit) the internal structure of

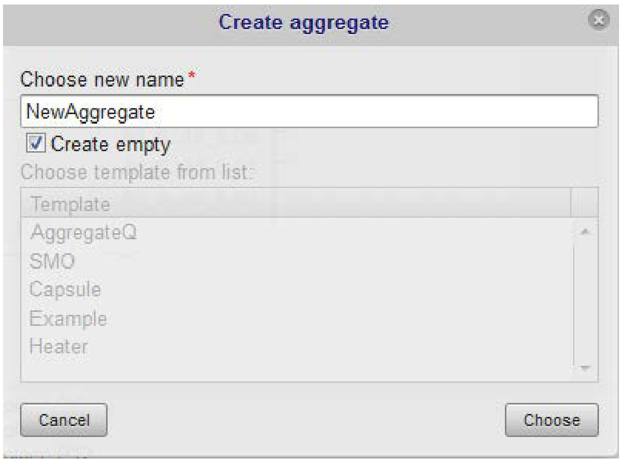


Fig. 6.3. Model aggregate creation window.

the unit. Adding the components of the E-net to the unit is performed by sequential clicking of the mouse first on the corresponding component in the Components panel, and then in the model editor. As a result of these actions, a new component is added to the unit, its image appears in the editor, and the name is entered into the list of components of this unit in the Model Components panel. The possibility of moving objects is provided, which makes it possible to represent the internal structure of the unit in the most convenient and ergonomic form.

After determining the required components of the unit, it is necessary to connect them in order to define a network that follows the logic of the conceptual model. Components are connected using the “CONNECT” element, which is also selected on

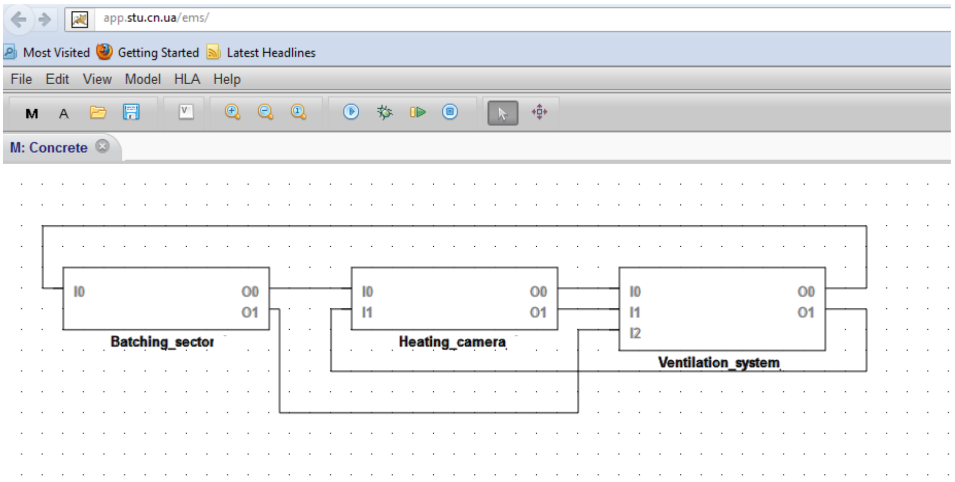


Fig. 6.4. Formalized aggregate model.

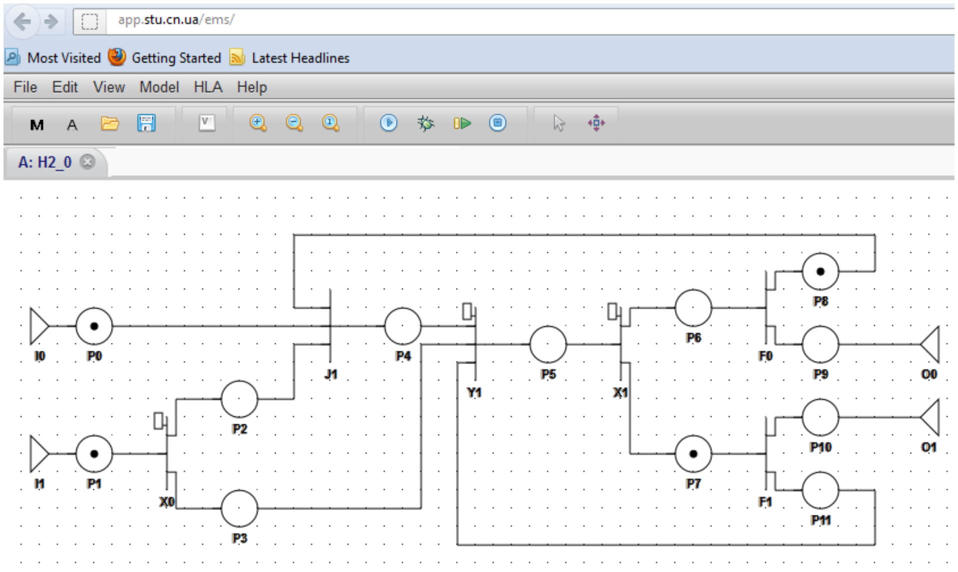


Fig. 6.5. CEN model of aggregate.

the Components panel. Note that connecting a transition with a transition, as well as a position with an E-net position, is impossible. However, it is possible to connect an aggregate with an aggregate, which is dictated by the need to create hierarchical

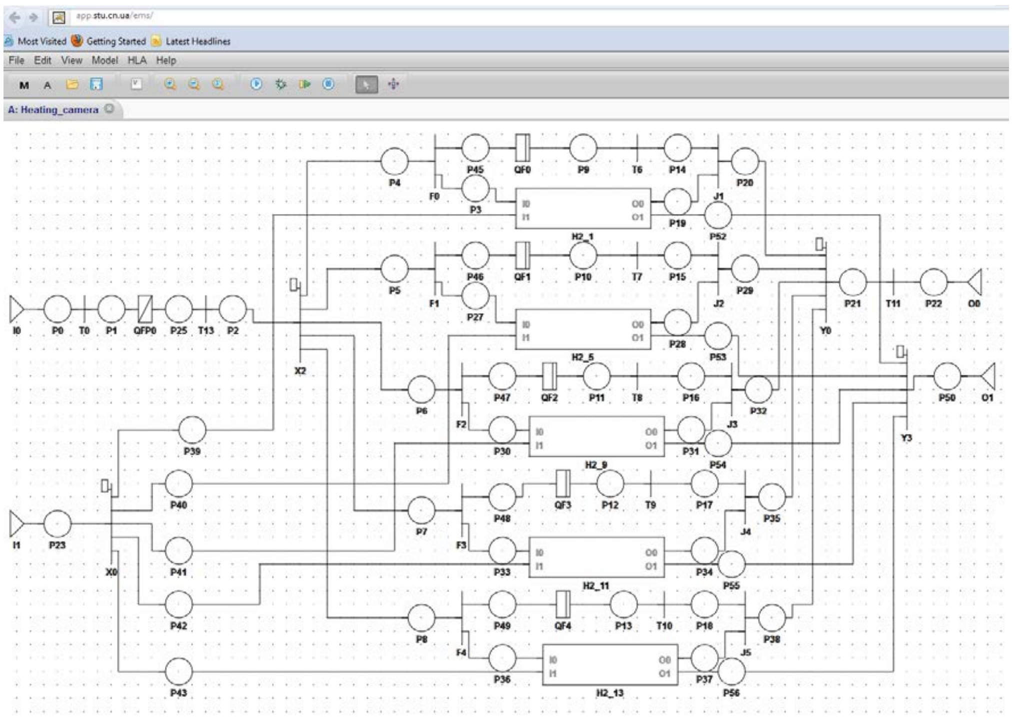
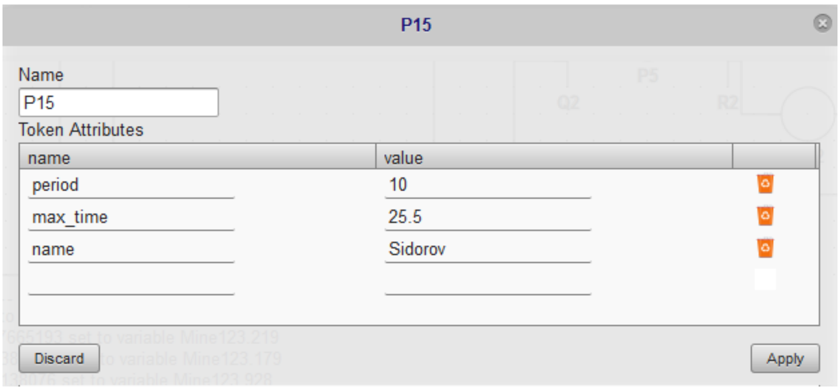


Fig. 6.6. CEN model with nested aggregates.


P15

Name

P15

Token Attributes

name	value
period	10
max_time	25.5
name	Sidorov

Fig. 6.7. Token Attribute Editor window.

aggregates; in addition, the aggregate can be directly connected to a position and play the role of a transition in the network. It is possible to construct broken communication lines and move them in order to improve the convenience of building a network.

An example of a formalized model consisting of aggregates is shown in Fig. 6.4.

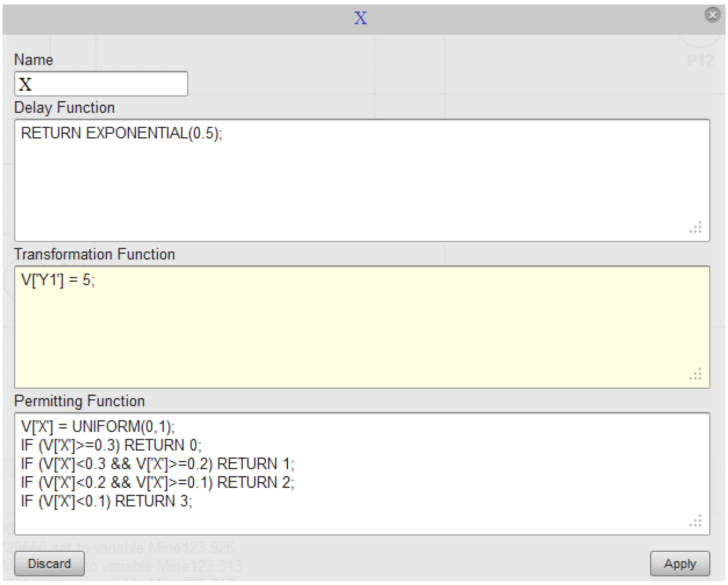
Another example of a formalized model of an aggregate in the form of a CEN is shown in Fig. 6.5. One more example (Fig. 6.6) shows the possibility of building hierarchical models that include nested aggregates that are used as CEN transitions.

3. The final stage of building a model aggregate is the task of marking the network, determining the variables of the network, as well as setting functions on the transitions of the network. For marking, it is necessary to select the "MARK" component and click on the corresponding position of the E-net – the token will be displayed in the middle of the position. Double-clicking on the token calls the attribute editing window (Fig. 6.7).

A similar window is called for editing network variables, using the main menu item of the system or the "V" button on the quick launch panel. The scope of a variable is limited to the aggregate, in which it is created and is available to nested aggregates. The top-level unit does not have access to the specified variable.

To set the values of the functions, it is necessary to select the required transition; double clicking on it calls the function editor window (Fig. 6.8).

The transition functions are defined in the special E-nets language (EL). By default, the delay function is zero, the permitting function also returns zero by default, which corresponds to the first input / output position of the transition, and the transformation function does not perform specific actions. The permitting functions are defined only for "X" and "Y" types of transition.



```
RETURN EXPONENTIAL(0.5);
```

```
V['Y1'] = 5;
```

```
V['X'] = UNIFORM(0,1);
IF (V['X']>=0.3) RETURN 0;
IF (V['X']<0.3 && V['X']>=0.2) RETURN 1;
IF (V['X']<0.2 && V['X']>=0.1) RETURN 2;
IF (V['X']<0.1) RETURN 3;
```

Fig. 6.8. Transition function editor.

6.1.2. E-net Language Definition

E-net Language is a high-level interpreted language that is used in EMS to define the CEN transition functions by the user. The interpreter converts EL constructs into Java sentences.

The EL supports the following data types:

1) INTEGER – integers in the range from -263 to 263 -1 (0, 128, -144);

2) REAL – real numbers, the exponent can take values from -2147483648 to 2147483647 (-0.05, 32E + 2, 2E-3);

3) BOOL – Boolean data type (TRUE, FALSE);

4) STRING – text data type, the number of characters in a line is not limited ("LINE", "LINE").

Language identifiers are used to specify variable names and consist of any sequence of letters, numbers, or underscore. A digit cannot be the first character. The names must not match the values of the keywords.

Variable declarations begin with the VAR keyword followed by the variable name. Repeated announcements are prohibited. At the moment of declaration, the variable can be initialized by a certain value, otherwise it will take an undefined value. For example:

VAR INIT = 4.5;

VAR UNINIT.

Using EL, the user has the opportunity to organize a full interaction with the model. To refer to aggregates, it is necessary to use the key character A, to the root aggregate – ROOT, to tokens – T, to positions – P, to variables – V.

To access an aggregate variable, you must refer to the required aggregate using the key character A with the aggregate identifier or key word ROOT to access the root aggregate and specify the variable name. For example:

VAR X = V [‘r01’];

ROOT.V [‘t02’] = TRUE;

A [8] .V [‘as’] = “STRING”.

To access the attributes of a token, you must refer to a certain position in the aggregate. If no aggregate reference is specified, the current aggregate is accessed. For example:

A [2]. P [11] .T [1] = 12.5;

P [2] .T [2’] = TRUE.

To check the presence of a token in position, the T key character is used. The type of the calculated value is BOOL:

VAR isPlace1Marked = P [‘place1’]. T.

Mathematical operations are defined for the INTEGER and REAL types. In the case of using an operand of a different type, the result of the operation will be an undefined value (unless the operator used for this type, for example, ‘+, can be used to concatenate strings). The resulting type is INTEGER or REAL. Allowed operators: ‘+’ – addition; ‘-’ – subtraction; ‘*’ – multiplication; ‘^’ – exponentiation; ‘-’ – sign change; ‘!’ – sign change for BOOL variables; ‘/’ – division. Division by zero results in an undefined value.

Comparison operations are defined for all data types. Allowed operators: >,> =, <, <=, ==,! =.

Logical operations are defined for the BOOL type. If an operand of a different type is used, the result of the operation will be an undefined value. Allowed operators: '&&' – logical AND; '||' – logical OR; '!' – negation; '-' – negation.

The concatenation operation is defined for STRING data type by using the '+' operator.

Conditional operators are used to skip or execute some statements depending on the computed values of the given constructs. Operators can be grouped into blocks using brackets. Defining constructs are of two types:

1) Explicitly specified logical expression. Example:

VAR X = 0; IF ('a' == 'b') X = -92; ELSE X = 92; RETURN X; // = -92

2) Check for the existence of a value. Example:

IF ('aa') RETURN 2; ELSE RETURN -2; // = 2

Operator RETURN returns the specified value, and operator TIME returns the current simulation time, type INTEGER value.

The following functions are defined for values of INTEGER and REAL types:

- SIN (X) – sine, result – REAL;
- COS (X) – cosine, result – REAL;
- TAN (X) – tangent, result – REAL;
- COT (X) – cotangent, result – REAL;
- ATAN (X) – arctangent, result – REAL;
- LN (X) – a natural logarithm, result – REAL. The function is defined for argument values > 0;
- SQRT (X) – a square root, the resulting type corresponds to the type of the argument, the function is defined with the argument values > = 0.

The following functions are defined for any value of type INTEGER and REAL:

- ABS (X) – a module, the absolute value of the argument. The resulting type matches the type of the argument.

- SIGN (X) - 1 for X> 0, 0 for X = 0, -1 for X <0
- ENTIER (X) is the largest integer not exceeding X. The computed value type is INTEGER.

Functions for obtaining random variables use the internal congruent generator of pseudo-random numbers, which generates sequences of random numbers. Each time before starting the interpretation, the generator is initialized with the current time value in ms. These functions are the following:

- SEED (X) – initializes the random number generator with the INTEGER, for example, SEED (60,000);
- POISSON (X) – generates integers distributed according to Poisson's law with parameter X. Type of parameter X is INTEGER or REAL, the returned value is INTEGER; the function is defined for argument values > 0, for example, POISSON (2.5);
- UNIFORM (A, B) – generates numbers evenly distributed over the interval [A, B] (B > A); parameters A and B are INTEGER or REAL. The returned value type is INTEGER. The function is defined for values B > A, for example, UNIFORM (2, 10.9);
- EXPONENTIAL (X) – generates exponential distributed numbers with the X parameter of INTEGER or REAL types, the returned value type is REAL, for example, EXPONENTIAL (10.9);
- NORMAL (A, B) – generates numbers distributed according to the normal law with expectation A and standard deviation B. Type of parameters A, B – INTEGER or REAL, the returned value type is REAL; the function is defined for values B > 0, for example, NORMAL (2, 10.9);
- BINOMIAL (A, B) – generates numbers distributed according to the binomial law with the number of trials A and the probability of success B; parameter A is INTEGER, range from 0 to 231-1 inclusive, B – INTEGER or REAL, range from 0 to 1; the type of the returned value is INTEGER; for example, BINOMIAL (2, 0.4).

In EMS, the followed operators of temporal logic DCTL are used:

- AG – the condition must be fulfilled in all ways, in all states;
- AF – the condition must be fulfilled on all paths, at least in one state;
- EG – the condition must be fulfilled in at least one way in all states;

- EF – the condition must be fulfilled in at least one path in at least one state.

The formula that will be checked with these operators determines the condition that must be met and will be checked for each state of the model. The formula can consist of several expressions that are combined by a logical operator during determination of experiment conditions.

6.1.3. Simulation Process in EMS

EMS organizes the simulation process on three levels:

1) meta level (conceptual model) – modeling of interaction of aggregates, i.e., work of model as a whole;

2) macro level (mathematical model) – modeling of functioning of separate aggregates;

3) micro level (software model) – modeling the functioning of E-network transitions.

Execution of the simulation program at the meta level begins with the compilation of a list of aggregates containing events scheduled for the current model time.

At the macro level, the first aggregate is activated, after which the list is adjusted by removing this one from it. This will happen until all the aggregates planned for the current time have elapsed, as a result of which the list of aggregates will be empty. The activity of the aggregates depends on whether they have ready-to-operate transitions. If a transition at the current time has to end its active phase, the related notification in the control list is removed and the transition becomes executable.

When all the planned and ready-to-operate transitions have been completed, the simulation program begins to check the presence of output signals in the model units by viewing the interface matrix. If the output signals are available, the states of the units change according to the coupling schemes, which, in turn, can lead to another adjustment of the list of ready-to-operate units. These steps are repeated until all the planned units have been worked out and all the external signals have been worked out.

Model time advances on the principle to the nearest event, which leads to a significant acceleration of the modeling process compared to the principle of a constant step. To implement the discrete-event modeling adopted in the EMS system, the capabilities of the basic language in terms of organizing simulation experiments were used. Scheduled moments of transitions firing are fixed in the control list. Since event notifications are arranged in a strict order of time, it allows simply calculating the time of the next event, leading to a change in the model state.

The micro level implements the process of functioning of individual transitions that make up the unit. Execution of any transition involves the sequential passage of the following three phases:

a. readiness, when the transition is not in a delay and the condition of its triggering (analysis of positions related to this transition) determined by a specific type of transition is satisfied;

b. delays, when the time is counted until the transition is triggered; the phase duration is determined by the transition delay time, which must be calculated before entering the delay phase; the state of the transition positions until the end of the delay phase does not change;

c. firing when, after the delay time has elapsed, there is an instant change in the marking of the transition positions by moving the tokens from its input positions to the output ones in accordance with the rules for triggering transitions of this type; at the same time, the values of the attributes of the tokens placed in the output positions and the model variables are changed in accordance with the specified transition transformation procedure.

Simulation with the use of predictive models involves checking of all possible paths in CEN model, which are formed through the use of X-Transition, which provides the choice of alternatives, and F-Transition, which provides the branching of the path.

When the "X" type transition fires, all input positions are checked and which of them was triggered during a specific simulation is compared. There can be only one such position. Its name is added to the path being checked, and information about it is added to the resulting path map.

When the transition type "F" is analyzed, the passage is analyzed in all paths. If the output position of this transition is not the last, its name is added to the stack and new path is added to the resulting path map. Next, a recursive method is called, which performs a passage to the end of the path with a return to the place of branching. Then another position is removed from the stack and the current path number is increased for further passage. After all the starting positions are passed, there is an exit from the recursion.

For all transitions of other types, only one position is selected, which is added to the map of the current path.

The check of execution of the set formula of TL is carried out for each position of each path, and results of such checking accumulate in the map of states of positions.

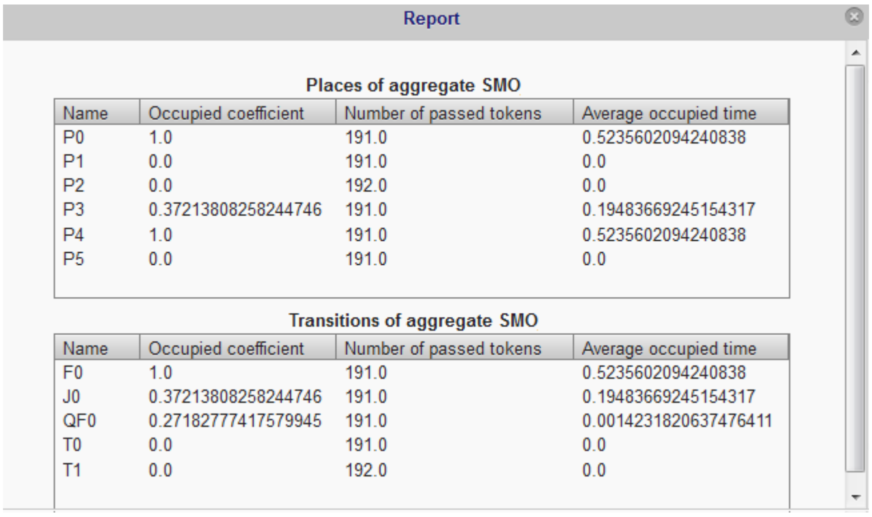
6.1.4. Processing of Statistical Information in EMS

EMS provides two types of collected statistics distinguished by their level of definition: primary and secondary. Primary statistics characterizes the operation of transitions and states of positions during the simulation and can be specified both for all aggregates of the CEN model and for those selected by the user using the graphical interface subsystem.

As a result of collecting primary statistics in the form of tables (Fig. 6.9), the following information is displayed:

1)position (transition) number;
2)occupied coefficient calculated as the total time of occupation of a position (delay for transition), which is divided by the simulation time, for queues – the average length of the queue is given;
3)number of passed tokens through the position (transitions), for queue positions – the number of tokens that visited the queue during the simulation interval;
4)average occupied time of a token staying in a position (staying in a delayed transition state), equal to the total busy time divided by the number of fires; for queues – the average time spent by tokens in the queue.

It is important to note that when calculating the standard numerical characteristics displayed in the tables of primary statistics, data are accumulated for all performed runs, while the value of the simulation interval is determined as the total time of all performed runs. Thus, it is possible to obtain data on the operation of the E-networ objects for the entire experiment.



Report

Places of aggregate SMO

Name	Occupied coefficient	Number of passed tokens	Average occupied time
P0	1.0	191.0	0.5235602094240838
P1	0.0	191.0	0.0
P2	0.0	192.0	0.0
P3	0.37213808258244746	191.0	0.19483669245154317
P4	1.0	191.0	0.5235602094240838
P5	0.0	191.0	0.0

Transitions of aggregate SMO

Name	Occupied coefficient	Number of passed tokens	Average occupied time
F0	1.0	191.0	0.5235602094240838
J0	0.37213808258244746	191.0	0.19483669245154317
QF0	0.27182777417579945	191.0	0.0014231820637476411
T0	0.0	191.0	0.0
T1	0.0	192.0	0.0

Fig. 6.9. Tables with the results of primary statistics.

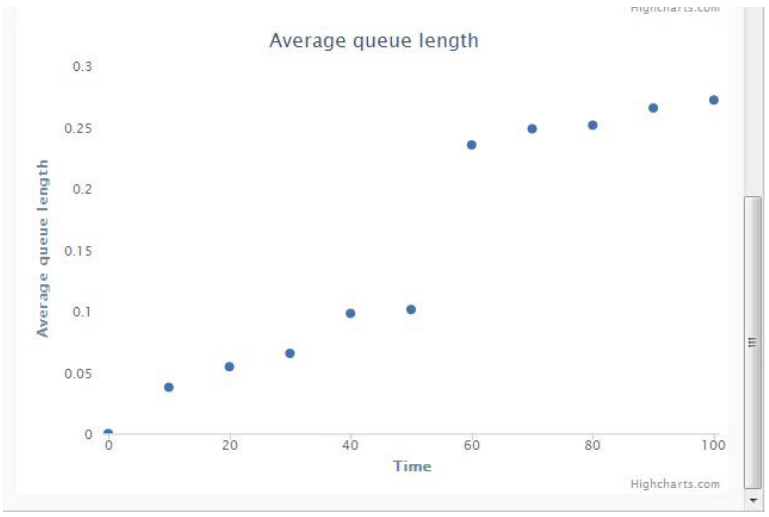


Fig. 6.10. Graph of average queue length changing in time.

Primary statistics provide detailed information about the dynamics of the simulation process and can be actively used at the stage of model debugging. However, it should be remembered that collecting primary statistics is time consuming. Therefore, when studying a model with a large number of runs, it is recommended to disable the collection of primary statistics, which can be easily done when forming an experiment using the graphical user interface subsystem. At the same time, the primary statistics may be sufficient to obtain complete information about the characteristics of the studied models of interest to the user. In addition, EMS provides for the selection of CEN objects, for which it is necessary to collect primary statistics during the experiment.

For a more complete presentation of the experiments carried out in EMS, the possibility of obtaining secondary statistics is implemented. Secondary statistics reflects the behavior of certain characteristics, which are responses of the modeled system to the given values of its parameters, which are used as factors. It is possible to vary the values of the factors. In this case, the resulting report will represent the dependence of the response on the selected parameter in the form of graphs or histograms, examples of which are shown in Figs. 6.10 and 6.11.

During the simulation with the use of a predictive model, a logical result variable is initialized, which is assigned the correct value. Then, depending on which operator was selected, the following is performed:

1. If "AG" – a passage through all the alternative paths and checking for at least one false value, the detection of which concludes that the resulting variable is also

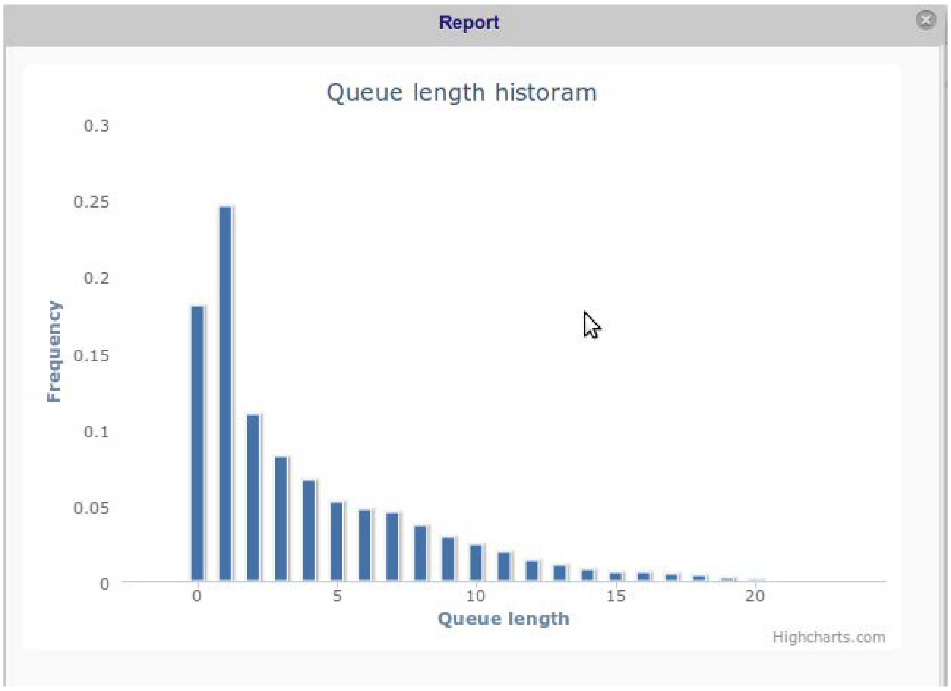


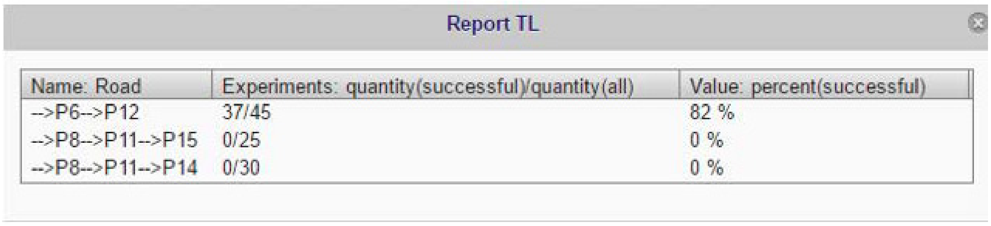
Fig. 6.11. Histogram of average queue length changing in time.

false, and further analysis does not make sense; otherwise, the result remains TRUE.

2. If "AF" – an auxiliary map is initialized to accumulate the results in every path; the passage is performed on all paths and it is checked whether the path has true states; the result of this check is recorded in the auxiliary folder; after passing, the result is formed, which will be true if the auxiliary map does not contain false values, otherwise – the result is false; the generated result is assigned to the result variable.

3. If "EG" – as in the previous case an auxiliary map is initialized to accumulate the results in every path; the passage is performed on all paths and it is checked whether the path has wrong conditions; the result of this check is recorded in the auxiliary folder; after the pass, a result is formed that will be true if the auxiliary map contains at least one true value, otherwise the result is false; the generated result is assigned to the resulting variable.

4. If "EF" – an auxiliary map is initialized to accumulate the results in every path; the passage is performed in all paths and the presence of at least one correct condition is checked; the result of this check is recorded in the auxiliary folder; after the pass, a result is formed that will be true if the auxiliary map contains at least one true value, otherwise the result is false. The generated result is assigned to the result variable.


Report TL

Name: Road	Experiments: quantity(successful)/quantity(all)	Value: percent(successful)
-->P6-->P12	37/45	82 %
-->P8-->P11-->P15	0/25	0 %
-->P8-->P11-->P14	0/30	0 %

Fig. 6.12. The resulting table of model checking.

Based on the result of model checking the execution of the formulas of DCTL, the resulting table is formed (Fig. 6.12).

Evaluation of the success of the TL formulas is given as the results during a multi-run experiment.

6.1.5. Organization of Simulation Experiments in EMS

EMS provides both strategic and tactical planning of the experiment. With regard to the strategic planning of the experiment, the system provides for only one-factor experiments. In this case, the parameters of the model are assumed to be constant, and only one of them is changed over the entire range of values. If necessary, you can sequentially conduct an experiment for each parameter separately.

The parameter that changes during the experiment is called a factor, the values of the parameter – the levels of the factor, the obtained values of the investigated quantity corresponding to the levels of the factor – the responses. The number of factor levels is not limited and is set indirectly by determining the initial and final values of the factor, as well as the interval of its change. In a particular case, only one point of the selected parameter can be specified, which will correspond to a single-level experiment.

Regarding the tactical planning of the experiment, EMS provides two options for carrying out the simulation:

1) with a predetermined number of runs to obtain each response point at fixed values of the factor;

2) the determination of the required number of runs in accordance with the rule of "automatic stop".

In the second case, the assumption of independence and normal distribution of response values is used. This assumption is based on the application of the central theorem of probability theory. The mathematical expectation of the response is taken as the parameter estimated using the expectation value.

The “automatic stop” rule is based on the confidence interval method. In this case, let us assume the accuracy d of the mathematical expectation E. The response y and the level of significance a, which guarantees that E falls inside the intervals $(Y + d, \; Y - d)$ with the probability $p = (1 - a)$, are assumed. In this case, Y is the mean value calculated over a sample of volume N and is an estimate of E.

EMS provides options for calculating confidence probabilities for the three most frequently used confidence probabilities p, equal to 0.90, 0.95 and 0.99. The values of the deviations d, characterizing the accuracy of the estimate of the mathematical expectation E, at the specified confidence probabilities p are calculated on the basis of two-sided statistics with a normal distribution and correspond to:

$$d = 1.64\sqrt{\frac{D}{N}}, \text{ if } p=0.90, \tag{6.1}$$

$$d = 1.96\sqrt{\frac{D}{N}}, \text{ if } p=0.95, \tag{6.2}$$

$$d = 2.58\sqrt{\frac{D}{N}}, \text{ if } p=0.99, \tag{6.3}$$

where d – a given confidence interval;

D – variance;

N – the number of runs (sample size).

The algorithm for choosing the number of model runs to obtain an estimate with a given accuracy includes the following steps:

1) the initial value of the sample size N is set, equal to 30, and the sample mean and the corrected variance of the estimate are found from the sample of this size;

2) according to one of the formulas (6.1, 6.2, 6.3), selected in accordance with a given confidence probability, the value of the achieved confidence interval d_n is found;

3) the specified accuracy d is compared with the achieved accuracy d_n. If the inequality $d > d_n$ is fulfilled, the required accuracy is achieved in N runs, and the algorithm goes to Step 5. Otherwise, Step 4 is performed;

4) one more run of the model is performed, the value of N is increased by 1 and the transition is to Step 1;

5) the end of the experiment.

Therefore, to enable the “automatic stop” rule, at the stage of setting the experiment parameters, the user must set the values of the confidence probability and the required accuracy of the result – the confidence interval.

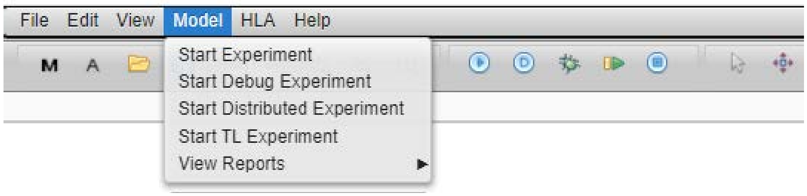
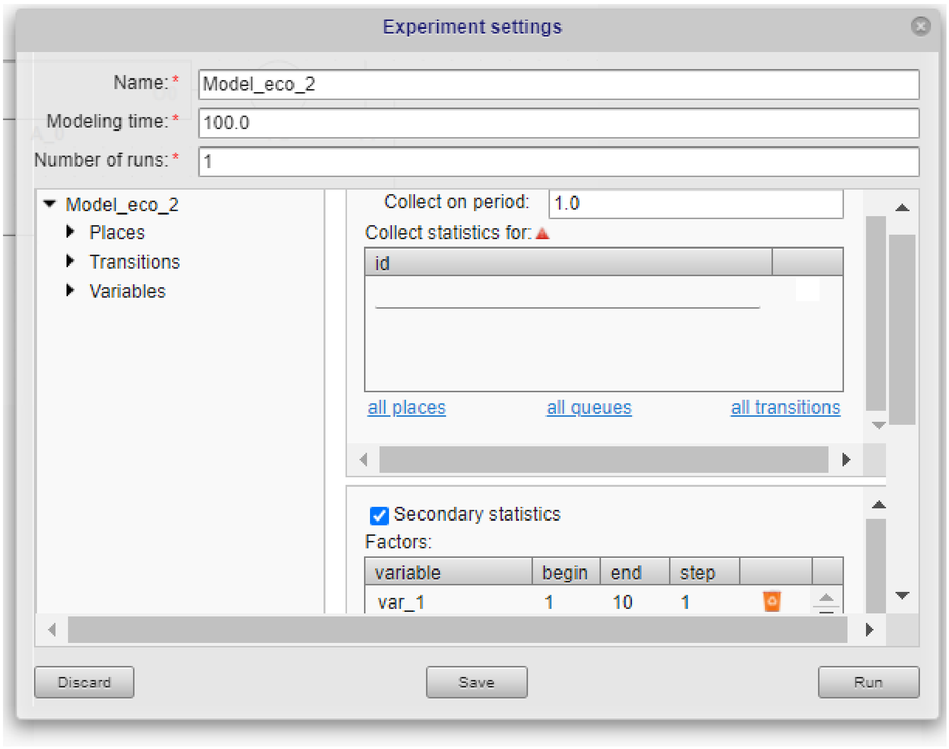


Fig. 6.13. Menu to start experiments.

The experiment is started via the model menu item (Fig. 6.13).

The experiment parameters are set in a special Experiment Setting window (Fig. 6.14).


Experiment settings

Name:* Model_eco_2

Modeling time:* 100.0

Number of runs:* 1

Collect on period: 1.0

Collect statistics for:

id

☑ Secondary statistics

Factors:

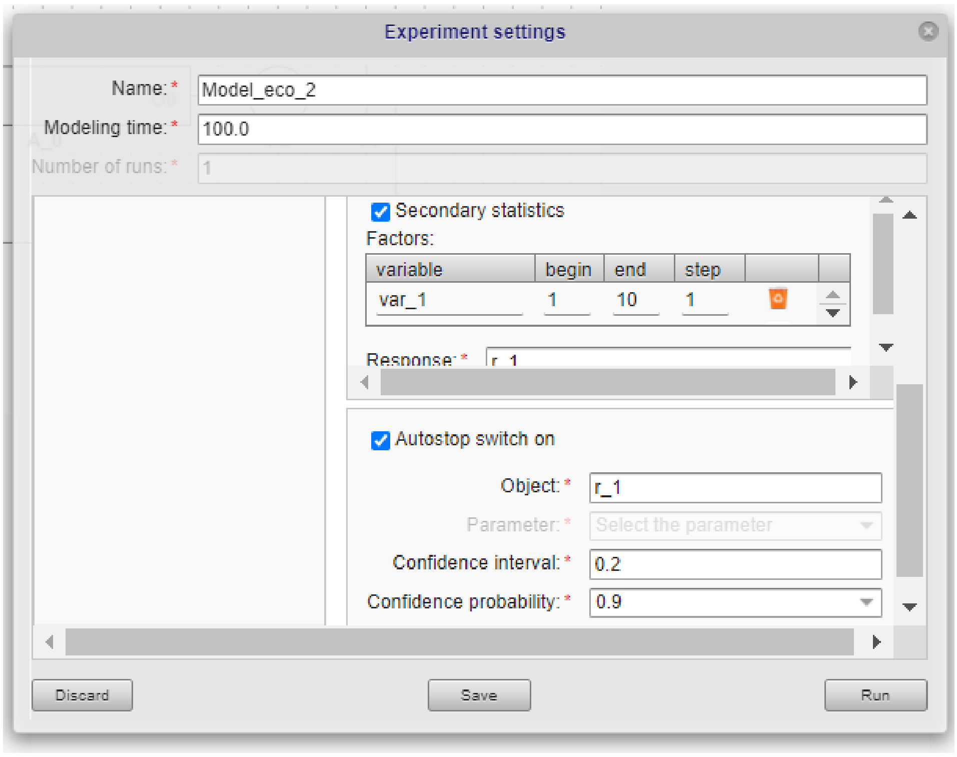
variable	begin	end	step
var_1	1	10	1

Fig. 6.14. Experiment Setting window.

Mandatory fields for starting an experiment are the following: experiment name, model time, and number of model runs. To collect statistical data, you must specify additional parameters of the experiment: select the required statistics collection mode (or both), specify the network components for collecting primary statistics, set the factor values and determine the response. In the case of distributed modeling, you must also specify as parameters the name of the federation in which the modeling is performed, as well as the IP addresses of the clients on which the models will run and the specified statistics are collected.

Additionally, parameters for automatic stop mode can be set in the Experiment window (Fig. 6.15).

Parameters for experiment with model checking of TL formula are set in



Name: * Model_eco_2

Modeling time: * 100.0

Number of runs: * 1

☑ Secondary statistics

Factors:

variable	begin	end	step
var_1	1	10	1

Response: * r_1

☑ Autostop switch on

Object: * r_1

Parameter: * Select the parameter

Confidence interval: * 0.2

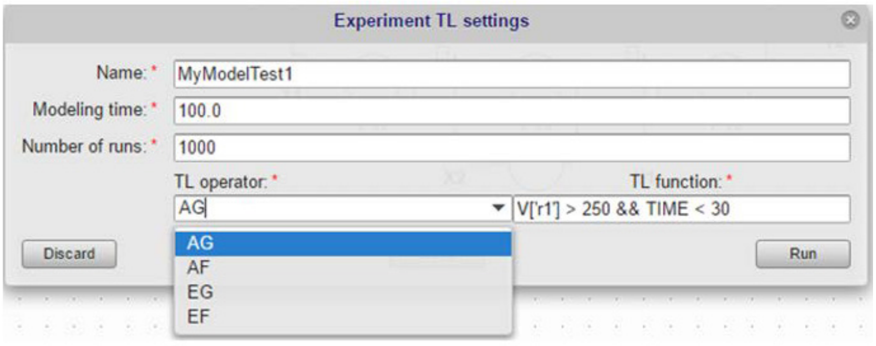
Confidence probability: * 0.9

Fig. 6.15. Automatic stop setting.

Experiment TL setting window (Fig. 6.15) that is opened by Start TL Experiment item.

Required fields to run a TL experiment are the following:

- name of the experiment;
- model time;
- number of runs of the model;
- operator of temporal logic;
- formula that will be checked by this operator.


Experiment TL settings

Name: * MyModelTest1

Modeling time: * 100.0

Number of runs: * 1000

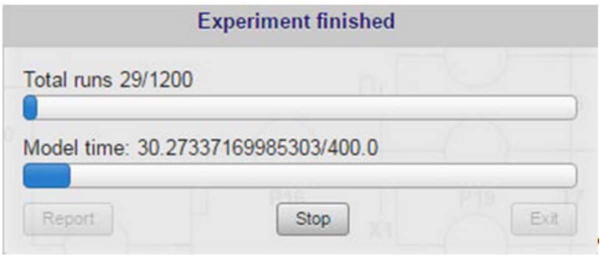
TL operator: * AG

TL function: * V['r1'] > 250 && TIME < 30

Fig. 6.15. Experiment TL setting window.

The formula determines the condition that must be met and will be checked for each state of the model. The formula can consist of several expressions that are combined by a logical operator.

The EMS provides a special window with information about the progress of the experiment, as a result of which the user receives information about the number of runs (if a multiple experiment is specified), factor levels (if a factorial experiment is


Experiment finished

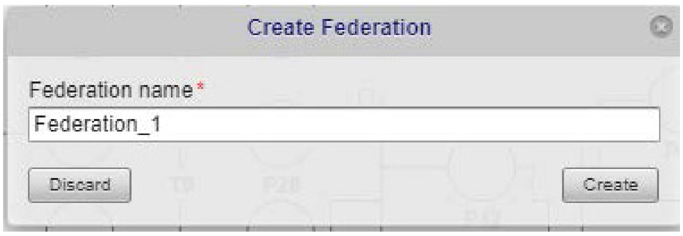
Total runs 29/1200

Model time: 30.27337169985303/400.0

Fig. 6.16. Experiment progress window.

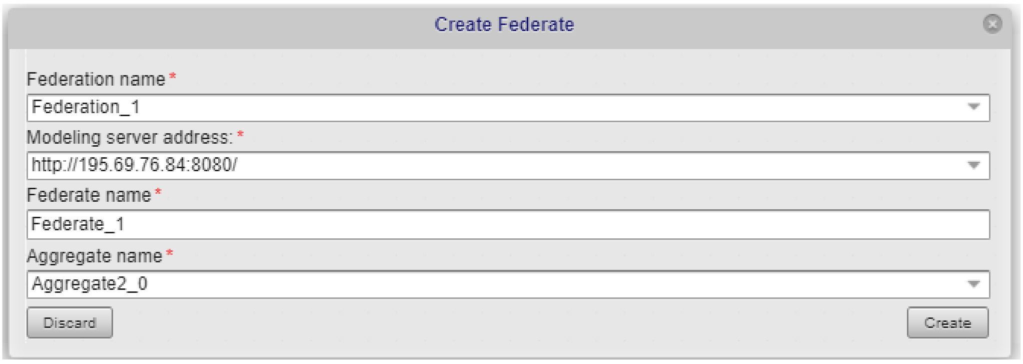
specified) and the time remaining until the end of simulation (Fig. 6.16).

For modeling large models, the system supports the distributed simulation capability based on the High Level Architecture (HLA) standard (HLA, 2017). In the distributed mode, the models are executed on the simulation servers, on which the Java Virtual Machine (JVM), the Tomcat application server and the EMS must be preinstalled. To perform distributed modeling, it is necessary to create federation and federates based on the developed models via HLA item of the main menu (Figs. 6.17 and 6.18).



Federation name*
Federation_1

Fig. 6.17. Federation creation window.



Federation name*
Federation_1
Modeling server address:*
http://195.69.76.84:8080/
Federate name*
Federate_1
Aggregate name*
Aggregate2_0

Fig. 6.18. Federate creation window.

The interaction of simulation servers in the NLA is carried out through a special service called Run-Time Infrastructure (RTI). The EMC uses the Portico software implementation of RTI (Portico, 2021). All data exchange between federates takes place through RTI (Fig. 6.19).

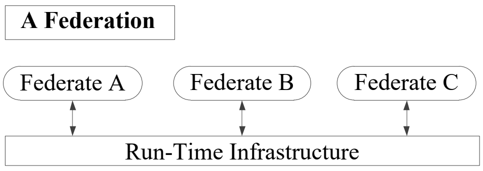


Fig. 6.19. HLA component interaction scheme.

6.1.6. Storing Models in EMS

EMS uses XML format for storing and serializing data. Petri Net Markup Language (PNML) is an international standard (PNML, 2021) that defines the format for storing the structure of a Petri net in XML. PNML allows you to store information about positions, transitions, connections between them, as well as data about the coordinates of each object, its color, signature, etc. when displaying a Petri net.

To work with CEN models, the international PNML standard has been expanded (Fig. 6.20) by adding new PNML attributes to transition and place objects, taking into account such features of CEN as:

- the presence of transitions and queues of different types;
- the presence of nested aggregates;
- the ability to set functions at each transition;
- availability of various input / output variables;
- the presence of a token with attributes.

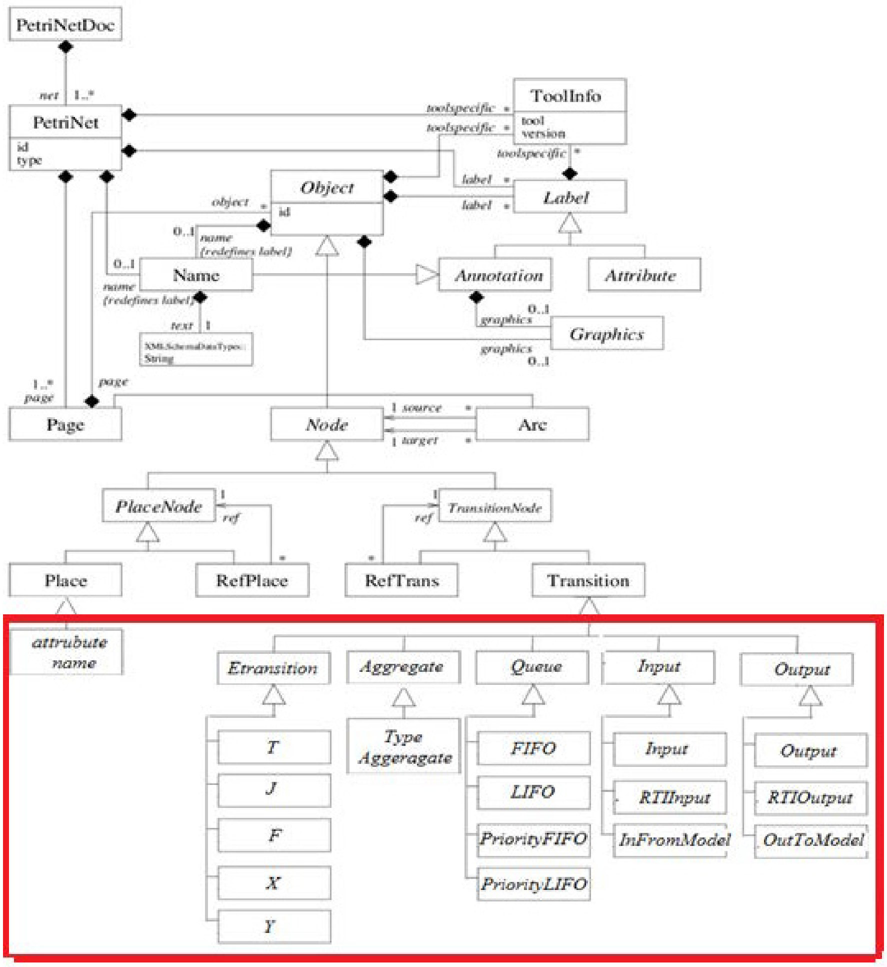


Fig. 6.20. CEN PNML expansion.

Different models may include aggregates of the same structure, but with different parameters. The EMS introduced the concept of the type of aggregate as a set of all aggregates with a single structure. This approach makes it possible to reuse the developed aggregate in different models, but at the same time to save the description of its structure only once. If necessary, it is possible to change all aggregates of the same type in different models, if during the development of the model the user made adjustments to its structure.

Below is an example of XML file describing the model in an extended PNML format:

```
<?xml version="1.0" encoding="UTF-8"?>

<model xmlns="http://www.cs.stu.cn.ua/jess/enetsdefinitions">

    <rootAggregate type="Root">

        <aggregate name="A1">

            <transition name="T1">

                <transformationFunction type="el">

                    RETURN 9 + 1;

                </transformationFunction>

            </transition>

            <aggregate name="child">

                <place name="P1">

                    <initialMarking>

                        <attribute name="attr" value="12.34" />

                    </initialMarking>

                </place>

            </aggregate>

        </aggregate>
```

```
            <aggregate name=”A2”>
                <variables>
                    <variable name=”varName” value=”34.56” />
                </variables>
                <transition name=”T1”>
                    <transformationFunction type=”el”>
                            RETURN rand;
                    </transformationFunction>
                </transition>
                <queue name=”queueName”>
                    <priorityFunction>
                            RETURN 1;
                    </priorityFunction>
                </queue>
            </aggregate>
        </rootAggregate>
</model>
```

This approach makes it possible to automate the process of constructing software models and serialization of data based on their XML descriptions, as well as to set standard graphic data defined by the standard for all objects of the CEN.

6.2. Technology of Embedding Models into the Control Loop

The advantage of the considered technology for the development and study of models for implementing AC using EMC is that these models actually implement control programs, which can then be directly executed by controllers. It means that for the practical application of the control algorithms developed and programmed in the CEN language, it is sufficient to construct a software interpreter of the implementation models capable of executing a set of CL instructions on the chosen hardware platform. For this purpose, it is necessary to solve two main problems: to develop algorithms for the operation of a software interpreter and to provide a model execution environment.

6.2.1. Model Interpreter

The main task of the model interpreter is to ensure the cyclicity inherent in reactive systems. In this regard, it is necessary to provide for the possibility of forced initialization of the aggregate when transferring control to it by sending a token. For this procedure, the input boundary positions have the INIT flag, which can be set for a position if the receipt of a token in it should cause the restoration of the initial marking of the aggregate and the initial values of its variables, as well as the reset of the delay time counters of all transitions.

Compliance with working cycle determines two main differences in the operation of the interpreter of the modeling system and the controller that directly controls the process. The first difference is the concept of time used. If in EMS the simulation occurs in model time, which advances from event to event recorded in the control list, then the controller monitors real time, which moves from cycle to cycle and is compared with the time of scheduled events.

The second difference is related to signal changes. In the simulation system, it is allowed working with signals as with ordinary variables, i.e., changing their values using assignment operators. When executing a control program in the controller, emulation of the values of input signals inside the transition transformation functions is not allowed. These functions can only set the output signals.

The controller cycle structure includes four main phases, which are executed sequentially (Fig. 6.21).

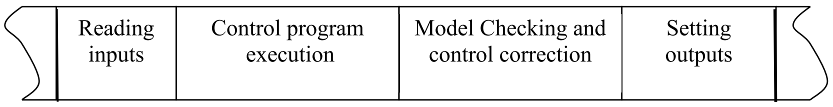

Reading inputs
Control program execution
Model Checking and control correction
Setting outputs

Fig. 6.21. Controller cycle structure.

At the beginning of the cycle, the interpreter reads the values of the input signals using calls to the runtime functions. The resulting input values are assigned to the aggregate variables. These variables make up shared memory, the state of which is frozen until the next cycle.

In accordance with the EMS methodology for constructing CA implementation models, there are three levels of operation of the software interpreter when executing the control program code:

1. Meta level (model level) – the work of the aggregate model as a whole, including the synchronous cyclic execution of aggregates and ensuring their interaction by passing tokens.

2. Macro level (aggregate level) – operation of an individual aggregate in accordance with its CEN scheme.

3. Micro level (transition level) – the work of CEN-transitions according to the rules encapsulated in the corresponding classes.

The meta-level algorithm includes the following steps:

1. Walk through the list of aggregates. Model aggregates are sequentially activated.

2. Check the marking of the boundary positions. If no boundary position has received a token, go to Step 4.

3. Activate the aggregate linking scheme. The marking of the input positions is changed. Go to item 1.

4. Change the model time. The model time value is set equal to the current real time. Go to item 1.

The work of the interpreter at the macro level includes the following steps:

1. Check the initialization sign. The marking of the input boundary positions with the initialization sign is checked. If at least one such position is marked, the initial state of the aggregate is set. It does not change the marking of the boundary positions.

2. Walk through the list of transitions of the aggregate. If the transition is of type X or Y, then it is entered into the list of pending transitions, otherwise it is activated. If there are no transitions ready for firing, go to Step 4.

3. Pass through the list of pending transitions. If at least one transition has worked, then go to Step 2.

4. Return the control. Go to the meta level.

At the micro level, the interpreter performs the actions caused by the activation of the transition, which include the following steps:

1. Change the transition delay time. If the transition is in delay, then the remaining delay time is calculated. Then go to Step 6.

2. Check the conditions of readiness. Calculate the decision function and verification of the transition readiness conditions. If the conditions are not met, then go to Step 6.

3. Check the activity of the transition. If the conditions are not met, then go to Step 6.

4. Calculate the delay time. If the value of the delay time is calculated, then place the transition in the list of delayed transitions and go to Step 6.

5. Fire the transition. Execute the conversion function. Change the marking of transition positions in accordance with the triggering scheme.

6. Return the control. Go to the macro level.

At the end of the processing of the control program code, the obtained values of the DCTL formulas in the units are checked. The formula receives the value TRUE only if this fact is confirmed by all the aggregates in the model. Then, for each formula, the program performs actions to change the transition functions, which consist in changing dynamically an internal variable, which influences the control program by changing the control parameters or control path.

The final act of the software interpreter is setting the values of the output signals. This operation is carried out by calling special functions of the runtime environment that provide interaction with the hardware environment.

The algorithm of the software interpreter described above is repeated when it is used as part of the EMS modeling system. In the latter case, at the meta level, both the plan of the experiment is additionally tracked and the end time of the simulation. In addition, at the stage of designing models, the use of a runtime environment with special functions for interacting with the external environment is not required. In this case, it is possible to implement a distributed simulation. However, it should be noted that when using models as control programs for controllers, there is no need to use a special mechanism for synchronizing the execution of transitions. Real time, which is involved in calculating the duration of cycles in the controller, is itself a synchronizing factor that tracks the pace of processes in the realization models.

6.2.2. Porting of Runtime Interpreter

Algorithms for interpreting implementation models when used in real control systems require the presence of a runtime environment in the form of the necessary software. Regardless of the chosen programming language, the runtime environment must provide:

- deterministic CA operation in real time;
- timing with high resolution;
- direct work with memory when using input-output ports;
- support for network communication using standard protocols in the case of distributed use;
- multithreading when executing model components.

To a large extent, these requirements are covered by real-time operating systems (RTOSs), the most famous of which are VxWorks and QNX. All these systems are commercial; therefore, it is possible to use their own methods of organizing computations and, most importantly, embed the runtime environment in arbitrary processor devices.

The greatest flexibility in resolving this problem can be provided by Linux, which is a full-blown operating system, and for which there are versions of Java virtual machines. More importantly, Linux is freely distributed along with the source code. This allows you to develop and modify this operating systems (OS) to the level of Real-Time Linux (RTLinux) in order to use it as a basic runtime environment for implementation models.

However, taking into account the need to support a distributed control scheme, it is required to provide a porting of Linux OS to a specifically used microprocessor hardware platform, taking into account the existing limitations on hardware.

As a target hardware platform, let us consider a RISK processor family with the MIPS architecture, which has already become widespread among embedded systems. For example, these processors are known to be used as a platform for a compiling system, including mobile applications.

Among the main requirements for Linux OS ported to the RISK platform, we define the following:

1. The use of real memory without a hardware memory manager (Memory Management Unit –MMU). Fulfilling this requirement will provide significant resource savings.
2. Compliance with existing standards, primarily for compliance with the POSIX (Portable Operating System Interface) specification. Fulfillment of this requirement will ensure compatibility of the developed programs for Linux OS

at the source code level.

3. The need to support a USB interface for connecting external devices and organizing field buses, as well as accompanying microcontrollers restrictions on available memory.

The result of solving this problem should be the creation of a Plinux OS capable of running on an RISK processor without an MMU as an execution environment for implementation models.

The developed Linux porting technology includes the following sequence of steps:

1) selection of the base kernel;

2) analysis of the structure of the kernel in order to determine the hardware dependent parts of the code that require modification;

3) determination of the core limitations due to the lack of MMU;

4) determination of the organizational features of porting;

5) kernel modification.

Choosing a base Linux kernel for modification.

Among all known Linux kernel versions targeting embedded systems without MMU, a version called uClinux (read as "you see linux") may be considered (Ungerer, 2005). It can be considered a universal relative to the supported platform. The advantages of this OS version include a large number of freely available versions of the kernel without MMU, libraries and utilities, as well as the USB support needed in the uClinux.

In the version of Plinux discussed below, uClinux was used as the base OS kernel, which was reworked taking into account the requirements formulated above. Note that choosing the most suitable kernel as the base is an important consideration as it determines the overall development time.

An analysis of the internal structure of the kernel of the uClinux operating system shows that the hardware-dependent sections of the program code that require changes when porting Linux to another hardware platform include:

1. In the task manager:

- the function of generating a new process. It is required to set the process context to the initial state;

• the function of loading the program into memory *exec*(). It is necessary to create a program loader in ELF and / or other formats, the implementation of which is largely determined by the hardware architecture of the chosen platform;

• the function for switching the context of the *schedule*() task;

• initialization of the null task *idle*(). It is required to set the process context to the initial state;

• changing the structure of the process context, as well as the sequence of its saving and restoration, taking into account the hardware architecture.

2. In the memory manager:

• changing the level of paging to work with real addresses (without MMU);

• changing the level of memory areas, since in real memory addressing, a memory area should be allocated from pages that continuously follow each other;

• removal of the *brk*() function, since there is no way to change the size of the memory area after its allocation (since the memory area must consist of continuously successive pages).

3. In the subsystem of interaction between processes (Interprocess Communication – IPC), all components are hardware-independent, except for the functions of shared memory. They cannot be implemented due to the lack of an MMU. Therefore, these functions should be redesigned as stubs, i.e., they should always return an error code.

Kernel functionality limitations.

Separately, we note the limitations in the functionality of the uCLinux OS kernel compared to the Linux kernel functionality caused by the lack of virtual addressing:

1. No memory protection – processes can address all memory types, including kernel memory.

2. No swapping – since memory pages are located at real addresses, it is impossible to swap pages between memory and disk.

3. The size of the memory area cannot be changed.

4. Since the *fork*() function is replaced by *vfork*(), it is impossible to create a copy of the process with the same addresses (as with virtual addressing).

Features of porting to the RISK platform.

Due to changes in the kernel architecture, the following have been developed additionally:

1. Loader of ELF files.

2. Library of the LibC programmer.

3. The interpreter of commands (shell) in a simplified version.

4. A set of command files and programs for initializing the system, which includes:

- unpacking a compressed binary image of the OS kernel into memory;
- transfer of control to the uncompressed binary kernel image;
- hardware initialization;
- system initialization;
- starting the zero (*idle*) and the first process (*init*), etc.

The Plinux process that this architecture generates is described as follows. Source codes are compiled using MIPS processor-specific cross compilers. The object files are then linked to the LibC programmer library. The resulting executable files in ELF format, together with the compiled kernel, make up an image that is loaded into ROM. Then the loader expands this image and transfers control to the initialization program, which starts the runtime environment.

6.2.3. Hardware Platform of Hybrid Embedded Models

For the practical use of implementation models of control algorithms in the form of CEN, a number of requirements for their hardware platform must be met (Khropatyi, Lohinov and Kazymyr, 2020):

- the ability to update control programs and conditions for interaction with the external environment;
- high-speed performance with support for the logical capabilities of control programs and during their execution;
- the possibility of dynamic verification of control algorithms;
- the availability of high-speed hardware memory elements;

- the possibility of flexible reconfiguration of the hardware platform for specific control tasks;
- the ability to synchronize the logical elements of the control program, which are SDT transitions and support for time delays on them.

The previously considered solutions based on real-time operating systems are not quite suitable for the embedded implementation models, since they do not provide effective load balancing in terms of program execution time, fast response to changes in external conditions. Moreover, such solutions create many problems with the deployment of the runtime environment on microprocessors.

Therefore, a hardware architecture should support not only the methodology for building implementation models in form of CEN specifications, but also provide high performance, including aggregating of models. Additional possibilities include interaction with databases, special management and configuration services.

The best way to satisfy these requirements is the use of a hybrid platform, in which high-speed hardware blocks provide the computational functions corresponding to CEN transition on the one level, and anoter level supports the interaction between aggregates corresponding to the control program modules by passing input and output signals.

A combination of ARM (Advanced RISC Machine) and FPGA architectures is the most suitable solution. ARM, originally Acorn RISC Machine, later Advanced RISC Machine, is a family of Reduced Instruction Set Computing (RISC) architectures for computer processors, configured for various environments. The 32-bit ARM architecture is supported by a large number of embedded and real-time operating systems, including Android, Linux, FreeRTOS, VxWorks, Windows Embedded Compact, Windows 10 IoT Core, ChibiOS/RT, DRYOS, eCos, Integrity, Nucleus PLUS, NuttX, MicroC/OSII, PikeOS, QNX, RIOT, RTEMS, RTXC Quadros, ThreadX, MQX, T-Kernel, OSE, OS-9 (Zlatanov, 2016).

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to the desired application or functionality requirements after manufacturing. Due to their programmable nature, FPGAs are an ideal fit for many different markets in particular for CPS applications (https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html):

aerospace & defense – radiation-tolerant FPGAs along with intellectual property for image processing, waveform generation, and partial reconfiguration;

automotive – automotive silicon and IP solutions for gateway and driver assistance systems, comfort, convenience, and in-vehicle infotainment as Xilinx FPGA enabled automotive systems;

Industrial – Xilinx FPGAs and targeted design platforms for Industrial, Scientific and Medical (ISM) purposes enable higher degrees of flexibility, faster time-to-market, and lower overall non-recurring engineering (NRE) costs for a wide range of applications such as industrial imaging and surveillance, industrial automation, and medical imaging equipment.

Actually, it is somewhat misleading to present an FPGA as a standalone component. FPGAs are always supported by development software that carries out the complicated process of converting a hardware design into the programming bits that determine the behavior of interconnects and CLBs (ttps://www.allaboutcircuits.com/technical-articles/what-is-an-fpga-introduction-to-programmable-logic-fpga-vs-microcontroller/).

The combination of FPGAs and microprocessors in one hybrid platform may be one of the best solutions. The possible structure of a hybrid platform is shown in Fig. 6.19.

In this example, an architectural solution for the hybrid platforms is based on microchips of the family Xilinx Zynq-7000 as it was proposed by Albert and Yao (2010). The structural scheme in Fig. 6.22 includes additional I/O between different components.

Combining FPGA with a high-performance ARM processor in one physical device is the main advantage of such a decision (Kalachev, 2013). The control program, presented in the form of CEN, is schematically divided into two components. One of them is the program blocks associated with the operation of CEN transitions. They are implemented by the FPGA level. All communication functions are assigned to the ARM. In fact, the

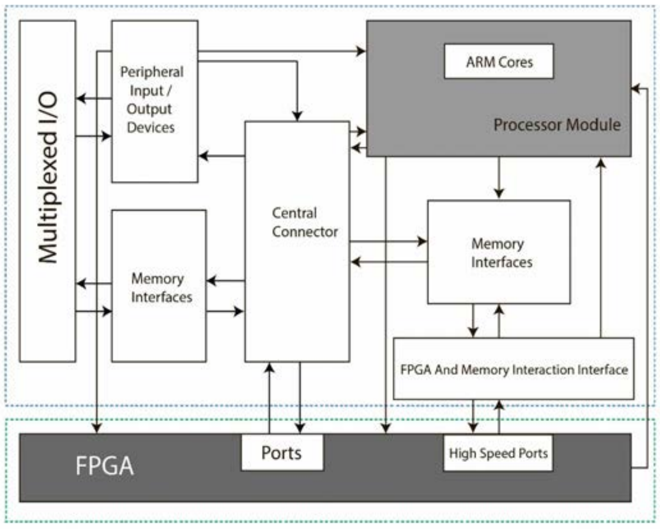


Fig. 6.22. Structure of the hybrid platform.

ARM performs an integrating function for linking CEN transitions. The control program represented in the XML format is converted into a code executed by the ARM. This code then uses predefined FPGA blocks to implement the CEN transition logic. Changing the program, if necessary, does not require reconfiguration of the FPGA – only the sequence of the transition calls for the execution of the prescribed logic is changed. It reduces the time for modifying the control program depending on the operating conditions.

The choice of the ARM processor is the most important architectural decision (Palagin and Yakovlev, 2017). However, it is important to mention the use of the AMBA AXI (Advanced Extensible Interface) broadband interface. Thus, it will be possible to implement data transfer from/to FPGAs at multi-gigabit speeds, simultaneously ensuring low-power consumption.

6.2.4. Neural-like Network Platform for Recovery Model Implementation

The Hamming network (Bruck and Blaum, 1989) is designed to solve problems of pattern recognition (images, etc.). It uses a set of templates to assign an input binary vector or several vectors if they have the same proximity measure. The Hamming distance is used as a maximum proximity measure.

Let $X = \{0,1\}$ be a binary alphabet and $F_n(X)$ – the set of all words of length n in the alphabet X that is called the complete set of words. It is obvious that . Let $|F_n(X)| = 2^n$. Let$v = (x_1x_2\dots x_n) \in F_n(X)$, and R be a cyclic shift operator with step 1, the definition of which has the form:

$\forall v = (x_1x_2\dots x_{n-1}x_n) \in F_n(X)\ Rv = (x_nx_1x_2\dots x_{n-1})$.

The subset V from $F_n(X)$ that is closed to the shift operator R, i.e., $\forall v \in V\ (Rv \in V)$. V will be called a cyclic Hamming code, which is a subset of words closed by Hamming distance in the alphabet X.

Let the unit word be a word consisting of units. Some chosen unit words will be called reference words. In the case of the Hamming distance equal to 1, all words from the set V will be generated by the cyclic shift operator. These words will be called the generating words.

Further we will consider:

$H_m(1)$ – a cyclic code of length n, which has the generative word with the m – component group ($m = 1 \div (n-1)$) of cyclically adjacent "1" and ($d = (n-m)$) – component group of cyclically adjacent "0";

$H(1)$ – a cyclic code of length n, which has generative word with n– component group of "1";

$H_m(0)$ – a cyclic code of length n, which has the generative word with m – component group of "0" and d – component $(d = (n - m))$ group of "1";

$H(0)$ – a cyclic code of length n, whose parent word includes n – component group consisting of "0".

Formulation of the problem.

Let $F_n(X)$ be a set of words with length n in the alphabet $X = \{0,1\}$ and $H(1)$ (or $H_m(0)$) be the cyclic code of length n defined above. It is necessary to build a logical structure that implements the mapping $\Im$, defined as follows:

$\Im(H_m(1)) = 1, \quad \Im(F_n(X) \setminus H_m(1)) = 0$,

(respectively, $\Im(H_m(0)) = 1, \quad \Im(F_n(X) \setminus H_m(0)) = 0$).

The problem can be solved by serial connection (Palagin, Opanasenko and Kryvyi, 2013) of cyclic structures such as AND and NOR, which have n inputs and n outputs, as well as an OR structure, which has n inputs and one output. In the general case, the problems of synthesis of multilevel logical structures for the classification of input binary vectors with one and many output structures for the task at hand have an r–level organization, the k –th level ($k = 1 \div r$) of which contains n AND logic gates that implement the transformation:

$$\omega(a_i,\ a_{(i+s)\,mod(n)}),\ \forall i = 0 \div (n-1),\ 1 \le s \le (n-1), \tag{6.4}$$

where ω – a logical function of two variables;

$a_i,\ a_{(i+s)}$ – the one-bit components of the input binary vector;

$s \in [1, n-1]$ – a step of cyclic shift.

All elements of the same level are configured to perform the same logical AND or NOR function.

The cardinality of the group of cyclic adjacent "1", based on the truth table of the logical function AND and the structure of links ($\forall k,\ s = 1$), with an increase in the level number k decreases by "1". Thus, to convert a codeword with a given value m, it is necessary to have $(m-1)$ levels.

The structures S_1 and S_2 ($n = 4$, the step $i = 1$ in the operator R) are selected as logical structures. Based on the truth table of logical operations AND and NOR, as well as the structure of links ($\forall k, s = 1$), the number of cyclically adjacent "1" with an increase in the level number k decreases by one for the AND operation, and for the NOR operation. On the contrary, the number of "0" will increase by 1 if the word is input containing d cyclic contiguous "1".

Let us consider the structures S_1 and S_2 synthesized earlier by Opanasenko and Kryvyi (2012). If to the input of the k -th level of a structure $S_1 (1 \leq k < r)$ with such connections is fed a word containing a group of d cyclic adjacent "0", then its output will be a word containing a group of $d + 1$ cyclically adjacent "0". It is shown that it is sufficient to investigate the type structure S_1, since the type structure S_2 after the first level turns into a type structure S_1 with an input word $H_{n-d-1}(1)$.

Theorem 1. a) If to the input of the first level of a computational structure S_2 with the NOR operation is fed a word with d cyclically adjacent "1", then the output will be a word with $d + 1$ cyclically adjacent "0".

b) If to the input of the k -th level of the structure S_1 $(1 \leq k < r)$ a word with d cyclically adjacent "0" is fed, then the output of this level will be a word with $d + 1$ cyclically adjacent "0".

The substructure S_1 identifies a nibble (a 4-bit word of "0" and "1") containing three or four "1s", and the substructure S_2 identifies a nibble containing one or all "0s". Nibbles consisting of all "0s" $H(0)$ or all "1s" $H(1)$ will be called singular points, the identification of which will be discussed below.

Let us consider now the general problem of classifying input nibbles that are not necessarily circular in structure. For this purpose, let us consider a basic structure consisting of two similar 4-bit substructures built from substructures S_1 and S_2, the outputs of which are indicated by symbols f_1, f_2, and g_1, g_2, respectively (Fig. 6.23).

This structure is justified by the following lemma.

Lemma. *If the singular points do not take part in the classification, then with the help of the given substructures the l units in the input byte are identified, where* $1 \leq l \leq 6$.

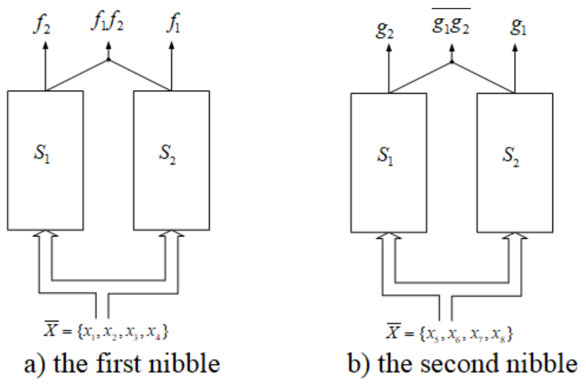


Fig. 6.23. Basic structure of cyclic code converters.

Substantiation. From Theorem 1 and the tables of values for the substructures it follows that the outputs of both substructures are mutually exclusive, i.e., if the output f_1 (f_2) gives 1, then the outputs f_2 and $\overline{f_1 f_2}$ (f_1 and $\overline{f_1 f_2}$) give the value 0. The same is true for (g_1,g_2).

The outputs of the substructures f_2, g_2 give 1 if in the input of the substructures S_1 and S_2 the nibble has exactly one "1" or all "0s". The outputs of the substructures f_1, g_1 give 1 if in the input of the substructures S_1 and S_2 the nibble has exactly three "1s" or all "1s". The output of the substructures $\overline{f_1 f_2}$ and ($\overline{g_1\, g_2}$) yields "1", if the input nibble of the substructures contains exactly two "1s".

It follows that the setting of both substructures for initialization in a byte of three "1s" takes the form: $(f_2\&\overline{g_1 g_2}) + (g_2\&\overline{f_1 f_2}) + (f_1 \oplus g_1)$. Setting for four "1s" in a byte (without taking into account the singular point $H(1)$):$(f_2\&g_1) + (g_2\&f_1) + (\overline{g_1 g_2}\&\overline{f_1 f_2})$. Setting for five "1s" per byte (excluding the special point $H(1)$): $(f_1\&\overline{g_1 g_2}) + (g_1\&\overline{f_1 f_2})$. Setting for six "1s" per byte (excluding singular point $H(1)$): $f_1\&g_1$.

The statement is proven.

An example of a network structure for identifying three "1s" in a byte is shown in Fig. 6.24.

Remark 1. In a general case, if a sample not from 2^k bits is fed to the input of such a structure, for example, a sample of 11 bits, then it is divided into two nibbles, and the remaining three bits are supplemented with one bit to a nibble, and, thus, the problem is reduced to the previous case.

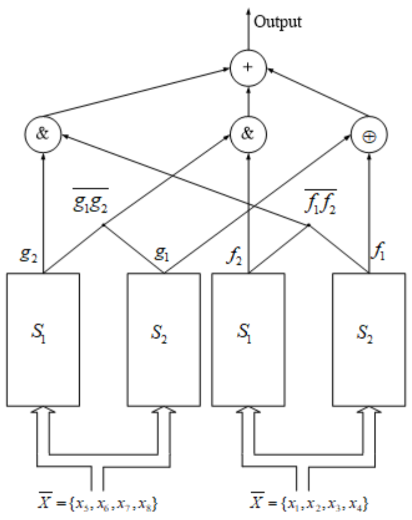


Fig. 6.24. Network structure for identifying three "1s" in a byte.

Identification of special points. From Statement 1 it follows that the setting for seven and eight units per byte requires taking into account singular points, in particular, points $H(1)$ in the input nibbles. Since the substructure S_2 (respectively, S_1) identifies together with three "1s" (three "0s") and four "1s" (four "0s"), it must be modified so that it is possible to separate the singular points from the rest. This can be done by simply modifying the basic structure S_2 (appropriately S_1) by replacing the OR output function with AND. The implementation of such structures is shown in Fig. 6.25 (structure for the singular point $H(1) - T_1$, and for the singular point $H(0) - T_2$).

Let us denote the outputs of these substructures f_3^0 (g_3^0) and f_3^1 (g_3^1) to identify four "0s" and four "1s", respectively.

Based on the above tables for base structures and modified ones, the output f_3^1 of the modified substructure will be 1 if the input nibble value is a singular point $H(1)$. Similarly to the second structure, the output f_3^0 will be 1 if the value of the input nibble is a singular point $H(0)$.

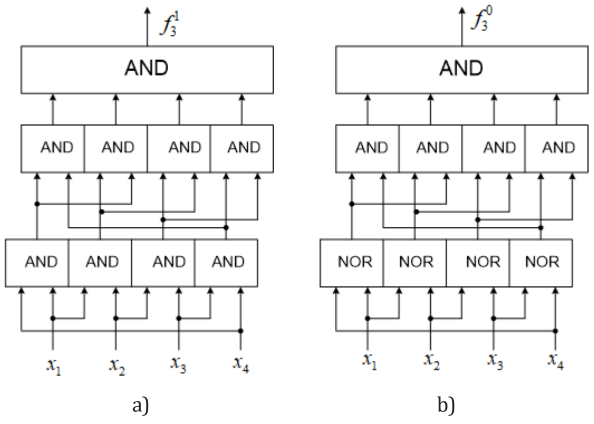


Fig. 6.25. Structure of a cyclic code converter based on AND and NOR operations for determining singular points: a) type structure T_1; b) type structure T_2.

Thus, connecting the outputs f_3^1 and f_1 using the AND operation, we get the output 1 only when there are four ones at the input in the nibble (singular point $H(1)$). Similarly, a semi-structure is constructed for identifying a singular point $H(0)$ on the basis of semi-structures f_3^0 and f_2. Then the above-described construction will look like as shown in Fig. 6.26.

Justification of the properties of the given structure gives the next theorem.

Theorem 2. The general structure identifies the arbitrary content of an input byte by setting its outputs to the number of "1" ("0") in the byte.

Evidence. It follows from the lemma and the substructures modified above that to

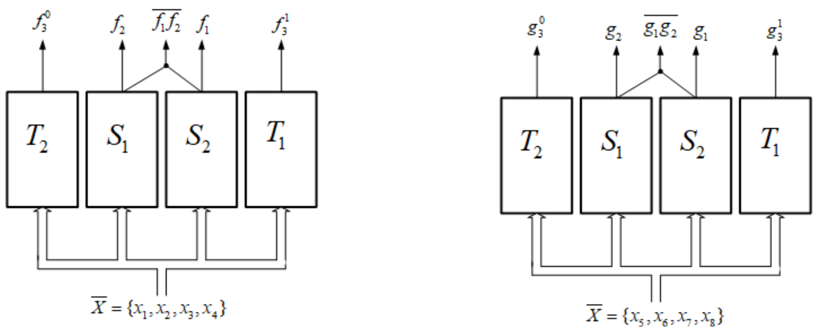


Fig. 6.26. The general structure of the network taking into account special points.

identify four "1" in a byte, the identification function will be as follows:

$$(\overline{g_3^1} \& f_2 \& g_1 \& \overline{f_3^0}) + (\overline{f_3^1} \& g_2 \& f_1 \& \overline{g_3^0}) + (\overline{g_1 g_2} \& \overline{f_1 f_2}) + [(f_1 \& f_3^1) \oplus (g_1 \& g_3^1)].$$

The units in the input byte can be distributed so that in the first nibble (in the second nibble) there will be one "1" on the first substructure, and in the second (first) nibble of the second substructure there will be three "1s".

From the form of the first two expressions in the above expression we find: if $g_3^1 = 0$, then it means that the input nibble of the second substructure contains no more than three "1s" and if $g_1 = 1$, then it contains exactly three "1s"; if $f_3^1 = 0$, then the input nibble of the first substructure contains more than three "0s", and if $f_2 = 1$, then it contains exactly one "1".

Then there will be exactly four "1s" in the byte. The validity of what has been said for the second expression follows from symmetry, while for the other two expressions it is obvious.

Identification of five "1s" in a byte is performed by setting up the following function:

$$(f_2 \& f_3^0 \& g_1 \& g_3^1) + (g_2 \& g_3^0 \& f_1 \& f_3^1) + (\overline{f_1 f_2} \& g_1 \& \overline{g_3^1}) + (\overline{g_1 g_2} \& f_1 \& \overline{f_3^1}).$$

Identification of eight "1s" in a byte is performed by setting to the following function:

$$(f_1 \& f_3^1) + (g_1 \& g_3^1).$$

The theorem is proved.

Taking into account Remark 1, we note that the proved theorem is valid for an arbitrary value of the Hamming distance and an arbitrary bit width of the input vector.

6.3. Summary

The use of model-based control requires the solution of two practical problems: the creation of models and their integration into the control loop. The main challenge is the development of technologies that support implementation, predictive and recovery models.

When choosing the means of software implementation of control E-networks, the emphasis was placed on ensuring the possibility of building distributed systems, including in the Internet networks. Java has been chosen as the basic programming language, which provides cross-platform program execution. A particularly important advantage of Java is the ability to dynamically compile transition functions. This creates conditions for the dynamic change of the program model in the process of its execution.

The developed simulation system is focused on supporting the full life cycle of aggregate implementation models, including the development of conceptual, formalized and software models. This modeling system allows creating new models for the implementation of CA, modifying the existing models and performing statistical experiments with models at the design stage of the CA. The modeling system contains a graphical specification language that provides the construction of models by a user who has no special training in programming. Implementation of distributed properties of the simulation system based on HLA concepts helps speed up the model design process, provides code reusability and increases the performance.

The use of models in the control loop involves the development of a software model interpreter, for the implementation of which it is necessary to perform the task of porting the runtime environment to a microprocessor platform. The synthesis of a multilevel structure on the basis of the cyclic Hamming codes converters is a promising solution for recognizing images using a subset of binary vectors of arbitrary width and Hamming distance as a proximity measure.

Chapter 7. Examples of MOC Applications

7.1. Model-Oriented Control of EBW Machine

7.1.1. Model-Oriented Control Problems of EBW Machine

Electron Beam Welding machines are outstanding representatives of the class of industrial robots (Schulze, 2012), on the basis of which, at the appropriate level of their development and application, an intelligent manufacturing system can be formed.

At present, the most developed areas of industrial use of EBW are aerospace, nuclear power, power engineering, electronics and precision electromechanics. When designing products of complex shapes, designers are increasingly focusing on electron beam welding. EBW is used to create aircraft engine assemblies, rocket bodies, equipment for nuclear power plants, and many other complex products.

The appearance of the EBW machine KL118 designed for welding aerospace structures and examples of such samples are given in Figs. 7.1 and 7.2, respectively (Electron, 2004). This EBW machine has seven controlled coordinates, of which four are controlled simultaneously.

The increased requirements for the quality of control of the EBW machines are explained not only by the complexity of the tasks being solved, but also by the peculiarities of the EBW process itself. Electronic welding, as a rule, is performed automatically according to a predetermined program and consists in the passage of a focused high-power electron beam exactly along the joint line. The prerequisites for performing electronic welding are:

- constant maintenance of high vacuum (about 10^{-5}mm Hg) in the welding chamber;
- smooth multi-axis movement of the electron gun and product;
- stable maintenance of the beam, focusing and bombarding current parameters specified by the program.

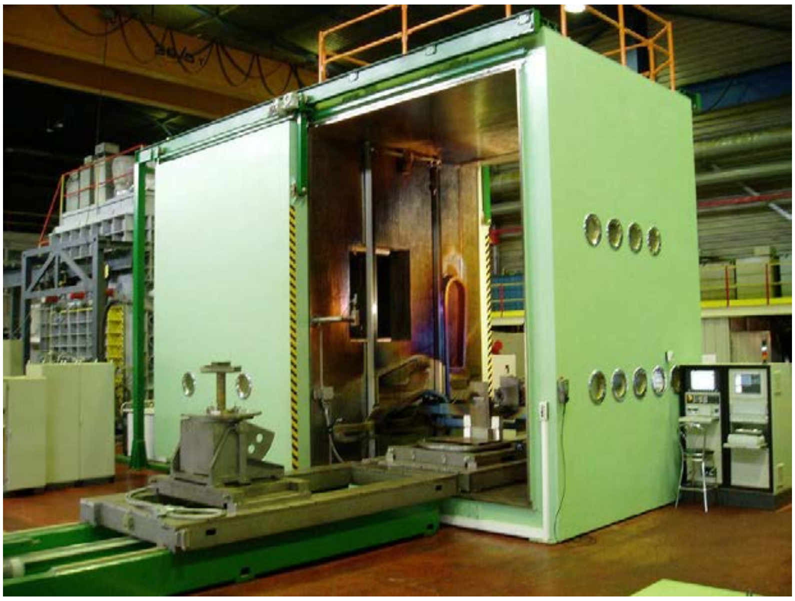


Fig. 7.1 The appearance of the EBW machines type of KL115 and KL118.

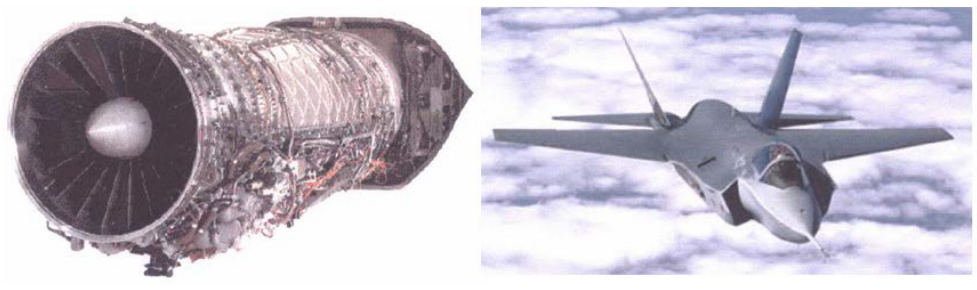


Fig. 7.2. Examples of samples welded in KL115 and KL118 EBW machines.

Each of the listed conditions, in view of the specifics of their implementation, is provided by its own subsystems, which must function as synchronously as possible. Thus, as a CO, an EBW machine is a complex structure, including:

- a vacuum chamber equipped with the necessary vacuum equipment;
- a mechanical system for moving the electron gun and sample inside the vacuum chamber;
- a power source providing the formation of an electron beam of a given power.

During the operation of the installation, all its structural components are influenced by many factors of both external and internal nature. These factors cause deviations of the functioning parameters from the values set by the program.

Most of the factors are stochastic in nature. First of all, this concerns the generated electron beam, where instability is already incorporated at the level of physical processes occurring in the power supply and the electron gun. In addition, during welding under the influence of high temperature, mechanical displacements of the joint position from the programmed beam path are possible, which must be detected and compensated for by the control system. As for the vacuum system, it is necessary to constantly monitor its condition, preventing the development of explosive situations associated with the operation of pumps and valves, as well as an unauthorized drop in vacuum.

In such conditions of the welding process, it is necessary to ensure the deviation of the beam from the joint line at the exit from the sample by no more than 0.1 mm and to prevent interruption of the welding process due to any unforeseen situation, for example, associated with the loss of vacuum in the welding chamber. The fulfillment of this requirement is significantly complicated in the case of using several electron guns as part of one EBW machine.

To the above-mentioned features of the EBW process itself, one should add the limited possibilities of the operator to influence the preparation and performance of welding, which are caused by isolation from the product located in the vacuum chamber. It is possible to control the welding process only according to the indications of measuring instruments. When using EBW machines in serial production, it is necessary to weld several samples at once, placed inside the vacuum chamber, and the operator cannot correct the position of the product or change the program during the welding process. In such cases, all actions to ensure the required quality of welding are assigned exclusively to the control system.

An important point is also adherence to the exact work schedule associated with the need to prepare the product for welding. Usually, the edges of the joint to be welded are pre-cleaned, the product is placed in a special tooling and then installed on the faceplate of the working table (rotator). In a continuous production cycle, all these works must be coordinated with the processes of evacuating air and ventilating the vacuum chamber for product replacement in order to avoid downtime and unexpected production delays.

After welding, the product undergoes final inspection to determine the quality of welding. This is usually done for a prototype by cutting the product at the joint and measuring the parameters of the resulting weld with a microscope. If the seam meets the specified parameters, the welding program is approved for serial use. In the future, all conditions for ensuring the required quality of the output product are assigned to the control system and the production process, which must be strictly adhered to in accordance with the requirements of the quality management system.

Summarizing the above features of the process of operation of EBW machines, we can conclude that the main problem from the point of view of control is the lack of accurate a priori information about the state of the CO and the external environment, including work planning and compliance with the technological conditions for their implementation. This circumstance does not allow using a simple structure of programmed control, since in this way it is possible to track only the execution of a given program of movement along a programmed path. However, it does not solve the problem of exact hitting of the beam into the joint and does not guarantee the required conditions for the welding process, which can change under the influence of external conditions.

In turn, model-oriented control implies precisely the adaptive control of the CO under conditions of uncertainty. In this case, the missing information about the state of the CO is replenished by the models of implementation, forecasting and recovery built into the control loop. Problems that can be solved by applying Model-Oriented Control include:

- automatic situational control of the vacuum system and the power source, carried out using built-in implementation and forecasting models;

- visual design of welding programs with multi-axis movements based on the use of virtual reality display models;

- automatic adaptive control of an electron beam when tracking a joint by using image recognition models;

- multi-agent control of the welding process when using several electron guns in one installation.

7.1.2. Principles of Constructing a Model-Oriented CS of EBW Machines

The application of MOC of EBW machines is based on the observance of a number of principles that should form the basis for the development of a computer control system (Morozov, 2003):

1. *The principle of the hierarchy.* For different subsystems in the EBW machine, the local control goals are clearly different: for the vacuum subsystem, the control goal is to ensure reliable and safe operation; for the power source – to maintain the specified currents, and for the motion control subsystem – to maintain the beam movement at a given speed along a given trajectory. All local targets must be consistent with the systemic control quality criterion, which is expressed in the accuracy of hitting the focused beam at the junction. It means that there is a relationship of functional tasks, which is expressed in the coordinated implementation of the models for the implementation of CA.

2. *The principle of dynamics*, which must meet the requirements for real-time systems. CCS must ensure the synchronous operation of all its subsystems at the rate of change of the CO. For this purpose, all models, including recovery models, must be coordinated in execution cycles with the duration of the interpolation cycle of the movement subsystem of the electron gun and the product, which is usually 2 or 4 msec.

3. *The principle of reliability and safety.* Reliability refers to the ability to complete a welding program satisfactorily within a given period of time, while safety refers to the likelihood that the system will function safely. Since the EBW machine is a potentially dangerous object, it is necessary to evaluate using forecasting models and prevent a potential emergency development of the process before an accident becomes inevitable.

4. *The principle of distribution.* It is an additional condition for ensuring high control reliability. It implies distributed control of various subsystems, due to which each of them is able to independently perform its functions. Compute-intensive display models should not limit the performance of implementation and predictive models.

5. *The principle of flexibility.* It refers to the ability of the control system by the ELS

installation to quickly readjust to perform various welding programs. Compliance with this principle implies the use of a flexible mechanism for creating and editing welding programs, automatic teaching of possible joint paths of arbitrary complexity, adaptation of ready-made programs to the actual location of the product inside the vacuum chamber. This principle is based on recovery models built into the CCS circuit.

6. *The principle of openness.* The EBW machine must operate according to the established production plan, obeying all the requirements of the quality management system. This can be achieved by connecting the EBW machine to project and quality management systems operating at the strategic level of enterprise management. At the same time, using the property of openness, it is possible to implement remote control of individual subsystems at the tactical level.

7.1.3. Hardware and Software Architecture of EBW CCS

The hardware and software architecture of the EBW CCS, which ensures the solution of the problems of the MOC on the above principles, is shown in Fig. 7.3 (Kazymyr, 2006).

The main feature of this distributed architecture is the integrated use of PCs

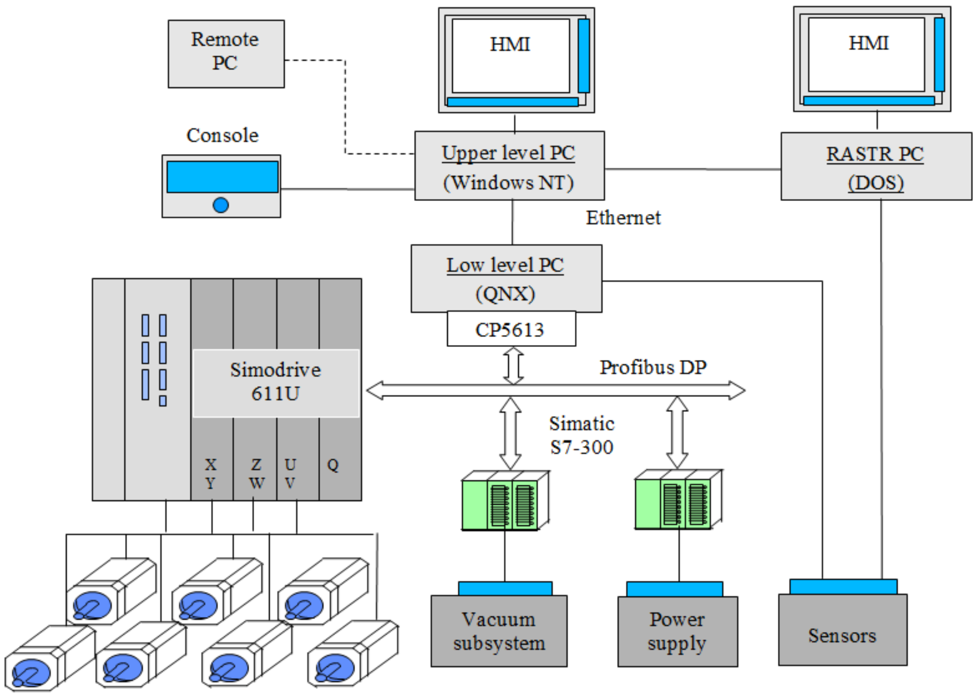


Fig. 7.3. Hardware and software architecture of EBW CCS.

that execute control programs in conjunction with executive devices of automation systems produced by Siemens:

- drive system Simodrive 611 U, which provides direct control of motors;
- Simatic S7-300 interface modules providing transmission and reception of signals from controlled equipment of the vacuum subsystem and power supply;
- communication processor CP 5613, which ensures the use of the industrial bus Profibus DP, through which interaction with actuators is carried out.

Integration of high-quality executive equipment with intelligent control based on built-in models ensures the reliability and efficiency of the EBW machine.

The execution of models built into the control loop is as follows:

- an upper-level PC that operates under Windows NT, implements tactical-level implementation models that coordinate the algorithms for controlling the installation equipment with other systems and models; recovery models are also executed on this PC, in particular virtual reality display models, which are presented in the HMI;
- a low-level PC that operates under the QNX real-time OS (QNX, 2017) executes implementation models and drive level prediction models that control the motion subsystem, vacuum subsystem, and power supply;
- PC of the RASTR system, which operates under DOS control and carries out models of seam image recognition; the image is formed by the observation equipment according to the information coming from the secondary-emission electron sensor installed on the electron gun;
- a remote PC, which can operate under both Windows and Linux operating systems, executes strategic-level models that interact with tactical-level models via the Internet / Intranet.

For the execution of implementation models in the PC of the upper and lower levels, the interpreters of the models described in Subchapter 5.2.1 are built in. Note that the real-time operating system QNX, like Linux, is a representative of the class of Unix-like systems, which facilitated the implementation of the developed software.

7.2. Situational Control of a Vacuum Subsystem Based on the Implementation and Predictive Models

7.2.1. General Characteristics of Vacuum Subsystem Control Process

The control task of the vacuum subsystem (VS) is to create and maintain a vacuum of a given level in the vacuum chamber and the electron beam gun during welding, as well as to ensure the safe operation of vacuum equipment in all modes of its use.

There are four operating modes of the vacuum subsystem: pump down, ventilation, standby and stop. In all modes, control is carried out by checking the status of the equipment and sending control commands through the appropriate interface modules. The VS equipment includes:

1. Pumps for evacuating air from the vacuum chamber. Typically, an aircraft contains one or more fore-vacuum pumps (RPs), a rotary pump (BP), and diffusion pumps (DPs). Diffusion pumps are potentially hazardous because they explode when air enters a heated pump.

2. Turbo-molecular pump (TMP) that provides evacuation of air from the electron beam gun, which takes a significant amount of time to prepare for operation and cannot be stopped instantly.

3. Remote control valves (VE, VM).

4. Vacuum level sensors – vacuum gauges (NV, VV).

Equipment characteristics that must be taken into account:

- time of valve actuation, starting and stopping of pumps;
- permissible vacuum level in the chamber and gun, at which the pumps are turned on (off), valves are opened (closed);
- temperature of DP.

Each of the VC modes has its own set of conditions that guarantee the safe operation of the equipment and ensure the performance of technological operations:

- maximum time for fore-vacuum pumping of air from the vacuum chamber;
- the boundary level of vacuum in the vacuum chamber and the electron gun, at which permission is given for welding;
- the boundary vacuum level in the chamber, at which permission is given to turn

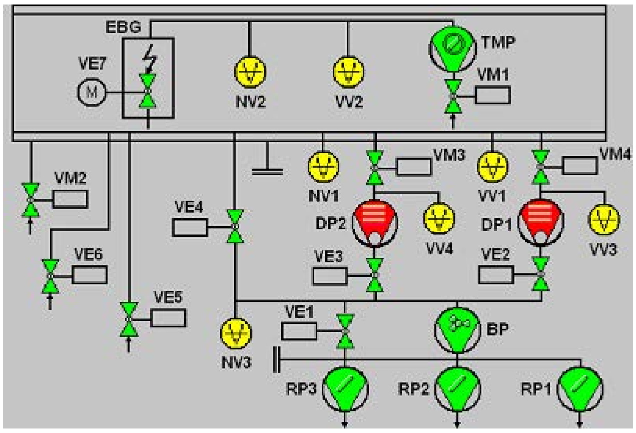


Fig. 7.4. Mnemonic scheme of the vacuum system.

on the rotary pump;

- the limit vacuum level in DP, at which their heating is allowed;
- the limit temperature level in DP, at which their ventilation stops.

VS control is carried out by issuing appropriate signals to switch the states of the vacuum equipment. If the normal operating conditions of the VS are violated, errors and warnings are issued via the operator interface, which are analyzed by the diagnostic system. For example, if the vacuum in the chamber falls below the permissible limit while the pumps are running, a message is displayed about air leakage into the chamber, after which actions are automatically performed to transfer the aircraft to a safe state. The vacuum in the gun and DP is similarly controlled.

In the operator interface, the vacuum system is presented in the form of a mnemonic diagram (Fig. 7.4), on which the current state of the vacuum equipment is displayed using the corresponding color symbols.

7.2.2. Implementation Model of VS Control Algorithm

The VS control algorithm in the form of an implementation model built using the EMS modeling system is shown in Fig. 7.5.

The control of the PUMPs is transferred from the upper-level program after the operator selects the pumping mode in the graphical interface. In this case, the token is placed in position P1, after which the execution of the model begins.

In the model, the transition functions have the definition given below.

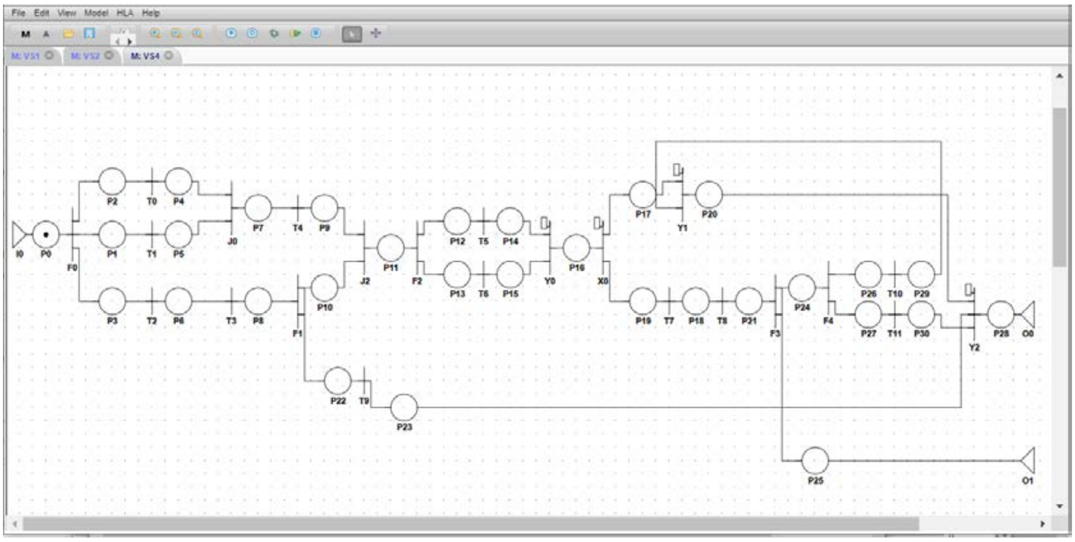


Fig. 7.5 Implementation model of VS control program.

Transition F0: initiates the pump down process.

Transition T0:

• activation function

{RP1 == 1 & RP2 == 0 & RP3 == 1;} – the transition is activated when all RP pumps are running;

• conversion function

{VE1 = 1; VE4 = 1;} – valves are opened through which air is pumped out of the vacuum chamber.

Transition T1:

• conversion function

{RP1 == 1 & RP2 == 0 & RP3 == 1;} – RP pumps are started.

Transition T2:

• activation function

{RP1 == 1 & RP2 == 1 & RP3 == 1 & VE1 == 1;} – the transition is activated when all RP pumps are running and the valve for pumping air from DP is open;

• conversion function

{VE2 = 1; VE3 = 1;} – air pumping from DP starts.

Transition J5:

• activation function

{NV3 == 1;} – the transition is activated when the vacuum in the chamber reaches a level sufficient to start the BP pump;

Transition T4:

• conversion function

{BP = 1;} – the BP pump starts up.

Transition T3:

• activation function

{VV3 == 1 & VV4 == 1;} – the vacuum in the diffusion pumps has reached the specified limit;

• conversion function

{DP1 == 1; DP2 == 1;} – DP pumps are switched on for heating.

Transition J2:

• activation function

{NV1 == 1;} – the vacuum in the chamber has reached the upper limit;

• conversion function

{VM3 = 1; VM4 = 1;} – DP pumps are connected to pumping air from the chamber.

Transition T9:

• activation function

{VV3 == 0 || VV4 == 0;} – there is no vacuum in any of the DP pumps;

• conversion function

{DP leaking = 1;} – issuing the signal “DP leaking”.

Transition T5:

• delay function

{TimeT12 = 1200000;} – a delay is set for the foreline pumping time of 20 minutes.

Transition T6:

• activation function

{VV1 == 1;} – the lower vacuum limit in the chamber is reached;

Transition X0:

• decisive function

{if (P16.M [1] == 1) return 1; else return 2; } – the choice of the direction of the process development: 1 – the vacuum has not reached the lower limit for the given fore-vacuum pumping time, otherwise – the fore-vacuum pumping was successful.

Transition Y1:

• conversion function

{Chamber leaking = 1;} – issuing the signal “Leaking into the chamber”.

Transition T7:

• conversion function

{VM1 = 1; TMP = 1;} – air pumping from the gun begins.

Transition T8:

• activation function

{VV2 == 1;} – the vacuum in the gun has reached the lower limit;

• conversion function

{ready = 1; } – a signal of readiness for welding is issued.

Transition T10:

• activation function

{NV2 == 1; VV2 == 0;} – the vacuum in the gun fell below the lower limit;

• delay function

{TimeT10 = 60000;} – a 1-minute delay;

• conversion function

{Gunr leaking = 1;} – issuing the signal "Leaking into the gun".

Transition Y2:

• conversion function

{ready = 0; } – removing the readiness for welding.

The normal operating conditions of the VS, which are checked using the predictive models, are determined with DCTL as follows:

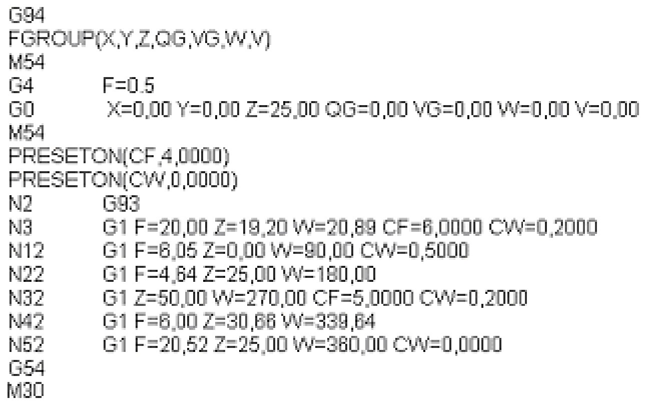
- AG ((RP1 and RP2 and RP3) {> 1200000} implies VV1) – always after 20 minutes of fore-vacuum pumping, the vacuum in the chamber must be above the lower limit;
- EF ((READY and not VV2 {> 60000}) implies CHAMBER_LEAKING) – if at any time, when ready to weld, the vacuum in the chamber remains below the upper limit for more than 1 minute, a "Leak into chamber" signal should be generated.

7.3. Visual Design of Welding Programs Based on Recovery Models

7.3.1. Formulation of the Problem

Usually, in EBW machines, the welding program is carried out using devices such as CNC (Computer Numerical Control). CNC programs are written in G-codes (Smid, 2007). The welding program in G-codes (Fig. 7.6) is a sequence of blocks (lines of code), in which the coordinates of the point to which you want to move are set for each segment of the path.

In addition to coordinates, the block specifies the speed of movement, the method and parameters of interpolation, as well as the values of welding current (CW), focusing current (CF) and beam deflections, set when the end point of the programmed path segment is reached.



```
G94
FGROUP(X,Y,Z,QG,VG,W,V)
M54
G4          F=0.5
G0           X=0,00 Y=0,00 Z=25,00 QG=0,00 VG=0,00 W=0,00 V=0,00
M54
PRESETON(CF,4,0000)
PRESETON(CW,0,0000)
N2          G93
N3          G1 F=20,00 Z=19,20 W=20,89 CF=6,0000 CW=0,2000
N12         G1 F=6,05 Z=0,00 W=90,00 CW=0,5000
N22         G1 F=4,64 Z=25,00 W=180,00
N32         G1 Z=50,00 W=270,00 CF=5,0000 CW=0,2000
N42         G1 F=6,00 Z=30,66 W=339,64
N52         G1 F=20,52 Z=25,00 W=360,00 CW=0,0000
G54
M30
```

Fig. 7.6. Fragment of a welding program in G-codes.

In the case of simultaneous use of several coordinates, when the final trajectory is a complex spatial curve, the traditional procedure involves the development of a motion program using a product drawing built by means of a CAD/CAM system. The duration of the preparation of such a program can be, depending on the complexity of the trajectory, from several hours to several days or even weeks.

It should also be taken into account that the operator-welder will spend additional time to adapt the pre-designed program directly at the installation to the real product, taking into account the inaccuracy of its manufacture and placement at the welding position. If such an adjustment is quite simple when using only linear coordinates (X, Y, Z), then when the trajectory is formed by simultaneous linear and angular displacements, the adaptation becomes extremely difficult.

The use of recovery models built into the control loop eliminates the traditional G-code welding programming and allows for visual design of multi-axis welding programs. The developed method of visual design assumes a sequential solution of the following tasks:

- construction of a three-dimensional virtual representation of the situation inside the vacuum chamber;
- automatic learning of the displacement system to follow the path of the joint;
- graphical representation of the recorded butt path for drawing up a welding program.

In the process of visual design, a synthetic EBW CCS environment is formed (Kazymyr, 2003), and then it is used by operators to control the installation.

7.3.2. Virtual Representation of the Sample and Welding Path

The operator needs a virtual three-dimensional representation of the product inside the vacuum chamber to select the key points and set the correct angular orientation of the gun during visual trajectory design.

The developed toolkit of virtual reality display models allows, in particular:

- creating three-dimensional images of sample;
- displaying the actual location of the sample and the gun inside the vacuum chamber;
- changing the scale and view of the generated display for its more convenient and detailed presentation;
- displaying the location of the programmed welding path on the surface of the sample;
- setting the permissible areas of movement of the gun and carrying out automatic control of movements in order to prevent damage to the sample and equipment inside the chamber;
- tracking on the surface of the product the position of the electron beam relative to the specified trajectory during welding.

With the help of display models, models of structures of the highest complexity level can be created (Paton, 2004).

Table 7.1 shows complex structures, when welding, which, taking into account the need to ensure the perpendicularity of the beam to the surface of the sample, requires at least four simultaneously controlled axes.

Figure 7.7 shows the designations of the coordinate axes of welding movements, the combination of which allows welding such products.

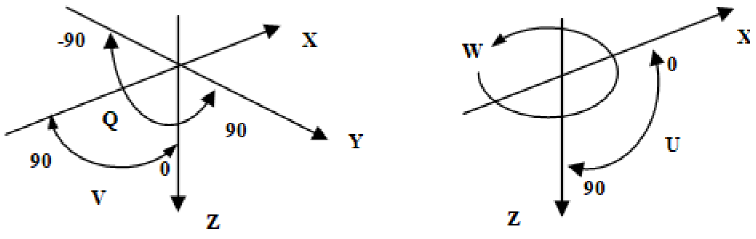
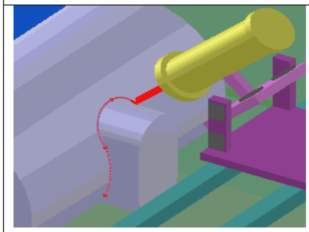
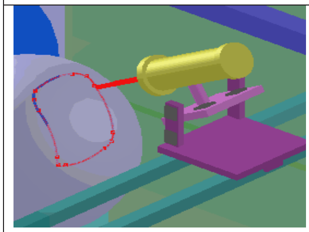
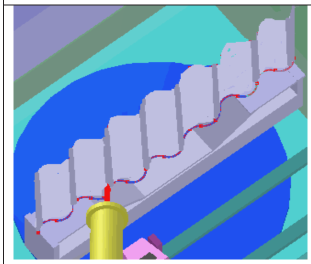


Fig. 7.7. Coordinate axis system.

Table 7.1 Types of Samples and Required Movements

Sample	Joint description	Required movements
	Oval-shaped branch pipe is welded into an extended thick-walled cylinder	Sample rotation, gun movement along two linear axes and its tilt
	Welding a rectangular segment with rounded corners into the surface of the ball	Gun tilt or turn, gun movement along three linear axes
	Welding the corrugated sheet to the trapezium base	Rotation and movement of the gun along three linear axes

In Fig. 7.7, the following designations are adopted: *X, Y, Z* – linear axes of gun movement; *Q* – cannon rotation axis; *V* – cannon tilt axis; *W* – rotation axis of the table faceplate, *U* – table tilt axis.

Sample images are formed by the operator from a basic set of auto shapes and displayed on the monitor in three-dimensional space (x, y, z). To display in the same three-dimensional space the trajectory of the beam on the surface of the sample formed as a result of welding movement in multidimensional space (X, Y, Z, V, Q, W, U), it is necessary to transform the seven-coordinate vectors describing the points of the trajectory into three-dimensional vectors of spatial display.

For this purpose, to take into account the angles of rotation and tilt of the gun, we find an additional vector

$$(X, Y, Z, V, Q, W, U), \tag{7.1}$$

where rX – the coordinate distance from the X beam axis to the gun tilt axis;

rZ – the distance from the axis of rotation of the gun to its end;

D – the distance from the end of the gun to the surface of the product.

Next, for each point, we rotate the vector A around the axis of rotation of the gun in the plane YZ by an angle Q, and then – around the axis of inclination of the gun in the plane XZ by an angle V. As a result, we get the coordinates of the point (x_1, y_1, z_1) relative to the center of rotation and tilt of the gun.

To take into account the angles of rotation and tilt of the table, we find an additional vector

$$B = (X_{rot}, Y_{rot}, Z_{rot} + dZ_{rot}), \tag{7.2}$$

where X_{rot}, Y_{rot}, Z_{rot} – coordinates of the center of the table in the base coordinate system;

dZ_{rot} – the distance from the axis of inclination of the table to its surface.

We move the center of coordinates to the end of the vector B and for each point rotate the vector (X, Y, Z) around the axis of rotation of the table in the plane XY by an angle W. Let us get a point (x_2, y_2, z_2). Then we will rotate the same vector around the table tilt axis in the plane XZ by an angle U. Let us get a point (x_3, y_3, z_3).

Ultimately, the base coordinates of the points B, according to which the points will be displayed inside the vacuum chamber, will be recalculated according to the following expression:

$$(x, y, z) = (X + X_{rot} + x_1 + x_2 + x_3, Y + Y_{rot} + y_1 + y_2 + y_3, Z + Z_{rot} + z_1 + z_2 + z_3). \tag{7.3}$$

Figure 7.8 shows the resulting three-dimensional image of the product and the trajectory for different viewing angles.

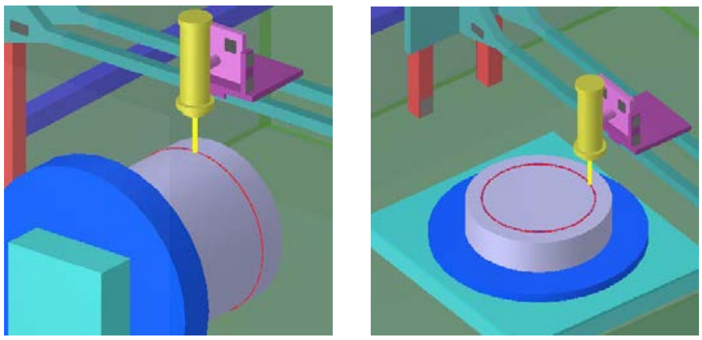


Fig. 7.8. 3D sample and joint path views.

7.3.3. Automatic Learning of the Joint Path

The joint path, which is displayed in a 3D virtual representation of the sample, forms the basis for constructing a welding program. First, it sets the program for multi-axis movements of the gun and the item. Second, the program for changing the welding currents must subsequently be linked to this trajectory.

When designing simple, for example, linear joints, the operator can specify the path in the form of a table, identifying the coordinates of the start and end points of movement. However, this method of programming trajectories becomes unacceptable in the case of complex spatial joints with an arbitrary trajectory. The problem is solved by creating a method for automatic trajectory learning based on the use of recovery models.

To obtain an image of a joint, the complex of developed models of image recognition is used (Kazymyr, 2006). The joint in the resulting image (Fig. 7.9) appears as a dark line against the background of the lighter surface of the product. The red cross indicates the location of the center of the electron beam.

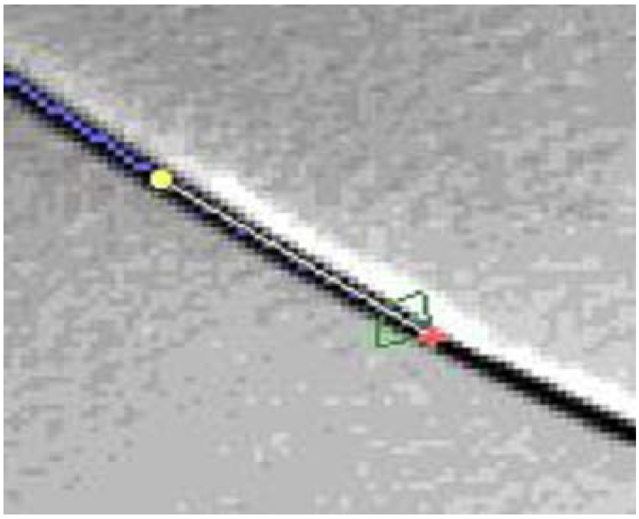


Fig. 7.9. Image of a joint during automatic trajectory learning.

At the beginning of training, the operator, manually moving the gun or the sample, aligns the electron beam with the starting point of the joint of the edges being welded, sets the initial direction and speed of movement, and issues a command to start the movement. Further, the learning process occurs completely automatically.

Automatic learning is carried out in the process of joint work of recovery models and implementation models of the executive level. The diagram of the hardware and software complex as part of the control system, which provides automatic training of

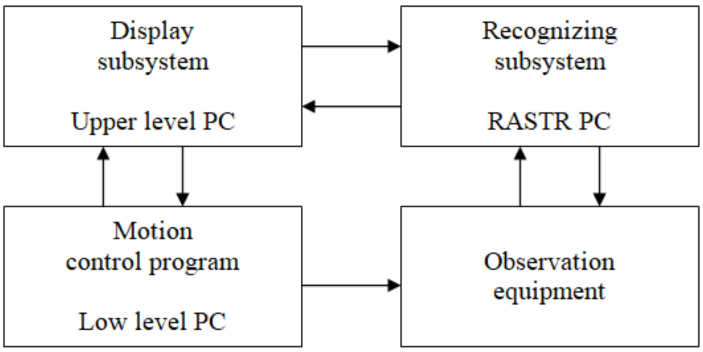


Fig. 7.10. The scheme of the hardware and software complex for automatic joint trajectory learning.

the joint trajectory, is shown in Fig. 7.10.

Automatic learning is performed based on the following assumptions:

- in the initial position, the beam is set in the middle of the joint;
- the initial direction of the joint search is set;
- the joint has a continuous (homogeneous) structure.

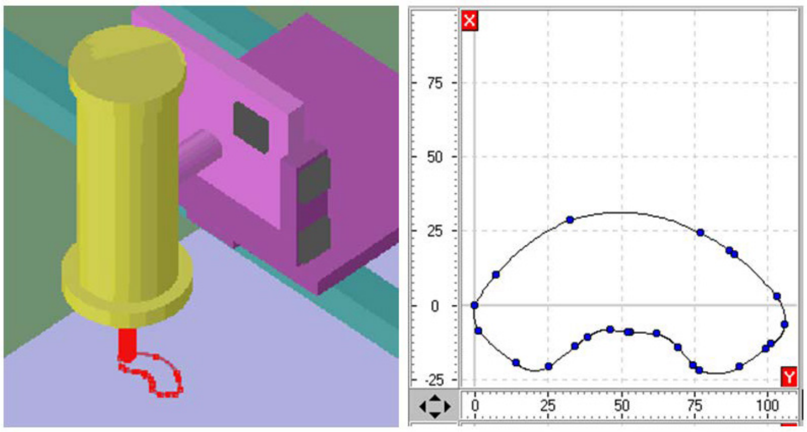
Calibration is a prerequisite for automatic trajectory learning. This operation is performed before the start of the training procedure, and its end result is to establish the correspondence of the sizes of one pixel to the linear dimensions of the image in millimeters.

In the process of movement, the recognition program finds a joint in each new image frame and determines a vector of displacement to a new trajectory point located in the middle of the joint. The process of automatic learning of the path is performed cyclically with stepwise linear movement to a new joint point. The choice of the next point from the joint area is carried out on the basis of obtaining piecewise-linear interpolation of the joint path with a possible deviation from the middle of the joint by no more than 0.1 mm.

When moving during training, the relative position of the electron gun and the product is constantly monitored. The coordinates of the displacement point are recalculated according to Eqs. (7.1)–(7.3) and checked for falling into the admissible area, which is set when forming a three-dimensional image of the product. The learning algorithm stops if the next found point coincides with the previous point or with the starting point of the trajectory (for a closed trajectory), or when the next point goes beyond the permissible displacement range.

7.3.4. Graphical Representation of the Joint Path

During training, the resulting seam path is displayed on a 3D virtual representation of the product and on reference planes, as shown in Fig. 7.11.



a) b)

Fig. 7.11. Display of joint path during automatic learning: a) representation of the trajectory on a three-dimensional virtual representation of the sample; b) representation of the trajectory on the base plane.

As path points are found, they are connected to the previous points by straight line segments. A piecewise-linear model of a real joint obtained in this way does not take into account its geometric features and contains an excessive number of points (the greater the curvatures of the joint, the more points are required for its piecewise-linear approximation). The number of trajectory points can be reduced by carrying out additional approximation at the command of the operator using the trajectory reconstruction models. After approximation, the trajectory is represented by segments of straight and circular lines, for which the corresponding parameters are automatically calculated. In any case, the accuracy of the resulting trajectory does not go beyond the specified criterion – 0.1 mm.

In addition to performing additional approximation, the operator can edit the resulting trajectory manually, changing the coordinates of points, interpolation methods (linear or circular) on selected sections of the trajectory, adding new points to an existing trajectory or removing some points from the resulting trajectory. As the point parameters, the operator also sets and edits all technological welding parameters, including beam and focusing currents, the speed of welding movement between points, etc.

7.4. Adaptive Control of the Electron Beam Position in Joint Tracking

7.4.1. Formulation of the Problem

Joint tracking is performed with the aim of precisely keeping the center of the beam in the middle of the butt during welding, when due to the emerging welding deformations, the joint may shift away from the specified path of movement. The beam is aligned with the joint by deflecting the beam along the X and Y axes in the plane of the gun and by an angle calculated by the control program.

The calculation of the deflection parameters is performed in such a way as to ensure the accuracy of keeping the beam in the middle of the joint within 0.1 mm. To achieve the specified accuracy, the coordinates of the middle of the joint are always determined taking into account the linear dimensions of the image pixel.

The problem is that during the welding process, a pool is formed at the location of the beam, which melts the joint, as a result of which it is impossible to determine the deviation of the beam from the middle of the joint at the welding location. In this regard, the problem arises of tracking the joint at the lead-in point, which is moved forward along the path of the joint at a given distance. In the case of arbitrary curved joints, the calculation of the lead-in point can only be performed using restoration models.

In this setting, we will have an adaptive control problem with a reference model, which we will consider the joint trajectory restored in the process of automatic learning. The management quality indicator will be determined by the expression

$$J = \min_{\gamma \in \Omega(\gamma)} \sqrt{(dx^2 + dy^2)}, \qquad (7.4)$$

where dx – the deviation of the middle of the joint from the position of the center of the beam along the X axis;

dy – the deviation of the middle of the joint from the position of the center of the beam along the Y axis;

Υ – control (beam deflection angle) from the area of permissible deviations $\Omega(\gamma)$.

7.4.2. Recovery Models Used in Tracking

When tracking the joint, the models of the recognition subsystem and the constructed trajectory model in the form of a graphical representation are used. Figure 7.12 shows a view of the image of the seam during tracking.

The frame outlines the joint search zone, which is located perpendicular to the direction of movement at a given lead distance. The current position of the deflected beam at the place of the proposed welding is marked with a red cross.

Additionally, the joint image displays:

- current position of the beam that is not deflected (current point of the trajectory) – light cross at the location of the weld pool;
- the position of the middle of the joint at the lead distance detected by the recognition program – a dark point;
- the estimated position of the middle of the joint at the lead distance – a light cross next to a dark point;
- permissible beam deflection zone – middle part of the search zone.

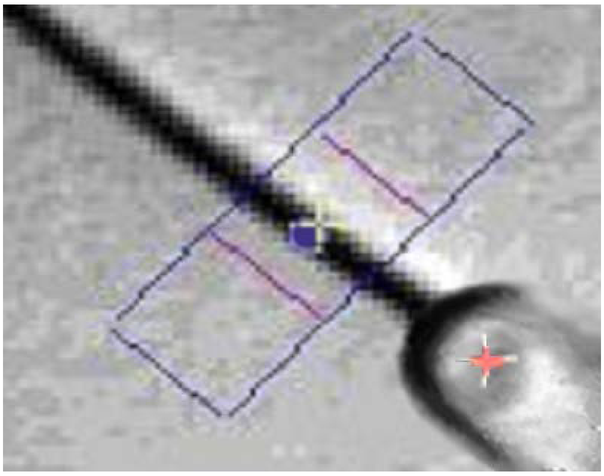


Fig. 7.12. Joint view during tracking.

The search area is built perpendicular to the direction of the velocity vector at the lead-in point. By the position of the cross and the point in the joint search zone, the required deviations of the beam along the X and Y coordinates are determined to accurately hold the beam in the middle of the joint. This calculation can be performed only when the motion control program and the program processing the recovery models work together synchronously. Moreover, the accuracy of the calculation will largely be determined by the accuracy of the graphical representation of the joint trajectory. In this regard, it is necessary to track the joint only after automatic learning of its trajectory.

7.4.3. Tracking Algorithm

The initial premise for the tracking algorithm is the calculation of the lead-in points of the joint path exactly at the moments of receiving images from the observation equipment. With a typical operating cycle for RASTR equipment of 300 msec, the points corresponding to the anticipated position of the beam at the moments of image acquisition should be calculated along the recorded path of the joint, i.e., the trajectory should be divided into sections in exact accordance with the period of operation of the RASTR equipment.

To coordinate the operation of the equipment and the motion control program, the observation system is launched into operation at the command of the lower-level PC at the moment the movement begins. Beforehand, the entire array of trajectory points corresponding to the received images is transferred to the recognition subsystem. The coordinates of the transmitted points should be projected onto the coordinate axis of the sensor installed at the end of the gun, with the origin at the point where the beam is located.

Taking into account the clarifications made, the tracking algorithm includes the following steps:

A. Stage of preparation

1. Divide the entire trajectory into sections with a length $l = \tau_R \cdot v_b$ that equals the duration of the observation equipment operation cycle τ_R, multiplied by the specified travel speed v_b. Let us get an array of n seven-coordinate points $P=(p_0, p_1,..., p_{n-1}), p_i = (X_i, Y_i, Z_i, V_i, Q_i, W_i, U_i), i = \overline{0, n-1}$.

2. For each point $p_i \in P, i = \overline{0, n-1}$, calculate the lead-in point $p' \in P'$ located at a given distance $p \in P$ from the point Δ.

3. Using vectors (7.1)–(7.2) and shifting the center of coordinates, project all points p_i' onto the end of the gun. Let us get points with coordinates (x_i', y_i'), assuming that for points p_i such coordinates are equal to zero: $x_i = 0, y_i = 0$.

4. For each leading point p_i' determine the direction α_i of the displacement vector to the next point.

5. Transfer the resulting list $\{(x_i', y_i', \alpha_i)\}, i = \overline{0, n-1}$ of displacements and directions of movement at the predicted point for processing to the recognition subsystem.

B. Tracking stage (performed sequentially for each image frame and each point in the array)

1. Determine the real coordinates of the midpoint of the joint in the place of the lead-in point $p_c' = (x_c', y_c')$.

2. Calculate the deviation of the calculated lead point from the middle of the joint $dp = (dx, dy) = (x_c^{'} - x^{'}, y_c^{'} - y^{'})$.

3. Make the deflection of the beam relative to the current deviation of its axis $\gamma = (\gamma_x, \gamma_y)$ by an angle $(d\gamma_x, d\gamma_y) = (arctg(dx/w), arctg(dy/w))$, where w is the working distance (the distance from the end of the gun to the surface of the sample). The specified deviation vector $d\gamma = (d\gamma_x, d\gamma_y)$ should be processed by linear increments in each interpolation cycle (IPO cycle) by the value $\delta\gamma_x = \frac{d\gamma_x}{\tau_{IPO}}$ and $\delta\gamma_y = \frac{d\gamma_y}{\tau_{IPO}}$, respectively, where τ_{IPO} is the duration of the IPO cycle.

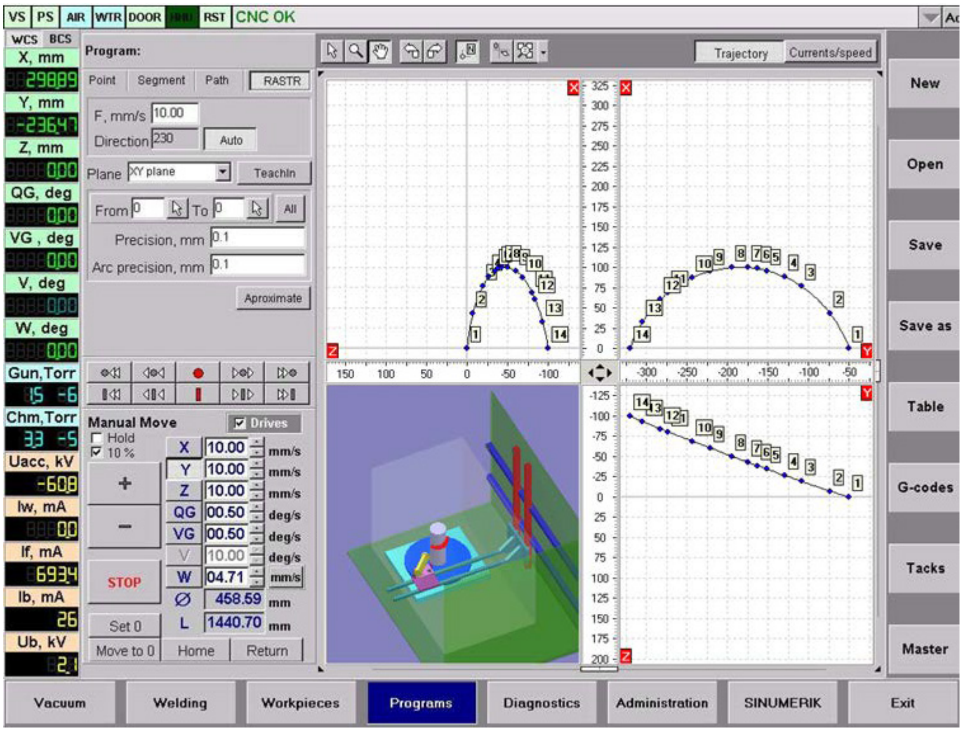
Thus, as a result of executing the tracking algorithm, we get the following implementation of adaptive control operators:

1. Operator of the main control loop:

$$dp(k+1) = H_1[dp(k), p_c^{'}(k), d\gamma(k)], \tag{7.5}$$

where $dp(k+1)$ – the vector of deviations of the beam center at the lead-in point at the next step;

$dp(k)$ – the vector of deviations of the center of the beam at the lead-in point at


VS PS AIR WTR DOOR RST CNC OK

WCS BCS

X, mm -29889

Y, mm -23647

Z, mm 000

QG, deg 000

VG, deg 000

V, deg 000

W, deg 000

Gun,Torr 15 -6

Chm,Torr 33 -5

Uacc, kV -60.8

Iw, mA 00

If, mA 6934

Ib, mA 26

Ub, kV 2.1

Program:

F, mm/s 10.00

Direction 230

Plane XY plane

From 0 To 0

Precision, mm 0.1

Arc precision, mm 0.1

Manual Move

Hold

☑ 10 %

☑ Drives

X 10.00 mm/s

Y 10.00 mm/s

Z 10.00 mm/s

QG 00.50 deg/s

VG 00.50 deg/s

V 10.00 deg/s

W 04.71 mm/s

Ø 458.59 mm

L 1440.70 mm

Fig.7.13. Operator interface screen view during trajectory learning.

the current step;

$p_c^{'}(k)$ – the vector of coordinates of the current position of the joint midpoint at the lead-in point;

$d\gamma(k)$ – the current deflection angle of the beam.

2. Operator adaptation:

$$p_c^{'}(k+1) = H_2[dp(k), p_c^{'}(k), d\gamma(k)]. \tag{7.6}$$

Note that the information about the process contained in the vector $p_c^{'}(k)$ is formed by using reconstruction models, namely, joint recognition models that take into account the current deflection of the beam. The control vector $d\gamma = (d\gamma_x, d\gamma_y)$ is recalculated at each step of the algorithm.

Figure 7.13 shows the operator interface of the KL115 and KL118 EBW machines during the performance of trajectory learning.

7.5. Multi-Agent Control of the Simultaneous Operation of Several Electron Beam Guns

7.5.1. Formulation of the Problem

The unique EBW machines KL 117, designed for welding of drill bits, use three electron beam guns, which should weld the three joints simultaneously. Guns are permanently installed and arranged in a circle at a distance of 120 degrees from each other, so their beams are focused on three joints. The appearance of the KL117 machine and samples of bits are shown in Figs. 7.14 and 7.15, respectively.

For obtaining a quality weld in this EBW machine, there is a need for stable operation of the parameters of the welding current and focusing current according to the program. These conditions provide the most efficient use of the energy capacity of the welding machine, maintaining a safe mode of operation.

Each electron gun is controlled by its module of computerized CS, but the program is the same for all guns. It is compiled for one base joint and includes values of beam speed, welding current and focusing current, which are determined by the profile of the workpiece. The problem that arises during the welding process is a discrepancy in the physical parameters of the electron guns. This is due to uncontrollable variations in the electrical equipment and uneven wear cathodes. For this reason, the predetermined value of the focusing current is fulfilled in each of the guns at its level, and the resulting welds differ in quality, which is unacceptable. The challenge is that, using the embedded computer models of physical processes in the guns to provide agreed job of them, it is possible to obtain the same quality of beam despite

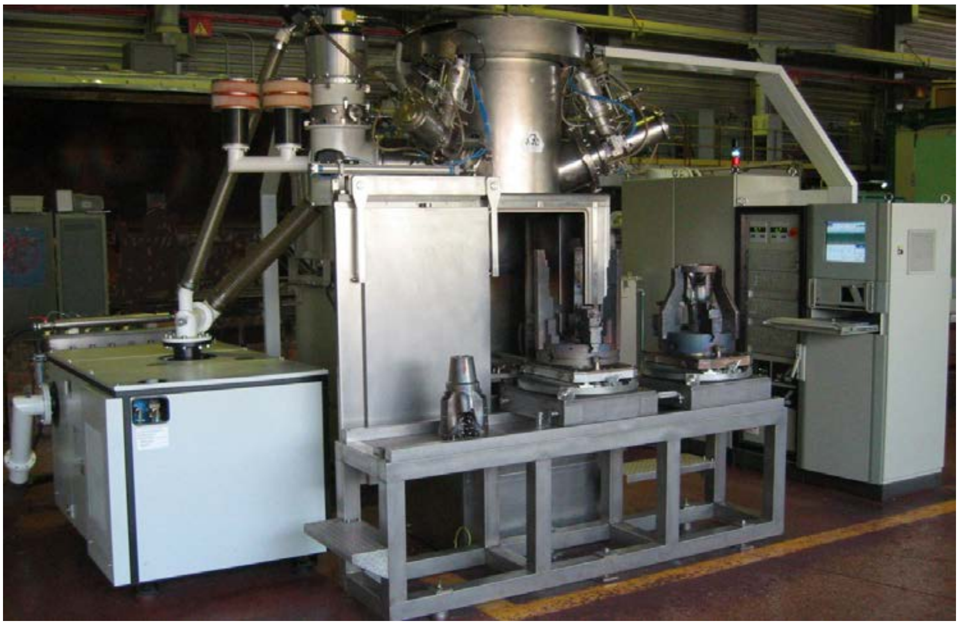


Fig. 7.14. The appearance of the KL117 machine.

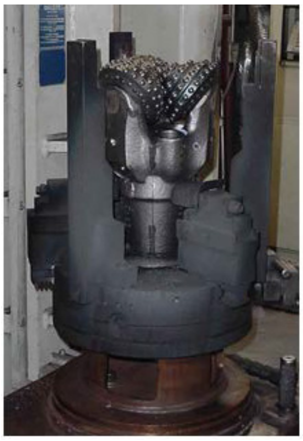
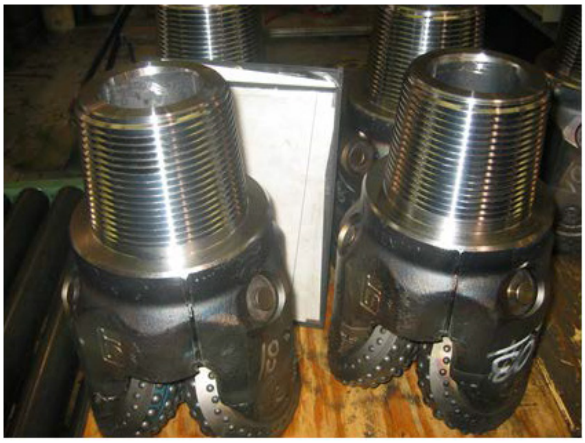



Fig. 7.15. The samples of bits.

the variations in the parameters of the guns.

Obtaining high quality welds with simultaneous operation of three electron guns within a single EBW machine can be achieved by applying the method of software control using intelligent agents.

7.5.2. Construction of Recovery Models

To get background information about a control object, there may be used

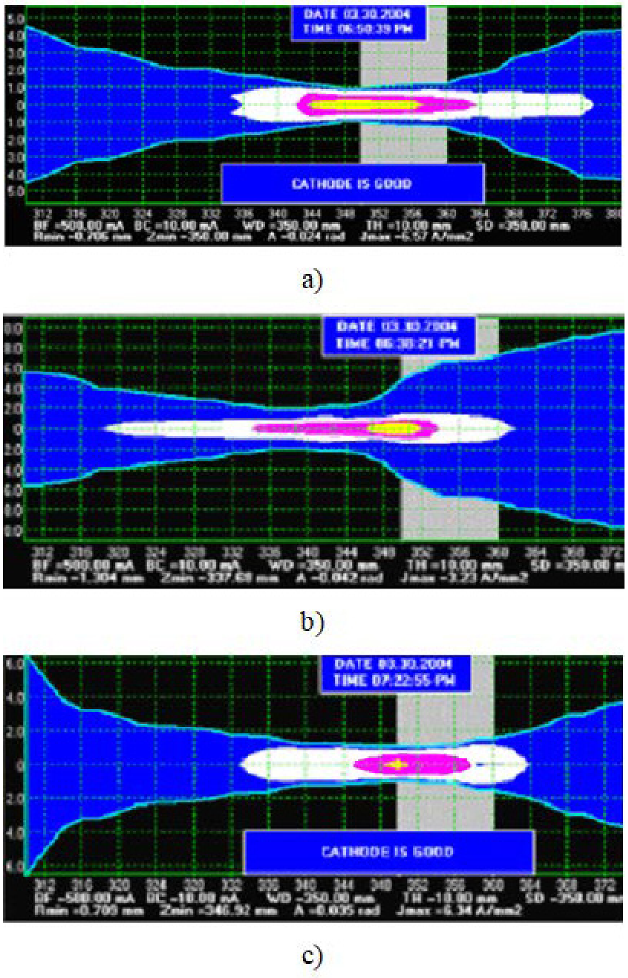

DATE 03.30.2004
TIME 06:50:39 PM
CATHODE IS GOOD
BF =500.00 mA BC =10.00 mA WD =350.00 mm TH =10.00 mm SD =350.00 mm
Rmin =0.706 mm Zmin =350.00 mm A =0.024 rad Jmax =6.57 A/mm2
a)
DATE 03.30.2004
TIME 06:38:21 PM
BF =500.00 mA BC =10.00 mA WD =350.00 mm TH =10.00 mm SD =350.00 mm
Rmin =1.304 mm Zmin =337.68 mm A =0.042 rad Jmax =3.23 A/mm2
b)
DATE 03.30.2004
TIME 07:22:55 PM
CATHODE IS GOOD
BF =500.00 mA BC =10.00 mA WD =350.00 mm TH =10.00 mm SD =350.00 mm
Rmin =0.709 mm Zmin =346.92 mm A =0.035 rad Jmax =6.34 A/mm2
c)

Fig. 7.16. Density beam profile models of the welding current:
a) gun 1; b) gun 2; c) gun 3.

the model of the current density profile obtained by means of diagnostic systems proposed by Akopyants et al. (2002). For three different guns within a single EBW machine, a density beam profile model of the welding current can be prepared (see Fig. 7.16).

The principle of this system is based on measuring the power distribution in the electron beam when the focusing current is changed. It uses a special sensor that intersects the beam at different values of the focusing current. As a result, the current density distribution is built from n cross-sections of the beam. Within each k-th section, the current density J_k is assumed to be constant. In order to find unknown J_k there is built a system of linear algebraic equations of the form

$$S \cdot J = A, \quad (7.7)$$

where S – the matrix of dimension $n \times n$ with the elements proportional to cross-sectional area;

J – n -dimensional vector of the beam density values;

A – n -dimensional vector of pulse amplitudes at each section.

According to the found values $J_k, k = \overline{1, n}$ the current density distribution is constructed for each cross-section of the beam. The cross-section with the highest current density on the axis J_{max} corresponds to the sharp focusing of the beam. For each distribution, the effective beam radius R is calculated on the assumption that the current density distribution can be approximated by the normal distribution law. Within this radius 63 % of the beam power extends. Thus, the sharp focus with the highest current density corresponds to the smallest effective radius of the beam R_{min}.

The current density distribution along the beam axis displays four areas: 10–25 %, 25–50 %, 50–75 % and 75–100% from the maximum density. These charts are used as recovery models of physical processes in the guns.

7.5.3. Theoretical Background of Control Method

A scheme explaining the calculation used in the control algorithm is shown in Fig. 7.17. Designations used:

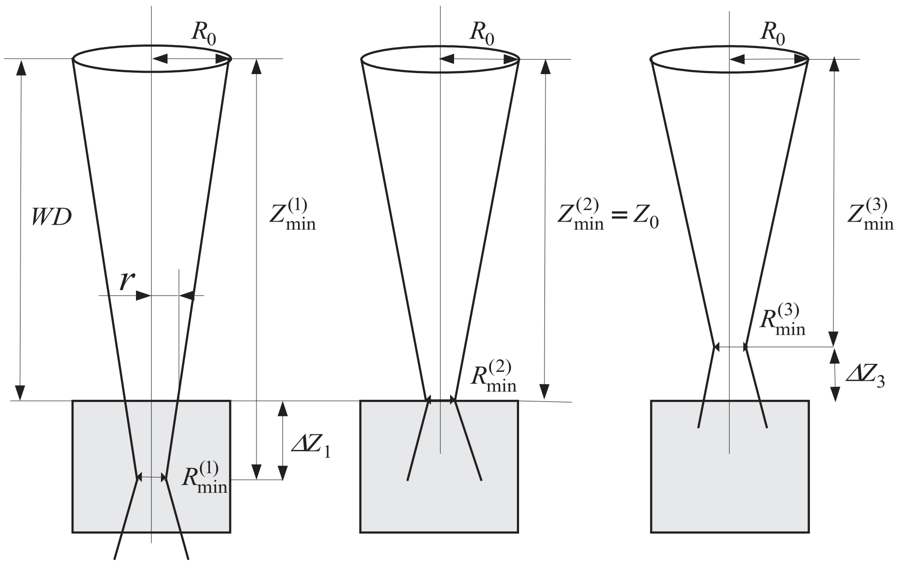


Fig. 7.17. Scheme of calculation in the multi-agent control algorithm.

- R_0 – the beam radius in the plane of the focusing lens;
- $R_{min}^{(i)}$ – the smallest effective radius of the i -th gun;
- $Z_{min}^{(i)}$ – the distance to the maximum zone of density beam of i -th gun;
- WD – the distance to the surface of the workpiece (working distance);
- ΔZ_i – removing distance of maximum density beam gun from the surface of i -th workpiece;
- r – the beam radius on the workpiece.

When there is a mismatch of $Z_{min}^{(i)}$ it is necessary to bring it to a common measure in order to provide the work of all the guns by the welding program, for example, in sharp focus. For this purpose, the software will calculate the correction for the value of the focusing current I_f by the formula

$$\Delta I_f^{(i)} = \Delta Z^{(i)} \cdot K_1^{(i)} \cdot K_2^{(i)}, \tag{7.8}$$

where $\Delta Z^{(i)} = Z_{min}^{(i)_0}$;

$K_1^{(i)}$ – the coefficient that takes into account the change of focusing distance $Z_{min}^{(i)}$;

$K_2^{(i)}$ – the coefficient that takes into account the change of focusing distance during the change of focusing current.

At the low values of the ray convergence angles, the coefficient $K_1^{(i)}$ is determined by the formula:

$$K_1^{(i)} = \frac{R_0 - R_{min}^{(i)}}{R_{min}^{(i)}}. \tag{7.9}$$

The coefficient $K_2^{(i)}$ for each gun is estimated by a special procedure when a cathode is changed. It is used in the calculation as the initial parameter. It should be noted that when the focusing current increases the value $Z_{min}^{(i)}$ decreases, i.e., always $K_1^{(i)} < 0$.

7.5.4. Implementation Model of Control Algorithm

In the described method of the current focus matching one control module for every gun is used. It operates on the principles of intelligent agents. Each agent uses an implementation model represented in the form of CEN. An example of such an agent model is shown in Fig. 7.18.

1. At the cooperative level the task of information exchange between agents, which

control guns, is solved in order to determine the agreed level of focus distance. Input data are received in the form of tokens from adjacent agents via the input boundary positions *P0–P3*. In particular, the positions *P0* and *P1* receive tokens containing two attributes (N, NA), where N is the number of the agent (the gun), and NA – a decision accepted by the agent about the number of the leading guns (to the focus level of this gun all other agents will lead their focusing current). Positions *P2* and *P3* get tokens that contain a set of attributes (N, DF), where DF means deviation required for a change in the focusing current that is calculated by the agent. Calculated cooperative knowledge is compared at transitions *J0* and *J1*, then the results are transmitted to the planning level of agent.

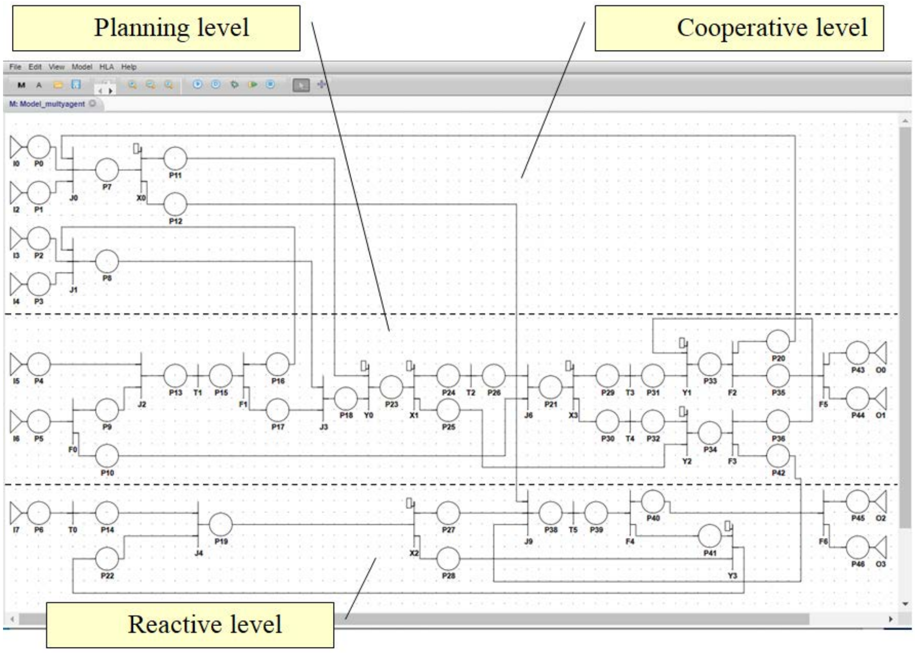


Fig. 7.18. CEN implementation model of an agent control algorithm.

The agent model shown in Fig. 7.18 defines three levels of focusing current control for one gun.

There are two possible options of coordination:

- all agents make the decision on the appointment of the leading gun with middle level

$$Z_{WD} = Z^k_{\min}, (Z^i_{\min} \le Z^k_{\min} \le Z^j_{\min}), k, j \in \{1,2,3\};$$

- the gun that does not run the limits for values of focusing current is taken as a leading gun

$$Z_{WD} = Z^{k}_{\min}, (Z^{k}_{\min} < Z^{j}_{\min}) \vee (Z^{k}_{\min} > Z^{j}_{\min}), k \neq j; k, j \in \{1,2,3\}.$$

2. At the level of planning on the basis of the results of diagnostics of the gun (token is in the input position $P4 = (Z_{min}, R_{min})$) and the boundary values of the current focus defined by the welding program (token in the input position $P5 = (I_{fmin}, I_{fmax}. WD)$), the following processes take place:

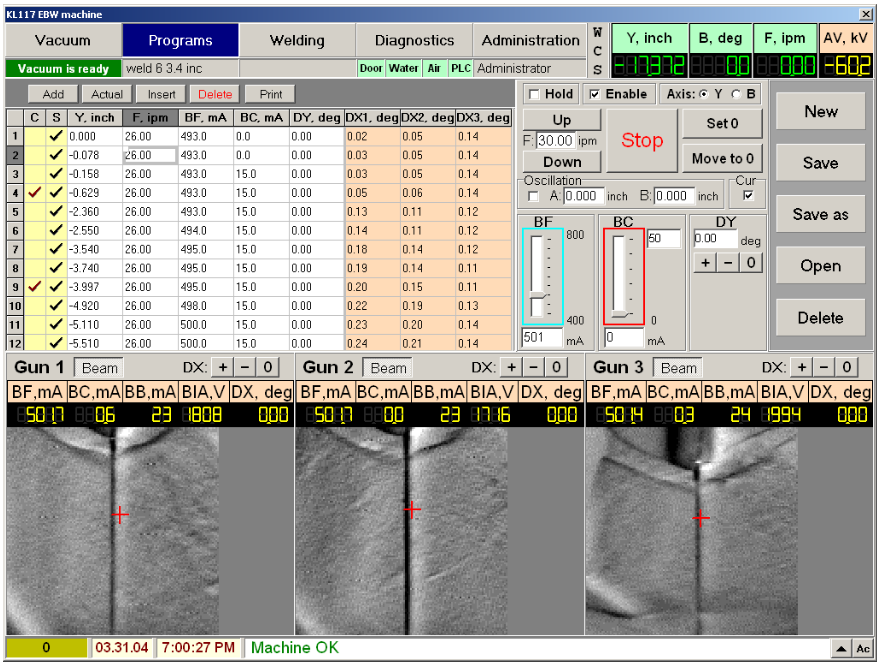
- checking of the possibility of working out a specified range of currents at the focusing current (transition *T1*) to meet the existing restrictions on focusing current

$$I^{0}_{f\min} \leq (I_{f} + \Delta I_{f} \leq I^{0}_{f\max};$$

- calculation by Formula (2) on the basis of an agreed level Z_{min} of the required change of focusing current (transition *T3*) that is on the transition *X18* checked again at the specified limit;

- output of data on the results of calculations in other agents (via the position *P43* and *P44* – the signs of acceptance/rejection of the agreed solution and via the position *P45* and *P46* – the calculated value ΔI_f are transmitted).

3. At the reactive level, the value of focusing current I_f (enters the position *P6*) set by the program using the conversion function of transition *Y3* is adjusted in ac-



Vacuum is ready | weld 6 3.4 inc | Administrator

Y, inch	B, deg	F, ipm	AV, kV
-17.372	0.0	0.00	-60.2

	C	S	Y, inch	F, ipm	BF, mA	BC, mA	DY, deg	DX1, deg	DX2, deg	DX3, deg
1		✓	0.000	26.00	493.0	0.0	0.00	0.02	0.05	0.14
2		✓	-0.078	26.00	493.0	0.0	0.00	0.03	0.05	0.14
3		✓	-0.158	26.00	493.0	15.0	0.00	0.03	0.05	0.14
4	✓	✓	-0.629	26.00	493.0	15.0	0.00	0.05	0.06	0.14
5		✓	-2.360	26.00	493.0	15.0	0.00	0.13	0.11	0.12
6		✓	-2.550	26.00	494.0	15.0	0.00	0.14	0.11	0.12
7		✓	-3.540	26.00	495.0	15.0	0.00	0.18	0.14	0.12
8		✓	-3.740	26.00	495.0	15.0	0.00	0.19	0.14	0.11
9	✓	✓	-3.997	26.00	495.0	15.0	0.00	0.20	0.15	0.11
10		✓	-4.920	26.00	498.0	15.0	0.00	0.22	0.19	0.13
11		✓	-5.110	26.00	500.0	15.0	0.00	0.23	0.20	0.14
12		✓	-5.510	26.00	500.0	15.0	0.00	0.24	0.21	0.14

☐ Hold ☑ Enable Axis: ● Y ○ B

F: 30.00 ipm

Oscillation ☐ A: 0.000 inch B: 0.000 inch Cur ☑

BF 800 400 501 mA

BC 50 0 0 mA

DY 0.00 deg + – 0

Gun 1	BF,mA	BC,mA	BB,mA	BIA,V	DX, deg
	501.7	0.6	23	1808	0.00

Gun 2	BF,mA	BC,mA	BB,mA	BIA,V	DX, deg
	501.7	0.0	23	1716	0.00

Gun 3	BF,mA	BC,mA	BB,mA	BIA,V	DX, deg
	501.4	0.3	24	1994	0.00

0 | 03.31.04 | 7:00:27 PM | Machine OK

Fig.7.19. Operator interface screen view during trajectory learning.

cordance with the generated correction ($I_f = I_f + \Delta I_f$), and an output signal is transmitted to perform in the power supply control module of the gun. Processing of the program values of focusing current is used in the cycle as they are received from the motion control module.

The presented implementation model of agent is created using the system simulation EMS. After preprocessing, it is built into the control circuit of EBW machine in a view of XML file, which is executed by the interpreter of CEN model under QNX OS. For each electron gun there is a control module similar to the presented agent model.

Trajectory learning is similar to the KL118 installation. Figure 7.19 shows the operator interface of the KL117 EBW machine.

7.6. Model-Oriented Planning and Quality Assurance

7.6.1. Tasks of Model-Oriented Manufacturing Management

Model-oriented control is useful not only at the drive and operational level of the IMS control, as it was shown in the examples described in the previous sections. We can say that modeling is currently one of the main tools for managing production activities at the strategic level during planning and quality assurance of the manufacturing activities.

Almost all modern enterprise management systems of the Enterprise Resource Planning (ERP), Manufacturing Resource Planning (MRP), or Customer Resource Management (CRM) classes (Ganesh, 2014) include model-based planning modules. For example, the Production Planning module of the Oracle e-Business Suite (Oracle, 2021) is a powerful system that combines planning models and forecasting methods with a runtime environment that allows you to quickly respond to changes in customer needs and working conditions. The same can be said about the SAP R3/S4 (Razem, 2020). In addition to the planning module, it includes a unique solution related to the dynamic modeling of enterprise processes (DEM – Dynamic Enterprise Modeling), which ensures the adaptation of software, tested on world leaders, for the business processes of a particular enterprise. The latest proposal in the field of ERP II systems from Microsoft, called Axapta, is more versatile in terms of building an electronic office of enterprise management (Microsoft, 2021). Focusing not only on large, but also on smaller enterprises, Axapta uses the most modern Western management technologies in order to optimize production activities.

If we sum up the models used in these and other enterprise management systems, then their spectrum can be represented by four main directions:

- planning of production resources;

- assessment and forecasting of financial activities;
- description of business processes;
- decision-making mechanism.

Each of these directions uses its own set of well-known mathematical methods. As a rule, these methods are focused on specific processes and have their own specific application. When planning resources, the methods of mathematical programming are most commonly used. At the same time, methods of statistical analysis and game theory are more suitable for assessing financial performance. The description of business processes is usually carried out using network methods, and in decision-making tasks it may be necessary to build a neural network or a whole hierarchical system of evaluating functions (Geunes, 2017).

The task of the enterprise management system is to ensure that all these methods work for a common goal. In modern enterprise management systems (as IMS), this task is accomplished using two basic CALS technologies that are invariant with respect to the object (products):

- project and task management (Project Management/Workflow Management);
- quality management (Quality Management).

Although these technologies have their own models, determined by the corresponding standards, they cannot be directly considered models for the implementation of management at the enterprise level. First of all, it is due to the fact that the known models of planning and quality management are not characterized by a direct impact on the control object. As a rule, the results of their work are taken into account only by the decision-maker that not only increases the reaction time of the CCS, but also introduces a pronounced subjective factor into it. In addition, these models significantly differ from models at other levels in terms of use. It creates certain difficulties in the methodological, functional and informational coordination of management levels. At the same time, the use of a unified approach to building implementation models at all levels of management will not only simplify their interaction, but also provide the basis for the complete automation of the production management process.

If we consider the planning and quality management models as models for the implementation of the corresponding algorithms, then a clear hierarchical management structure is built, when the higher-level models call the lower-located models in the order prescribed by the management system. Depending on the results obtained in the called models, management influences are formed at the next level in the hierarchy. With the existing cyclical nature of the production process, the models used will be called many times, forming a logical sequence of actions, covered by a feedback loop.

Thus, with the model-oriented method of management, the difference in approaches to the use of models at different levels of management, which was characteristic of the enterprise CCS at the early stages of their formation, is erased. It means that at the strategic level the principles of building implementation models, given in the form of CEN, must be followed. If these principles are observed, we can talk about the continuity of technologies for building and using models at different levels of management. However, we note that at the level of strategic management, the participation of the operator, or manager, is still manifested more fully, which puts forward additional requirements for the user interface.

Considering model-based management in relation to basic planning and quality management processes, we will rely on the main CALS standard ISO / IEC 10303 (ISO 10303, 2021) – the standard for the exchange of product data models (Standard for the Exchange of Product Model Data – STEP) and the international quality standard ISO 9000 (ISO 9000, 2015). First of all, we will be interested in the following requirements of these standards:

1. Project management:

- unlimited hierarchy of work in projects;
- the ability to include one work in several projects;
- association of any objects (documents, products, technological processes, etc.) with the work of the project;
- managing the revision of project milestones;
- formation of various reports on the progress of projects.

2. Workflow management:

- automation of management of formalized enterprise processes;
- support for cyclical processes, for example, returning drawings for revision;
- automatic notification of completed and overdue works;
- support of the hierarchy of processes (product development – unit development – part development).

3. Quality data management:

- ensuring a process approach to quality management;

- computer support of the quality management system;
- tracking the conformity of manufactured products to the established requirements;
- maintenance of quality records;
- presentation of the results of control of lots and copies of products;
- quality documentation management;
- monitoring and analysis of enterprise processes.

4. Messaging:

- built-in mail subsystem allows you to exchange messages among employees. In this case, both files and links to any database objects can be transferred along with messages.

5. Organization of access to data:

- the ability to set access rights to any object of the system both for an individual user and for groups of users;
- the ability to automatically assign access rights to the created objects.

7.6.2. Implementation Models of Planning Processes

Regardless of the forms of running the economy, planning has always occupied and continues to occupy a leading place in the production activities of enterprises. With the help of planning, it is possible to solve the issues of choosing a development strategy, determining plans for the purchase, production and sale of products, efficient use of enterprise resources, etc.

A particularly important role in planning is played by computer technology, which makes it possible to automate the process of drawing up plans and monitor their implementation. We can say that today, thanks to computers, planning is becoming an element of management in the modern enterprise.

There are two main approaches to planning automation. The first approach, characteristic of a market economy, was based on models and methods of project management, aimed at solving specific practical problems. The most famous mathematical models used to describe projects are PERT and GERT networks (Wiest, 2011). This direction was more in line with organizational needs, but clearly lost in the validity of decisions. The second approach, actively implemented in the system of the planned economy, consisted in the creation of automated systems for planned calculations (ASPC). It was based on the use of computational and logical systems, which were a further development of intelligent software packages for collective problem solving. The basis of the computational-logical system was an aggregated model that used second-level economic and mathematical models as modules.

The fact that the first direction prevails in today's enterprise management systems does not at all mean that there should be a complete rejection of the use of calculation methods. The challenge is to make project management more meaningful, taking into account the needs of the IMS. It can be achieved by using as models of the dynamics of projects not simple network schemes, but more complex logical systems capable of calling specific calculation methods and taking into account the results of calculations in the process of making management decisions.

Following our accepted concept of CALS, which requires the use of a unified strategy for presenting data at all levels of management, we can use control E-networks to describe work structures. First, we will consider the PERT network as a conceptual model. Without going into the details of the PERT method, we only note that in PERT networks, only nodes of the "AND" type are allowed: the event indicated by the node is performed only if all the work that preceded it has been completed.

Figure 7.20 presents the PERT network of the project, the purpose of which is to manufacture a new industrial plant.

In Fig. 7.20, jobs are marked with letters and nodes with numbers. A description of the jobs with an indication of the time of their execution is given in Table. 7.2.

The durations of all jobs are assumed to have a triangular distribution, which in this case approximates the beta distribution commonly used in network analysis. The density function of the triangular distribution is defined as follows:

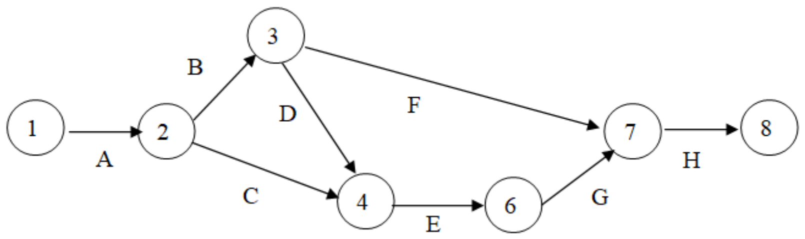


Fig. 7.20. PERT-model of the industrial plant manufacturing project.

Table 7.2 Description of the Jobs of the Industrial Plant Manufacturing Project

Jobs	Job description	Mode	Minimum	Maximum	Mean
A	Design	7	4	13	8
B	Drawings	5	3	7	5
C	Purchase	13	10	19	14
D	Construction	9	3	12	8
E	Assembly	3	1	8	4
F	Instructions	9	8	16	11
G	Testing	6	3	9	6
H	Shipment	3	1	8	4

$$f(x) = \begin{cases} \dfrac{2(x-a)}{(m-a)(b-a)}; a \le x \le m \\ \dfrac{2(b-x)}{(b-m)(b-a)}; m < x \le b \end{cases}, \tag{7.10}$$

where the mathematical expectation and variance are respectively equal

$$\mu = \frac{a+m+b}{3} \tag{7.11}$$

and

$$\sigma^2 = \frac{a(a-m)+b(b-a)+m(m-b)}{18}. \tag{7.12}$$

The minimum value is interpreted as an optimistic estimate, and the maximum value is interpreted as a pessimistic estimate of the duration of work.

If we take the average values for the actual duration of the work, then the time characteristics of the project can be calculated analytically using the critical path method (CPM) (East, 2015). The results of such calculations are shown in Table 7.3.

In Table 7.3, the following designations are used:

• $\omega_k^{'}$, $\omega_k^{''}$ – a set of immediately preceding and immediately following jobs with respect to the k-th job;

• τ_k – duration of the k -th work;

• $\xi_k^{'}$, $\xi_k^{''}$ – the earliest and latest completion dates of the k-th work;

Table 7.3 Calculation of the Time Characteristics of the Project by CPM

Job	$\omega_k^{'}$	$\omega_k^{''}$	τ_k	$\xi_k^{'}$	$\xi_k^{''}$	$\Delta\tau_k$	Critical path
A	–	B, C	8	8	8	0	*
B	A	D, F	5	13	14	1	
C	A	E	14	22	22	0	*
D	B	E	8	21	22	1	
E	C, D	G	4	26	26	0	*
F	B	H	11	19	32	13	
G	E	H	6	32	32	0	*
H	F, G	–	4	36	36	0	*

- $\Delta\tau_k$ – reserve time of the *k*-th job.

The target time is 36 days, the critical path is formed by works A-C-E-G-H. Now we will show how a PERT network can be modeled using CEN. Using EMS, let us build a CEN model corresponding to the PERT network of the project shown in Fig. 7.20. We assign to each work a transition of the "T" type. In the places of branching, we add transitions of the "F" type, and in the nodes of the merge of works – transitions of the "J" type. The delay time at transitions of the "T" type will be calculated according to the triangular law with the parameters given in Table 7.3. The value of the activation functions is set equal to 1. The CEN model for this project will have the form shown in Fig. 7.21.

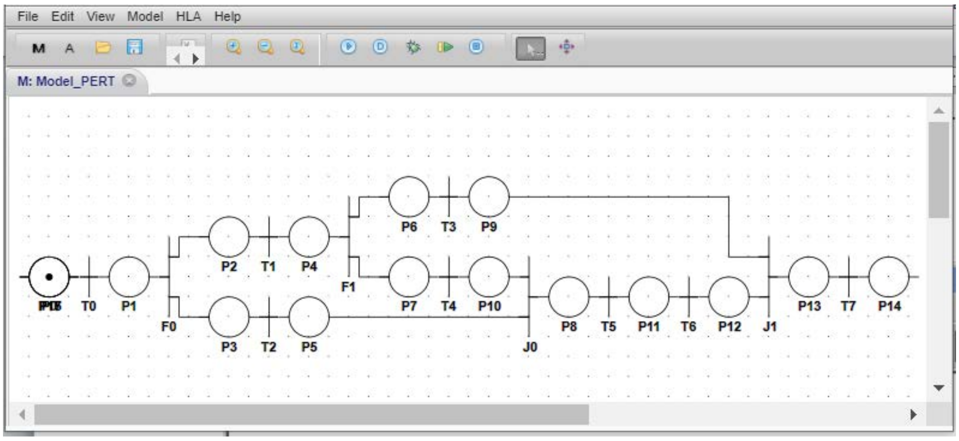


Fig. 7.21. CEN model of the industrial plant manufacturing project.

It is clear that for the CEN network modeling the PERT network, it is possible to determine an analytical method for calculating both the directive time and other characteristics of the project network schedule based on the MCP. The calculation algorithm in this case will be as follows:

1. Based on the structure of the CEN network, we construct an incidence matrix of transitions of the "T" type, which we denote as $\|\alpha_{ij}\|$, where $i, j = \overline{1, |T_T|}$. In what follows, we will consider only transitions $t \in T_T = \{t_k\}, k = \overline{1, |T_T|}$. We will consider a transition t to be incidental to a transition t_k if it is the first t-transition in the firing chain after the transition t_k. Let us assign the value 1 to the elements of the matrix that satisfy this condition. For example under consideration, the incidence matrix of T-transitions can be represented in the form of Table 7.4.

Table 7.4. T-Transition Incidence Matrix

t\t	T0	T1	T2	T3	T4	T5	T6	T7
T_0	–	1	1	0	0	0	0	0
T1	0	–	0	1	1	0	0	0
T2	0	0	–	0	0	1	0	0
T3	0	0	0	–	0	1	0	0
T4	0	0	0	0	–	0	0	1
T5	0	0	0	0	0	–	1	0
T6	0	0	0	0	0	0	–	1
T7	0	0	0	0	0	0	0	–

2. For each transition t_k, we determine the set of immediately preceding transitions $\omega'_k = \{t_j | \alpha_{jk} = 1\}$ and the set of immediately following transitions $\omega''_k = \{t_i | \alpha_{ki} = 1\}$.

3. In all transitions t_k for which $\omega'_k = \emptyset$, let us assign $\xi'_k = \tau_k$.

4. In all transitions t_k for which $\omega'_k = \emptyset$, let us assign $\xi'_k = \max\limits_{t_j \in \omega'_k}(\xi'_j + \tau_k)$.

5. We calculate the target time using the formula $D = \max\limits_{k} \xi'_k$.

6. In all transitions t_k for which $\omega'_k = \emptyset$, let us assign $\xi''_k = D$.

7. In all transitions t_k for which $\omega'_k = \emptyset$, let us assign $\xi'_k = \max\limits_{t_i \in \omega''_k}(\xi''_i - \tau_i)$.

8. We calculate the time reserve for all transitions $\Delta\tau_k = \xi''_k - \xi'_k$.

9. In all transitions for which $\Delta\tau_k = 0$, let us assign the sign of belonging to the critical path.

However, the advantage of CEN is not the ability to perform an analytical solution on the web. It is important that with the help of CEN, using the method of simulation, it is possible to obtain statistical estimates of these characteristics, which take into account the influence of random factors. In this case, various distribution laws can be used to calculate the delay time during transitions. Simulation runs of the model can be organized both in the EMS modeling system and in the model runtime built into the project management system. For example, statistical experiments with the model shown in Fig. 7.21, performed with EMS, gave the following results: for 50 model runs, the mean value of the directive time was 36.2 with a variance of 12.8, which coincided with the analytical calculations.

In addition to preliminary evaluation of the characteristics of the plan, the CEN model can be used to organize programmatic management of the project progress. For this purpose, it is sufficient to include in the definition of the activation functions of T-transitions taking into account signals indicating the completion of the previous work, and use the delay time at transitions only in forecasting models. Then the CEN network will be transferred to the category of implementation models that initiate the implementation of subsequent work by issuing the corresponding output signals. Simultaneously, for dynamic estimation of the directive time in the control process, both analytical and simulation methods of modeling can be applied. In the latter case, the actually measured delay time is set for the work already performed. Carrying out statistical experiments with a model that is used as a control algorithm becomes possible due to the fact that at the strategic level (when planning and managing the work flow), the response time requirements are not as stringent as at the drive or operational and tactical levels. It increases the cycle time to the level required for multiple runs of the model.

When modeling PERT networks using CEN, all the capabilities of the latter as a means of describing control processes are not implemented. Due to their analytical orientation, PERT networks still have significant structural limitations: they do not allow for the use of loops and exclude probabilistic branching. Therefore, the PERT model cannot include the feedback operations required for project management. More preferable for CEN in terms of conceptual models are GERT networks, which are free of the above drawbacks, although they also have their limitations for the same reasons as PERT networks.

The best solution can be obtained by combining the capabilities of the PERT and GERT networks by including additional "OR" nodes at the input and output in the PERT network. With the help of the extended PERT network, it is possible to represent not only the network diagram of the project, but also more complex diagrams describing the workflows.

Let us consider the application of the extended PERT network as a conceptual model of the semiconductor manufacturing process, which includes the works related to checking and preparing the oven (Fig. 7.22).

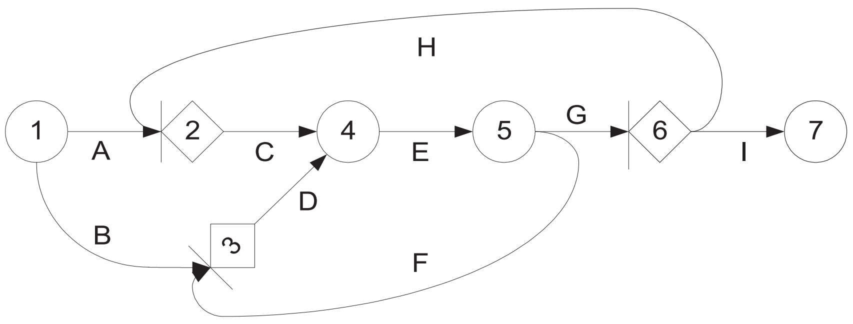


Fig. 7.22. Conceptual model of the semiconductor manufacturing process.

As shown in Fig. 7.22, the process begins with the receipt of raw materials (work A). At the same time, the oven power supply is checked (operation B). After molding the raw material (C) and checking the furnace (D), the loaded samples (E) are fired. Then the obtained samples go to the final control (G), and the oven is cleaned and prepared for the next heating (F).

After inspection, the samples obtained are either recognized as good crystals and packed for shipment (I) or sent for recycling (H).

CEN interpretation of the extended PERT network will be as follows:

- nodes with input function “OR” will be represented by transitions of type “Y”;
- nodes with an output function “OR” – transitions of the “X” type;
- conditions of probabilistic choice for the specified nodes will be determined using decision functions;
- we assign transitions “J” to the nodes of the “I” type, and nodes without a probabilistic choice at the output – transitions of the “F” type;
- as before, the planned works will be depicted using transitions of the “T” type;
- the execution time of work will be modeled by the delay time at the transition, which is calculated according to a given distribution law.

For the example under consideration, the CEN model built according to these rules will have the form shown in Fig. 7.23.

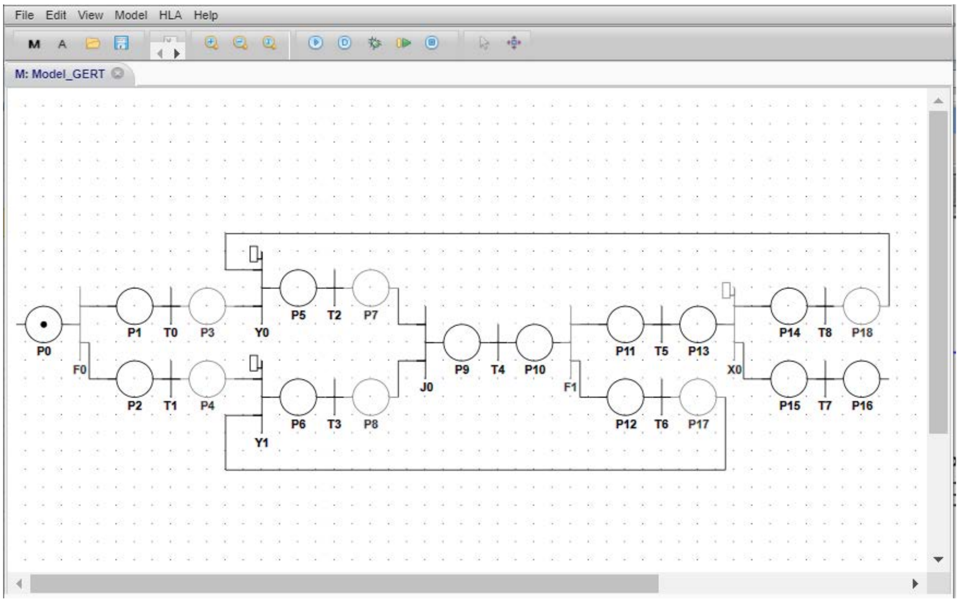


Fig. 7.23. CEN model of the semiconductor manufacturing process.

The formal definitions of extended PERT networks using CEN are used in the same way as in the case of conventional PERT networks; however, analytical solutions cannot be obtained for them. By means of statistical experiments, estimates of the average and variance of the time of complete processing of a batch of raw materials are calculated (see Fig. 7.23). The condition for the end of the cycle is the remaining amount of raw materials, which is calculated each time at transition T8 and stored in the tag attribute. At transition X0, the remainder of the raw material is checked, and if it turns out to be less than the specified level, the mark is sent to position P15 – the process ends.

Summarizing the consideration of planning models, we note that the use of CEN as network diagrams of plans and workflows has several advantages over other methods. These advantages lie in the capabilities provided by CEN:

- simulation of various options for project development, taking into account a large number of random factors and existing risks;
- creation of aggregated models that call functional modules for calculating the current values of production indicators;
- monitoring the progress of the project and analysis of its current state;
- dynamic evaluatation of design performance using both analytical and statistical methods;

- use of estimates of design indicators when choosing the direction of project development;

- predicting the development of the project with the existing trends in its dynamics;

- management of the project progress by generating conditions for the start of work.

7.6.3. Risk Assessment in Model-Oriented Planning and Management

In network planning, risk is the result of unforeseen events, which make the work impossible to be performed on time. It leads to delays in the entire production process or project execution time and to unforeseen losses of financial, time and labor resources. Therefore, the implementation of any plan largely depends on its reliability, which is determined by the probability of fulfillment of the technical and economic indicators laid down in the plan.

Even greater demands are placed on real-time work planning. In general, the task of real-time network planning is to ensure the implementation of the project under the given constraints, which must be checked during the implementation of the plan. In this case, the problem of planning is in many respects similar to the problem of dynamic verification, in the solution of which varieties of temporal logic can be used to formally determine the constraints on the operating conditions.

According to PMBOK (Project Management Body Of Knowledge) (PMBOK, 2017), risk management is the process of identifying, analyzing risks and making decisions, which include maximizing the positive and minimizing the negative consequences of risky events. Thus, risk is an activity associated with overcoming uncertainty in a situation of choice, in the process of which there is an opportunity to quantify and qualitatively assess the probability of achieving the predicted result or deviation from the goal.

According to risk management standard (ISO/IEC, 2019), risk is defined as a consequence of the impact of uncertainty on the achievement of objectives.

The basic concepts of risk include:

- risk assessment is a process that includes risk identification, risk analysis and comparative risk assessment;
- level of risk is a measure of risk or a combination of several types of risk characterized by the consequences and their plausibility (probability).

To estimate the probability, the following resources are used:

- chronological data to identify an event or situation that occurred in the past and extrapolate their occurrence in the future;

- forecasting methods, such as error tree analysis and event tree analysis;
- expert assessments in a structured probability assessment process.

In a situation where there are several options, risk assessment must be performed for each of the alternatives.

There are the following groups of risk assessment methods:

- methods of observation (method of expert assessments);
- scenario analysis;
- functional analysis;
- statistical analysis (Markov analysis, Monte Carlo method, Bayesian analysis).

Methods of statistical analysis carry out quantitative risk assessment. For example, according to the Bayesian method (Kruschke, 2014), even before the data are obtained, decision maker considers the degree of their confidence in possible models and presents the data in the form of probabilities. Once the data are obtained, Bayes' theorem allows us to calculate a new set of probabilities that represent new degrees of confidence in possible models that take into account new information from the received data.

Of the three stages of risk assessment, which include risk identification, risk analysis and comparative risk assessment, these methods cover only the last stage, i.e., none of the above methods of statistical analysis provides support for all components of the risk assessment process.

Recent studies of methods for assessing the reliability and risks are the following:

- The method based on oriented graphs – the model of the project implementation process is considered in the form of a Cyclical Alternative Network Model (CASM) (Voropayev, 2013). With the help of CASM, it is possible to take into account the alternative nature of both the technology of production of works and methods of allocating resources for work to carry out their optimal purpose with the optimal rate of use. However, this method takes into account only consistent work and a priori risk assessment before the experiment.
- The method of finding the optimal strategy within the Markov decision-making process with non-Markov rewards (Thiebaux et al., 2006) includes the task of checking the properties of the system expressed by probabilistic temporal logic. However, this method is not suitable for solving the problem of dynamic risk assessment and decision-making in network planning because in a general case planning processes are non-Markov.
- The method based on the use of alternative stochastic network models includes analysis of the stochastic graph using a simulation model, and it is a combination of Ford-Falkerson algorithm (Laube and Nebel, 2016) to find the maximum path length, logically justified calculations, and elements of the sta-

tistical test method. The disadvantage of this method is that the evaluation of the implementation goes to the beginning of the project, and in real time it does not work.

- Cognitive map method. The essence of the method is to build cognitive maps and implement on their basis modeling of different scenarios. It allows you to predict the occurrence of certain events that may adversely affect the results of project activities or activities of the enterprise.

However, these methods do not take into account the intersection of time intervals and changes in the state in the process of work with the simultaneous analysis of alternative paths in the work plan. This problem is solved by the method of dynamic risk assessment in network planning and management, based on the combination of implementation models as CEN and predictive models as CTL in simulation mode.

Let us assume that the work plan of CPS manufacturing looks like as shown in Fig.7.24 (Kazymyr, 2017). Alternative ways to implement the work plan in Fig. 7.24 are shown by a dotted line.

With the beginning of execution of the plan of works (1) there is a division of ways of works which will be carried out in parallel, namely: a choice and the further works with hardware (2) and software (3). When choosing the hardware, the choice of the tablet is between two manufacturers. The difference in choice will affect the financial costs. This is followed by the division (13) into the parallel performance of such tasks as: the development of the identification system (14), the choice of server configuration (15) and the choice of network equipment (16).

When choosing software, tasks (3) should be divided into client (6) and server (7) parts. For client software, the question is: under which platform to develop? That is why there is a choice of alternatives (6) among the available platforms. Choosing

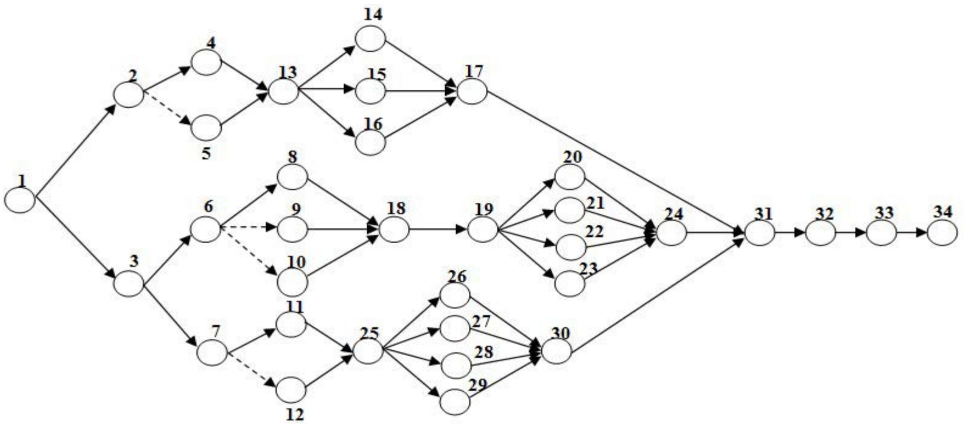


Fig. 7.24. Work plan of CPS manufacturing.

one of the options will affect the development time. This is due to the relevance, openness, popularity, functionality of each of the platforms, which will certainly affect the time spent searching for information, ways to solve problems, the development and further support.

Next is the work that will be used to select the architecture (18) and programming language (19). After selecting everything you need to develop software for the client part, the coding time (20–23) comes next. It is reasonable to divide the tasks into several parts for parallel programming by several programmers, which will allow you to do the job faster than one person would do. It is important to choose a database for the server software. Therefore, there is a choice of alternatives (7) between the commercial and open systems (11 and 12). The choice depends on the financial costs, as well as development time, in particular due to the openness and quality of documentation.

Next is the stage of development of the server part. As mentioned earlier, to optimize the development time for the available number of human resources, the work is divided (25) among developers into the following tasks: development of the repository layer (26), domain layer (27), service layer (28) and application layer (29). Documentation is maintained at all levels – hardware and software (17, 24, 30). After the hardware and software development is completed, there is an integration phase (31), a testing phase (32) where you can detect bugs and return the system for refinement, a documentation phase (33), and the final startup phase – implementation (34) of this system.

The implementation model of the work plan is shown in Fig. 7.25.

Let restrictions introduced on the work plan relate to the maximum value of the cost of work – no more than 10,000 monetary units, and the maximum time of execution of the work plan – no more than 250 units of time.

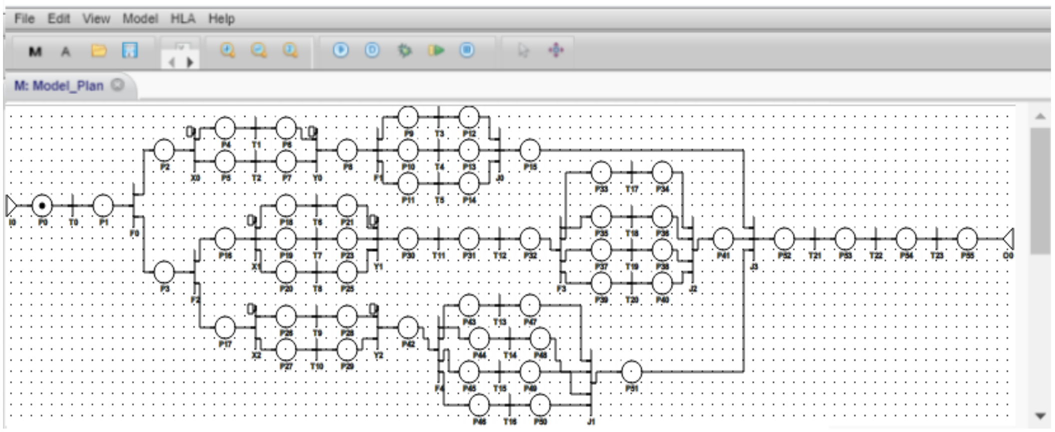
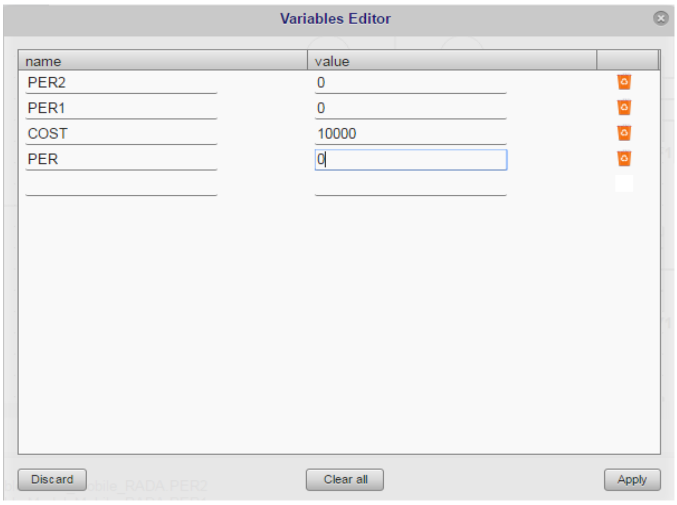


Fig. 7.25. Work plan implementation.

Figure 7.26 presents an editor window with certain model variables:

- COST – initialized with a value that limits the financial resources, 10000;
- PER, PER1, PER2 – auxiliary variables that are needed to determine the alternative path after each transition type "X".



name	value
PER2	0
PER1	0
COST	10000
PER	0

Fig. 7.26. Model variable editor window.

The variable that limits the project execution time is not defined – it will use the system variable TIME, which is available for all functions of the model.

In the model, transitions *X0, X1* and *X2* have alternative paths. Thus, at the transition *X0* there is a choice of tablet type (place *P4* or *P5*), at the transition *X1* – the choice of client software platform (place P18, or *P19*, or *P20*), and at the transition *X2* – the choice of database for server software (place P26 or *P27*).

The transition delay function *X0*, defined as *RETURN 1*, means the transition delay is determined by 1 unit of time. The crucial function of this transition is determined by the entered variable *PER*:

V ['PER'] = UNIFORM (0,1);

IF (V ['PER'] <0.5) RETURN 0;

IF (V ['PER']> = 0.5) RETURN 1.

It means that the choice of an alternative way to continue the process of work will be in accordance with the law.

The delay function of the transition *X1* is defined as *RETURN 1*; the crucial function of this transition is determined using the entered variable *PER1*:

V [‘PER1’] = UNIFORM (0,1);

IF (V [‘PER1’] <0.33) RETURN 0;

IF (V [‘PER1’]] = 0.33 && V [‘PER1’] <0.66) RETURN 1;

IF (V [‘PER1’]> = 0.66) RETURN 2.

Function of delay of transition *X2* is defined as *RETURN 1*; the decisive function of this transition is defined by means of the entered variable *PER2*:

V [‘PER2’] = UNIFORM (0,1);

IF (V [‘PER2’] <0.5) RETURN 0;

IF (V [‘PER2’]> = 0.5) RETURN 1.

Type “Y” junctions are required to join branches of alternative paths. The crucial function of the transition *Y0* is defined as follows:

VAR isPlace2Marked = P [‘P7’]. T;

IF (isPlace2Marked == TRUE) RETURN 1;

ELSE RETURN 0.

In this case, the presence of the label in the input positions of the transition is checked using the entered local variable of the transition *isPlace2Marked*.

The crucial function of the transition *Y1* is defined as follows:

VAR isPlace2Marked = P [‘P23’]. T;

VAR isPlace3Marked = P [‘P25’]. T;

IF (isPlace2Marked == TRUE) RETURN 1;

IF (isPlace3Marked == TRUE) RETURN 2;

ELSE RETURN 0.

The decision function of the transition *Y2* is defined as follows:

VAR isPlace2Marked = P ['P27']. T;

IF (isPlace2Marked == TRUE) RETURN 1;

ELSE RETURN 0.

Other functions (delays and conversions) for Y transitions are not specified.

At transitions T, the concrete work executed in the project is modeled. At the same time, it may take some time, and possibly some resources. This is determined by the delay and conversion functions. For example, for the transition *T0* these functions are defined as follows:

- delay function: "*RETURN 8;*", which means a delay at the transition of 8 units of time;
- conversion function: "*V [' COST '] = V [' COST '] - 200*;", which means the consumption of 220 units of monetary resources on this transition.

For the transition *T2*, these functions are defined as follows:

- delay function: "*RETURN UNIFORM (90,251);*", which means the delay at the transition in the interval [90, 251] units of time, set by the uniform law;
- conversion function: "*V [' COST '] = V [' COST '] - 2000;*", which means the cost of this transition 2000 units of monetary resources.

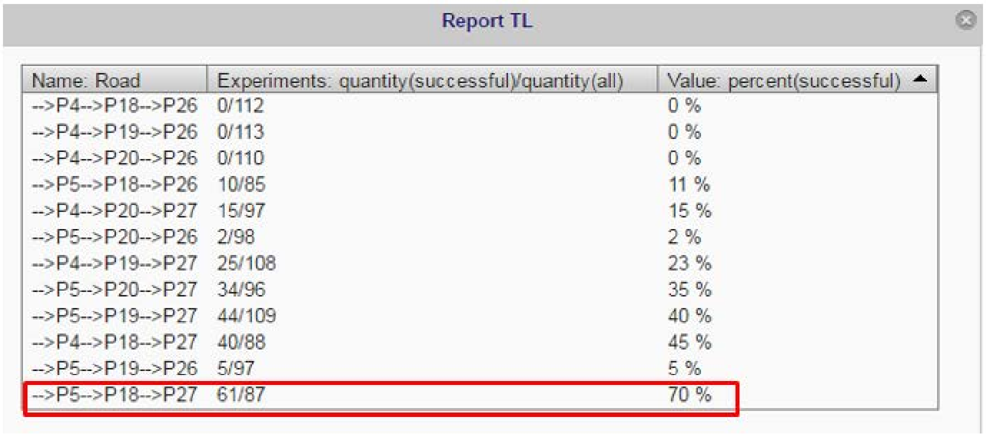
The transition functions of the other T-junctions are set similarly.

The following parameters are set for the experiment:

- 400.0 – system simulation time, it will be enough with all the delays that may occur during the run of the model;
- 1200 – the number of runs of the model;
- AG – an operator of temporal logic;
- V ['COST']> = 0 && TIME <250 – a formula that checks the condition whether the specified financial resource has been exceeded and whether the project execution time has been exceeded by 250 units of time.

Figure 7.27 shows a table with the results of the experiment.

In the table of results, the path -> P5 -> P18 -> P27 with the lowest risk (30%)


Report TL

Name: Road	Experiments: quantity(successful)/quantity(all)	Value: percent(successful)
-->P4-->P18-->P26	0/112	0 %
-->P4-->P19-->P26	0/113	0 %
-->P4-->P20-->P26	0/110	0 %
-->P5-->P18-->P26	10/85	11 %
-->P4-->P20-->P27	15/97	15 %
-->P5-->P20-->P26	2/98	2 %
-->P4-->P19-->P27	25/108	23 %
-->P5-->P20-->P27	34/96	35 %
-->P5-->P19-->P27	44/109	40 %
-->P4-->P18-->P27	40/88	45 %
-->P5-->P19-->P26	5/97	5 %
-->P5-->P18-->P27	61/87	70 %

Fig. 7.27. Results of the experiment.

to meet the constraint specified in the form of the CTL formula can be selected, the accuracy of the experiment is defined by d= 0.02 (the deviation is at a confidence level of 95%) and the number of alternative paths N=12. The model can be complicated by increasing the number of alternatives (the factors on which the implementation of the work plan depends) and setting additional constraints.

Thus, the developed simulation model allows predicting the implementation of work plans taking into account the risks, as well as ensuring the stability of plans under conditions of uncertainty.

7.6.4. Implementation Models of Quality Management Processes

If we proceed from the principles of building implementation models, then quality management in its form should not differ much from project management. However, unlike planning models, quality management models do not have such pronounced analogues that can be used as conceptual models when constructing formalized schemes of control algorithms using CEN. In this regard, the basic methodological approaches that form the basis for the construction of quality management systems should be analyzed in order to determine the role and place of MOC in these systems.

To ensure the required level of quality in international practice, two approaches have been used: product-oriented and process-oriented. Both approaches require a quality management system. Such a system defines the objectives of management in relation to quality, establishes its policy and details the necessary actions.

In the first approach, the emphasis is placed on quality control by checking the finished product. This approach is based on the assumption that the more errors are detected and removed during the final control of the product, the higher its quality.

In the second approach, the emphasis is placed on taking measures to prevent, promptly identify and remove product flaws by timely defining responsibilities, provision plans, basic procedures for ensuring the quality of products, as well as taking appropriate measures sequentially, starting from the initial stages of the life cycle.

The process approach in our time can be considered generally accepted. It underlies the concept of Total Quality Management (TQM) (Kiran, 2016), which is implemented in numerous international standards, draft standards and working materials. Internationally, specialized systems of standards are formed by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IES). International standards of the ISO 9000: 2015 (ISO 9000, 2015) series establish the principles of enterprise management based on the process approach.

According to ISO 9000 standards, regardless of the product category, process-based quality management takes into account, inter alia, the following aspects:

- description of processes according to the rule "inputs – operations – outputs";
- identification of links between processes;
- identification of operating procedures.

As you can see, this definition exactly corresponds to our approach to building implementation models. It remains only to clarify the concepts of the control object for these processes, the list of processes and the details of their operational content.

As a CO for the processes of the quality management system, documents are considered that can be located on any data carrier: paper, diskette, disk, etc. Quality management documentation is divided into three levels. The first level documents include "Quality Manual" and "Quality Policy". These documents are fundamental, defining the structure of the processes. Within processes, they can be used as links, although in some cases they can be controlled objects, in particular when making changes.

Second level documents are methodologies and instructions that describe procedures for fulfilling the requirements of the standards. They set the content of the processes tied to the specific conditions of the enterprise.

Finally, the third level documents include work instructions, test procedures and quality records. They also include documents accompanying input and output data, as well as internal documents that ensure the implementation of work procedures.

The main processes that form quality at the stages of the full life cycle of products defined by the ISO 9001: 2015 (ISO 9001, 2015) standard include:

- requirements management (requirements);
- development and planning of the project (developing);
- control (testing);
- support (maintenance);
- changes (changing);
- inspection (inspection).

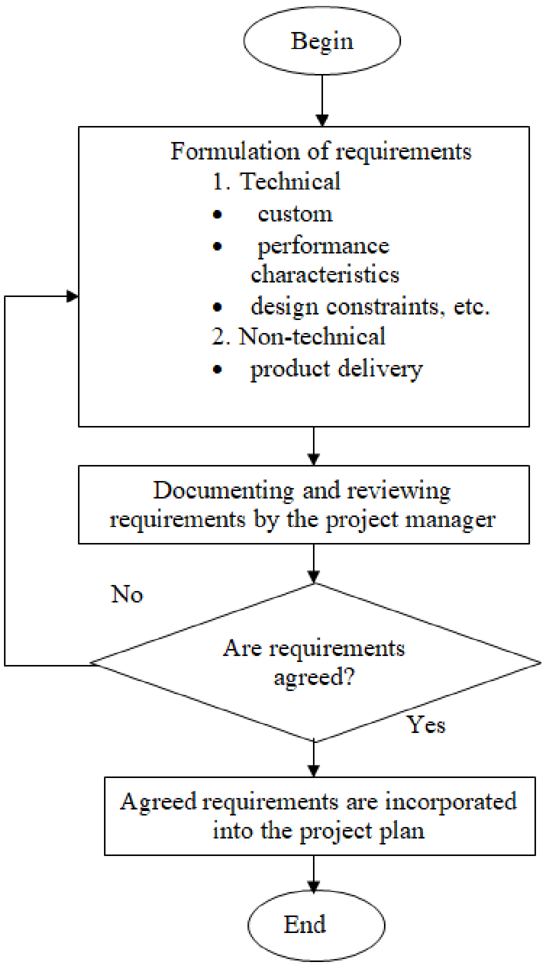

Begin
Formulation of requirements
1. Technical
• custom
• performance characteristics
• design constraints, etc.
2. Non-technical
• product delivery
Documenting and reviewing requirements by the project manager
No
Are requirements agreed?
Yes
Agreed requirements are incorporated into the project plan
End

Fig. 7.28. Requirements management process algorithm.

Each of these processes can be described using flowcharts. As an example, let us consider the requirements management process, the algorithm diagram of which is shown in Fig. 7.28. Requirements management aims at achieving and maintaining an agreement with the customer on the requirements for a development project. A customer can mean a marketing group, an internal organization, or an external customer. This agreement is referred to as the "System Requirements" set for the project and covers both technical and non-technical requirements (e.g., lead times). The agreement forms the basis for costing, planning, executing and tracking project work throughout the entire product life cycle.

System requirements for a product, equipment, and other system components, such as people, can be enforced by a team of analysts, and developers do not have to directly control this distribution. The development team also takes appropriate steps to ensure that requirements are under control that fall under the responsibility of the developers. To ensure this control, the development team reviews the original and revised system requirements and tries to resolve possible issues before the requirements are introduced into the development project. Any change in system requirements is accompanied by changes in the broken development plans in order to align them with the updated requirements.

A formalized model of the requirements management process developed using EMS is shown in Fig. 7.29.

Transitions are modeled by the following procedures:

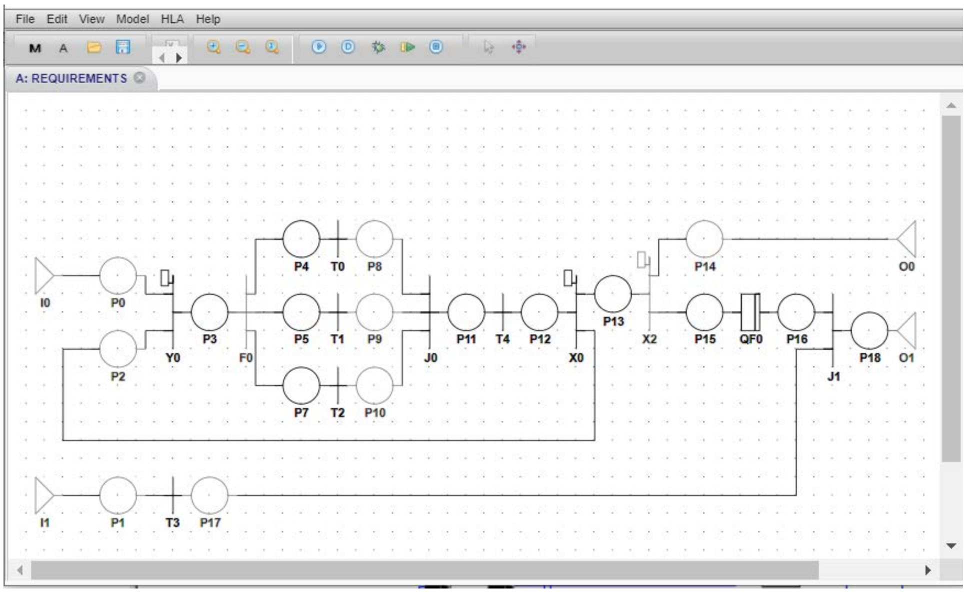


Fig. 7.29. CEN model of the requirements management process.

- Y0 – receiving a change request;
- F0 – sending requests to groups;
- T0 – requirements development by the project manager;
- T1 – formulation of requirements by the support team;
- T2 – formulation of requirements by a group of analysts;
- J0 – a summary of the drawn up requirements;
- T4 – review of requirements by the project manager;
- X0 – checking and agreeing on consistency requirements and
- adequacy;
- X2 – making changes to the development project;
- QF – distribution of documents in the order of the queue;
- J1 – sending documents for inspection on demand.

The decisive function of the transition X0 determines the direction of the development of the process depending on the value of the first attribute of the label according to the rule:

if (P9.m [1] = 0) then R1: = 1; // if the requirements are agreed;

If (P9.m [1] = 1) then R1: = 2; // if requirements are not agreed.

The aggregate states corresponding to the marked positions have the following definitions:

- P3 – request for the formulation of requirements;
- P4 – request processing by the project manager;
- P5 – request processing by the support team;
- P6 – request processing by a group of analysts;
- P8 – documentation of requirements by the project manager;
- P9 – documenting requirements by the support team;
- P10 – documenting requirements by a group of analysts;
- P11 – receipt of documents for consideration by the project manager;
- P12 – receipt of a document for approval;
- P14 – request for development and planning;
- P0 – request for changes.

In a similar way, other life cycle processes are modeled in form of DEVELOPING, CHANGING, INSPECTION, MAINTENANCE and TESTING aggregates. These processes are not self-contained.

The scheme of interaction of processes is modeled by establishing connections between the aggregates as it is shown in Fig. 7.30. The links between the units are established by connecting the corresponding boundary positions, through which the transfer of labels (control) between the processes takes place. As a result, even the execution of a single project occurs within the TQM quality management cycle, which includes: planning, execution, control and corrective actions. Note that the inspection process, as in our case, is usually outside the scope of this cycle, since it affects all processes.

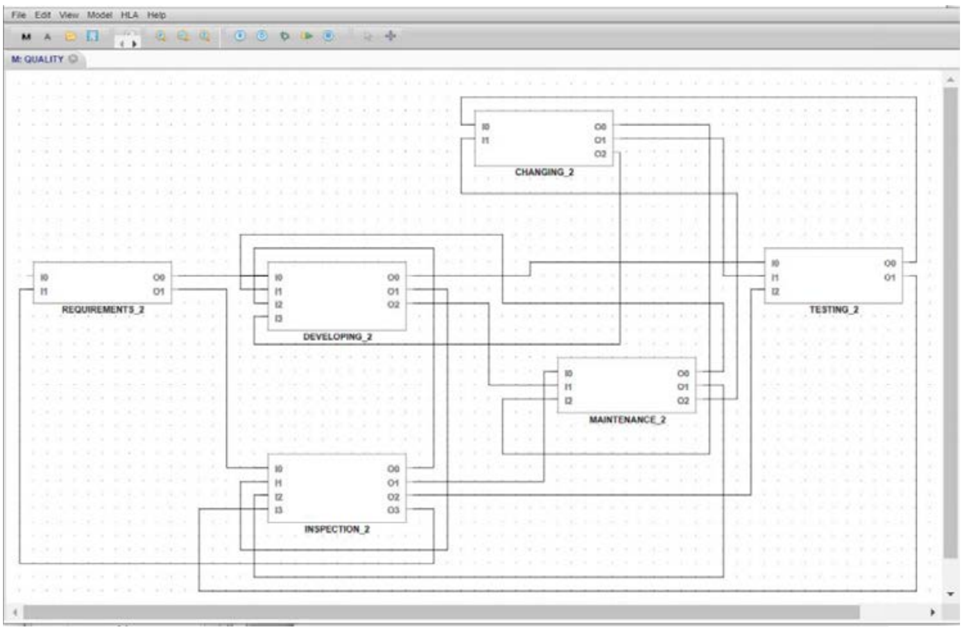


Fig. 7.30. Aggregate implementation model of product life cycle.

Implementation models make it possible to abandon expert assessment of many characteristics of quality management processes. Using the attributes of labels and transformation functions of transitions for the accumulation and calculation of data characterizing the performance of certain operations, it is possible to evaluate the following characteristics:

- quality of definition of input and output data, their sufficiency;
- quality of registration of execution of process operations;
- quality of technologies for performing process operations;
- quality of managerial decision-making by a manager (regularity of control, analysis of reports, etc.);
- quality of documentation.

If we talk about the quality of forecasting the development of processes by the leadership, then the real forecast can be compared with the results of the assessment obtained using forecasting models based on the dynamic analysis of CEN.

Estimates of the temporal characteristics of processes can be obtained even more simply: the duration of preparation, the duration of operations and the total time of the process, which are mapped into the standard numerical characteristics of transitions obtained as a result of statistical experiments. As with scheduling tasks, these experiments can be performed in real time during the process, which will be controlled by the implementation model.

7.7. Summary

EBW machines are outstanding representatives of the class of industrial robots, on the basis of which an intelligent production system can be formed. The basis of such a system is a model-oriented CS, built on the principles of a hierarchy of goals, synchronicity, reliability, distribution, flexibility and openness, which are realized by integrating high-quality executive equipment with intelligent control based on embedded models.

The use of implementation models in combination with forecasting models when controlling a vacuum system, a power source and a displacement system made it possible to solve the problem of timely switching of CO operating modes in order to prevent the development of dangerous situations.

The use of recovery models built into the control loop made it possible to implement:

- a visual method of designing welding programs with multi-axis movements;
- an adaptive method of tracking the butt during welding;
- a multi-agent control of the simultaneous operation of several electron guns as part of one ELS installation.

The use of model-based control methods for EBW machines makes it possible to significantly improve the quality of products and the efficiency of the production process as a whole, which is confirmed by the given technical characteristics of the CS and the experience of operating these installations at enterprises of the aerospace and metallurgical industries.

Implementation and predictive models are one of the main tools for organizational management of production activities at the level of the automated control system, where planning and quality management processes prevail. These models can serve as the basis for building management modules that provide financial activities, decision support and organizational management of the enterprise.

To manage planning processes, it is proposed to use implementation models built on the basis of conceptual models in the form of extended PERT networks. In this case, both methods of analytical calculation of the main parameters of the plan and simulation experiments with built-in models can be applied. The complex of developed basic models of quality management processes provides support for the full life cycle of products.

References

Akopyants K. S., Nazarenko O. K., Gumovsky V. V., Chernyakin V. P. (2002) Diagnostic system of an electron beam in electron beam welding machines// Automatic welding. No. 10. pp. 30–33.

Albert W., Yao L. (2010) A petri nets-based process planning system for wastewater treatment, Asian Journal of Control.

Alpern B., Scheider F. B. (1985) Defining liveness. Information Processing Letters. Vol. 21, No. 4. pp. 181–185.

Alur R. (1991) Techniques for automatic verification of Real-Time systems. PhD thesis, Stanford University, 275 p.

Alur R., Henzinger T. A. (1990) Real-time logics: complexity and expressiveness. In Proceedings of the Fifth Annual Symposium on Logic in Computer Science, IEEE Computer Society Press, pp. 390–401.

Alur R., Henzinger T. A. (1993) Real-time logics: complexity and expressiveness. Information and Computation. No. 104(1), pp. 35–77.

Alur R., Peled D., Penczek W. (1995) Model Checking of Causality Properties. In Proc. 11th IEEE Conf. Logic in Computer Science, pp. 90–100.

Arica E. and Powell D. (2017) Status and future of manufacturing execution systems// In Industrial Engineering and Engineering Management (IEEM), 2017 IEEE International Conference, pp. 2000–2004.

Asteziano E., Zucca E. (1995) D-oids: a Model for Dynamic Data Types. Mathematical Structures in Computer Science. Vol. 5, No. 2, pp. 257–282.

Astrom K, Wittenmark B. (1996) Computer-Controlled Systems, Theory and Design, Prentice Hall.

Astrom K. (2008). Adaptive control. Dover, 2008. pp. 25–26.

Bailo Clark P. Yen C. J. (1997). Open modular architecture controls at GM Powertrain: technology and implementation Proceedings of the SPIE, Vol. 2912, pp. 52–63.

Baranov S. Logic Synthesis for Control Automata. Springer, (1994)393 p.

Basile, F., Chiacchio P., Coppola J., Gerbasio D. (2015). Automated warehouse systems: A cyber-physical system perspective. In 2015 IEEE 20th Conf. Emerging Technologies & Factory Automation (ETFA), pp. 1–4.

Bassi L. (2017) Industry 4.0: hope, hype or revolution? Conference: IEEE 3rd International Forum on Research and Technologies for Society and Industry - Innovation to Shape the Future for Society and Industry (RTSI), pp.1-5. DOI:10.1109/RTSI.2017.8065927.

Bechet, D., De Groote, P., Retoré, C. (1997), "A complete axiomatisation for the inclusion of series-parallel partial orders", Rewriting Techniques and Applications, Lecture Notes in Computer Science,1232, Springer-Verlag, 1997. pp. 230–240.

Beyaert B., Florin G., Long P., Matkin S. (1981) Evaluation of Computer Systems dependability using stochastic Petri Nets // FTCS–11: The Eleventh Annual Int. Symp. Foult – Tolerant Computing, Portland, 1981. pp. 79–81.

Berg (1985). "CAD/CAM's Pioneer Bets It All". The New York Times.

Berkeley (2020), Electrical Engineering and Computer Sciences at UC Berkeley, https://ptolemy.berkeley.edu/projects/cps/ [accessed in May 2020].

Bibel, Wolfgang (2007). Early History and Perspectives of Automated Deduction. KI 2007. LNAI. Springer (4667): 2–18.

Bird R. S. (1993) Lectures on constructive functional programming // Conctructive Methods in Computer Science. – NATO ASI Series F: Springer Verlag. –. No. 55, pp. 151–218.

Boyer, S. (1999). (1999) SCADA Supervisory Control and Data Acquisition, 2nd Edition, ISA.

Bowen J., Hoare C. A. R., Langmack H., et al. (1996) ProCoS II: A ProCoS II project final report // Bulletin of the EATCS. – No. 59, pp. 76–99.

Brauer W., Reising W., Rozenberg G. (1987) Petri Nets: Applications and Relations to other Models of Concurrency. Berlin: Springer, 516 p.

Brayton R. K., Hachtel G. D., Sangiovani-Vincentelli A. et al. (1996) VIS: a system for verification and synthesis //Lecture Notes in Computer Science. – Vol. 1102, pp. 428–432.

Bruck J., Blaum M. (1989) Neural networks, error-correcting codes, and polynomials over the binary n-cube. IEEE Transactions on information theory. Vol. 35, No. 5, pp. 976–987.

Bryant R. E. (1992) Symbolic boolean manipulation with ordered binary decision diagrams // ACM Computing Surveys. Vol. 24, No. 3, pp. 293–318.

Buffa (1984). Meeting the Competitive Challenge: Manufacturing Strategy for U.S. Companies. Dow Jones-Irwin. Competition, International. 190 p.

Buslenko N. P. (1978) Modelling of complex systems. Moscow.

Busse, T. (1998) ERP ousourcing options grow // Infoworld. 1998. Vol. 20, No. 37, 55 p.

Chang E., Pnueli A., Manna Z. (1994) Compositional Verification of Real–Time Systems // Proc. 9’th IEEE Symp. On Logic in Computer Science. pp. 458–465.

Chaochen Z., Ravn A. P., Hoare C. A. R. (1993) An Extended Duration Calculus for Hybrid real-Time Systems // Lecture Notes in Computer Science. Springer-Verlag,. Vol. 736, pp. 36–59.

Chibani, A., Amirat Y., Mohammed S., Matson E., Hagita N., Barreto M. (2013). Ubiquitous robotics: Recent challenges and future trends. Robotics and Autonomous Systems, 61(11), 1162–1172.

Clarke, Edmund M. (2008) The Birth of Model Checking. In: Grumberg, Orna and Veith, Helmut eds.: 25 Years of Model Checking, Vol. 5000: Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp.1–26.

Clarke E., Emerson E. (1981) Design and synthesis of synchronization skeletons using Branching Time Temporol Logic. Lecture Notes in Computer Science. – Vol. 131, pp. 52–71.

Clarke E., Grumberg O., Peled D. (1999) Model Checking. MIT Press, 314 p.

Colombo A. W., Bangemann T., Karnouskos S. (2013) A system of systems view on collaborative industrial automation. In 2013 IEEE International Conference on Industrial Technology (ICIT), pp. 1968–1975.

CPS (2019), CPS-IoT Week 2019, http://www.cpsweek.org/, [accessed in May 2019].

Culler D., Karp R., Patterson D., et al. (1993) LogP: Towards a realistic model of parallel computation // SIGPLAN Notices. No. 7, pp. 1–12.

Delaney B. (2017) Virtual Reality 1.0 -- The 90’s: The Birth of VR, in the Pages of CyberEdge Journal Paperback, 2017. 439 pages.

Dixon M. (2018). What are the most popular plc programming languages? https://realpars.com/plc-programming-languages/

Derhamy H., Eliasson E., and Delsing J. (2016) Iot interoperability-on-demand and low latency transparent multi-protocol translator. IEEE Internet of Things Journal.

Dorf R., Bishop R. (1998) Modern control systems. Addison-Wesley, 1998.

Drusinsky D. (2000) The Temporal Rover and ATG Rover. Proc. Spin2000 Workshop, Lecture Notes in Computer Science. Springer-Verlag, Vol. 1885. pp. 323–329.

East W. Critical Path Method (CPM) Tutor for Construction Planning and Scheduling.

McGraw-Hill Education; 1st edition, 2015, 225 p.

El Mohadab M., Khalene B.B., and Saf S. (2017) Enterprise resource planning: Introductory overview, Electrical and Information Technologies (ICEIT), 2017 International Conference, pp. 1–4.

Electron beam welding machine KL118.00.00.000. (2004) Maintenance manual. – K.: Paton electric welding institute, 443 p.

Emerson E. A. (1990) Temporal and Modal Logic. In Handbook of Theoretical Computer Science, Elsevier Publishers, pp. 1–24.

Emerson E., Trefler R. (1999) Parametric quantitative temporal reasoning. In Logic. In Computer Science, pp. 336–343.

Erl T. (2007) SOA Principles of Service Design (The Prentice Hall Service- Oriented Computing Series from Thomas Erl). Upper Saddle River, NJ, USA: Prentice Hall PTR.

Ferrer B. R., Afolaranmi S. O., and Lastr J. L. (2017) Principles and risk assessment of managing distributed ontologies hosted by embedded devices for controlling industrial systems, in IECON 2017 – 43rd Annual Conference of the IEEE Industrial Electronics Society, pp. 3498–3505.

Flatt H., Schriegel S., Jasperneite J., Trsek H., Adamczyk H. Analysis of the Cyber-Security of industry 4.0 technologies based on RAMI 4.0 and identification of requirements. In IEEE International Conference on Emerging Technologies and Factory Automation, ETFA.

Fleischmann H., Brossog M., Beck M., and Franke J. (2017) Automated generation of human-machine interfaces in electric drives manufacturing, in 2017 7th International Electric Drives Production Conference (EDPC), pp. 1–8.

Ferrer B. Ramis and Lastra J., Martinez L. (2017) Towards the encapsulation and decentralisation of OKD-MES services within embedded devices, International Journal of Production Research, pp. 1–13.

Frey G., Litz L. (2000) Correctness Analysis of Petri Net Based Logic Controllers // Proc. American Control Conference, ACC 2000, Chicago (IL). pp. 3165–3166.

Fuhs H. G. (1995). Applications of the Continuous Acquisition and Life-cycle Support (CALS) initiative to the evolved SEASAPPROW Missile program Monterey, California. Naval Postgraduate School, 79 p.

Gaines B. R., Norrie D. H. (1995) Knowledge Systematization in the International IMS Research Program. // Proc. of IEEE Conference on Systems, Man and Cybernetics Intelligent Systems for 21st Century. Vol. 1. pp. 958–963.

Ganesh K., Mohapatra S., Anbuudayasankar S. P., Sivakumar P. (2014) Enterprise Resource Planning: Fundamentals of Design and Implementation. Springer;182 pages.

Garcia C. E., Prett D. E. and Morari M. (1989) Model predictive control: Theory and practice – a survey. Automatica. (25) pp. 335–348.

Gluch D., Srinivasan G. (1998) A Study of Practice Issues in Model-Based Verification Using the Symbolic Model Verifier (SMV). Technical report. Carnegie Mellon, Software Engineering Institute, 43 p.

Gomi H., Kawato M. (1993) Neural Network control for a closed loop system using feedback error leaning // Neural Neyworks, Vol. 6. pp. 933–946.

Gonzalez R., Woods R. (2018) Digital Image Processing, 4th Edition. |Pearson.

Grädel E., Kolaitis P.vG., Libkin L., et al. (2007) Finite Model Theory and Its Applications. Computer Science Theoretical Computer Science. Texts in Theoretical Computer Science. An EATCS Series, 429 p.

Geunes J. (2017) Operations Planning Mixed Integer Optimization Models. CRC Press, 218 p.

Gunes, V., Peter S., Givargis T., and Vahid F. (2014). A survey on concepts, applications, and challenges in cyber-physical systems. KSII Transactions on Internet and Information Systems, 8(12), 4242–4268. doi: 10.3837/tiis.2014.12.001.

Gurevich Y. (1994) Evolving Algebras 1993: Lipary Guide. Specification and Validation Methods/ Oxford University Presspp. 9–36.

Haber R. E., Juanes C., R. del Toro, and Beruvides G. (2015) Artificial cognitive control with self-x capabilities: A case study of a micromanufacturing process. Computers in Industry, Vol. 74, pp. 135–150.

Hagan M., Demuth H., and Orlando De Jesus (2002). An introduction to the use of neural networks in control systems. International journal of robust and nonlinear control, pp. 959 – 985.

Hamilton K., Watkins D. (2009). Evidence-Based Design for Multiple Building Types. Hoboken, NJ: John Wiley & Sons, Inc.

Harel D., Pnueli A. (1985) On the development of reactive system. In: Logics and Models of Concurrent Systems. Krzysztof R. Apt. Berlin: Spring-Verlag, pp. 477–498.

Hardin R. M., Harel Z., Kurshan R. P. (1996) COSPAN. Lecture Notes in Computer Science. – Vol. 1102, pp. 423–427.

Hartley J. (1984) FMS at Work., Elsevier Science Ltd. - 286 pages.

Hatcliff J., Dwyer M. (2001) Using the Bandera Tool Set to Model–check Properties of Concurrent Java Software. Lecture Notes in Computer Science. Springer–Verlag, – Vol. 2154. pp. 39–58.

Havelund K., Lowry M., Penix J. (2001) Formal Analysis of a Space-Craft Controller Using SPIN. IEEE Transactions on Software Engineering. Vol. 27(8). pp. 749–765.

Havelund K., Rosu G. (2001) Monitoring Java Programs with Java PathExplorer // Proc. of the 1st International Workshop on Runtime Verification (RV'01), Elsevier Science, Electronic Notes in Theoretical Computer Science. No. 55(2), pp. 97–114.

Henzinger T. A. (1991) The temporal specification and verification of Real-Time Systems. PhD thesis, Stanford University, 287 p.

Henzinger T. A., Manna Z., Pnueli A. (1993) Towards Refining Temporal Specification into Hybrid Systems. Lecture Notes in Computer Science. Springer-Verlag, Vol. 736. pp. 60–76.

HLA (High Level Architecture), Release 3.0, AIOTI WG03 – loT Standardisation. European Communities (2017).

Hoare C. A. R. (1985) Communicating Sequential Process. Prentice Hall. 256 p.

Holzmann G. (1997) The model checker Spin . IEEE Trans. on Software Engineering. Vol. 23, No. 5. pp. 279–295.

Iarovyi S., Mohammed W. M., Lobov A., Ferrer B. R., and Lastra J. L. M. (2016) Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems," Proceedings of the IEEE, Vol. 104, No. 5, pp. 1142–1154.

ISA (2020). The International Society of Automation (ISA). Available: https://www.isa.org. Accessed on September 2020.

ISA95, International Society of Automation, Enterprise-Control System Integration. https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95 (Accessed on September 2020).

ISO 10303-1:2021(en). Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles, https://www.iso.org/obp/ui/#iso:std:iso:10303:-1:ed-2:v1:en (Accessed in 2021).

ISO 9000:2015 Quality management systems – Fundamentals and vocabulary, https://www.iso.org/standard/45481.html. Accessed in 2021.

ISO 9001:2015 Quality management systems – Requirements. https://www.iso.org/standard/ 62085.html. Accessed in 2021.

ISO/IEC 31010:2019. Risk management – Risk assessment techniques, https://www.iso.org/standard/72140.html (Accessed in 2021).

Jakobson G., Buford J., Lewis L. (2007). Situation Management: Basic Concepts and Approaches // Information Fusion and Geographic Information Systems. Lecture Notes in Geoinformation and Cartography book series (LNGC), pp 18–33.

Jamshidi M., Ed. (2008) Systems of Systems Engineering, CRC Press, November.

Janicki R., Lauer P. (1992) Specification and Analysis of Concurrent Systems. The COSY Approach. Monographs in Theoretical Computer Science. An EATCS Series.

Jansen G., Gollmar P. (2020) Reactive Systems Explained. O'Reilly Media, Inc.

Jensen K. (1981) Coloured Petri Nets and the Invariant Methods. Theoretical Computer Science. – Vol. 14, pp. 317–336.

Kagermann, H., Wahlster W., and J. Helbig, eds., (2013): Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industry 4.0 Working Group.

Kalachev A. (2013) Multi-core configurable computing platform Zynq-7000. Modern electronics. No. 1, pp. 22–31.

Kaldewaij A. (1986) A Formalism for Concurrent Process. Eindhowen,168 p.

Kang, H. S, Lee J. Y., Choi S., Kim H., Park J. H., Son J. Y., Kim B. H., and Do Noh S. (2016). Smart manufacturing: Past research, presentencing, and future directions. International Journal of Precision Engineering and Manufacturing-Green Technology, 3(1), pp. 111–128.

Karnouskos S., Colombo A. W. (2011) Architecting the Next Generation of Service-based SCADA/DCS System of Systems, in 37th Annual Conference of the IEEE Industrial Electronics Society (IECON 2011), Melbourne, Australia, 7–10 Nov. 2011.

Kazymyr V. V. (2003) Simulation of a synthetic environment for reactive systems. Mathematical modeling. No. 2 (10), pp. 24–32.

Kazymyr V. (2006) Model-Oriented Control of Intelligent Manufacturing Systems: Th. Doctor of Sciences. Kyiv. 301 p.

Kazymyr V, Kondratenko U., Kharchenko V. (2017) University industry cooperation. Volume 4. Capacity, Building, Trainings. TEMPUS CABRIOLET "Model-oriented approach and Intelligent Knowledge-Based System for Evolvable Academia-Industry Cooperation in Electronic and Computer

Engineering” (544497-TEMPUS-1-2013-1-UK-TEMPUS-JPHES), 332 p.

Kazymyr V., Prila O., Usik A., Sysa D. (2019) New Paradigm of Model-Oriented Control in IoT / Information and Software Technologies. Part of the Communications in Computer and Information Science book series, Springer Verlag. CCIS, Vol. 1078, pp. 605–614.

Kazymyr V. V., Sira G. A. (2011) Distributed Modeling in EMS Based on HLA. Mathematical Machines and Systems. No. 4, pp. 125–135.

Kazymyr, V., Shkarlet, S., Zabasta, A. (2020) Practical-oriented Education in Modeling and Simulation for Cyber-Physical Systems. 10th International Conference on Advanced Computer Information Technologies, ACIT 2020 – Proceedings, 2020, pp. 691–694, 9208876.

Kerzner H. (2003) Project Management: A Systems Approach to Planning, Scheduling, and Controlling (8th ed.). Wiley.

Khoussainov B., Nerode A. (2012) Automata Theory and its Applications. Springer Science & Business Media, 432 p.

Khropatyi O., Lohinov O., Kazymyr, V. (2020) Embedded Models Realization Platform in IoT. IDAACS-SWS 2020 – 5th IEEE International Symposium on Smart and Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems, Proceedings, 9297061.

Kim, K. D. and Kumar P. R. (2013). An overview and some challenges in cyber-physical systems. Journal of the Indian Institute of Science, 93(3), 341–352.

Kiran D. (2016) Total Quality Management: Key Concepts and Case Studies. Butterworth-Heinemann; 1st edition, 580 p.

Kramer B., Schmidt H. (1991) Types and Modules for Net Specifications // In K. Jensen and G. Rozeberg, editors, High-Level Petri Nets: Theory and Application. Springer, pp. 171–188.

Kroening D., Strichman O. (2008) Decision Procedures: An Algorithmic Point of View. Springer Science & Business Media, 306 p.

Kruschke J. (2014) Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan. Academic Press; 2nd edition, 776 pages.

Lamport L. (1983) Specifying concurrent program modules. ACM Trans. on Prog. Lang. Syst. No. 5. pp. 190–222.

Lanting, C. J. and Lionetto A. (2015) Smart Systems and Cyber Physical Systems paradigms in an IoT and Industry/ie4. 0 context. Paper presented at the 2nd International Electronic Conference on Sensors and Applications.

Latecki L., Gross A. (1995) Digitization constraints that preserve topology and geometry// In Proc. Intl. Symp. on Computer Vision. pp. 127–132.

Lee E. A. (2007) Computing foundations and practice for cyber-physical systems: a preliminary report, Tech. Rep. UCB/EECS-2007-72, University of California, Berkeley.

Lee J., Bagheri B., Kao H. A. (2015) A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing", 3, 18–23.

Lutz P. (1998) Comparison between the OSACA and OMAC API approaches on an Open Controller Architecture", in: "Open Architecture Control Systems", ITlA Series.

Manna Z., Pnueli A. (1989) The anchored version of the temporal framework. In J.W. de Bakker, W.–P. deRoever, and G. Rozenberg, editors, Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency, Lecture Notes in Computer Science. Springer-Verlag,Vol. 354. pp. 201–284.

McMillan K. L. (1994) Symbolic model checking. Boston, M.A.: Kluwer Academic Publishers, 234 p.

Microsoft Dynamics AX. ERP for big companies and international organizations, https://www.isystems-group.com/solutions/microsoft-dynamics-ax/ (Accessed in 2021).

Milner R. (1989) Communication and Concurrency, Prentice Hall, International Series in Computer Science.

Mordechai Ben-Ari. (2012) Temporal Logic: Formulas, Models, Tableaux. Mathematical Logic for Computer Science, pp. 231–262|.

Model (2020). Structure of the Administration Shell, Apr. 2018. [Online]. Available: https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/structure-of-the-administration-shell.pdf. (Accessed in 2020).

Morkevicius A., Bisikirskiene L., and Bleakley G. (2017) Using a systems of systems modelling approach for developing Industrial Internet of Things applications, in 2017 12th System of Systems Engineering Conference (SoSE), pp. 1–6.

Morozov A. A., Litvinov V. V., Kazymyr V. V. (2003) Adaptive control with models in electron beam welding // Mathematical machines and systemsNo. 3,4, pp.170–180.

Murata T. (1989) Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, Vol. 77, No. 4, pp. 541–580.

Nan Wu and Xiangdong Li (2011). RFID Applications in Cyber-Physical System, Deploying RFID - Challenges, Solutions, and Open Issues. InTech.

Nazarenko O.K., et al. (1993) Observation of the process of electron beam welding and automatic tracking of the joint. Automatic welding. The Paton welding journal. No. 5. pp. 35–38.

Laube U., Nebel M. (2016) Maximum Likelihood Analysis of the Ford–Fulkerson Method on Special Graphs. Algorithmica, Vol. 74, pp. 1224–1266.

Nic, N. (2008) Disruptive civil technologies: Six technologies with potential impacts on US interests out to 2025. Technical Report.

Noe J. D. (1980) Nets in Modeling and Simulation. Brauer W. (ed.) Net Theory and Applications. Berlin: Springer, pp. 347–368.

Nutt G. J. (1972) Evaluation Nets for Computer Systems Performance Analysis. FJCC, AFIPS PRESS. – Vol. 41. pp. 279–286.

Olano, M., Mukherjee, S., Dorbie, A. (2001). Vertex-based anisotropic texturing. Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware. pp. 95–98.

OPC Unified Architecture Interoperability for Industry 4.0 and the Internet of Things, https://opcfoundation.org/ (Accessed in August 2020).

Opanasenko V. N., Kryvyi S. L. (2012) Partitioning the full range of boolean functions based on the threshold and threshold relation. Cybernetics and Systems Analysis. Springer New York Publishers. Vol. 48, No. 3. pp. 459–468.

Oracle E-Business Suite, https://www.oracle.com/uk/applications/ebusiness/ (Accessed in 2021).

Palagin A., Yakovlev Y. (2017) Design Features of Computer Systems on an FPGA Crystal // Mathematical Machines and Systems, No. 2, pp. 3–14.

Palagin A., Opanasenko V., Kryvyi S. (2013) The structure of FPGA-based cyclic-code converters. Optical Memory & Neural Networks (Information Optics). Vol. 22, No. 4. pp. 207–216.

Parker J. R. (1999) Algorithm for Image Processing and Computer Vision // Wiley Computer Publishing. pp. 176–188.

Paton B. E., Nazarenko O. K., Nesterenkov V. M., Morozov A. A., Litvinov V. V., Kazymyr V. V. (2004) Computer control of electron beam welding with multi-coordinate displacements of the gun and workpiece. The Paton welding journal. No. 5. pp. 2–5.

Paulson L. C. (1998) The Inductive Approach to Verifying Cryptographic Protocols. Journal of Computer Security. No. 6. pp. 85–128.

Peled D., Pnualy A. (1994) Proving partial order properties. Theoretical Computer Science.

Vol. 126, pp. 143–182.

Penczek W. (1990) A concurrent branching time temporal logic. Lecture Notes in Computer Science. Vol. 440, pp. 337–354.

Phillips D., Garcia-Diaz A. (1990) Fundamentals of network analysis. Publisher: Prospect Heights, Ill.: Waveland Press.

PNML. http://www.pnml.org/ (Accessed in Marth 2021).

Pnueli A. (1986) Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends. In J. W. de Bakker, W.–P. de Roever, and G. Rozenberg, editors, Current Trends in Concurrency, Lecture Notes in Computer Science. Springer-Verlag, Vol. 224. pp. 510–584.

PMBOK Guide – Sixth Edition (2017), https://www.pmi.org/pmbok-guide-standards/foundational/pmbok (Accessed in 2021).

Pollini L., Innocenti M. (2000) A Synthetic Environment for Dynamic Systems Control and Distributed Simulation. IEEE Control Systems Magazine. –pp. 49–61.

Portico project. http://www.porticoproject.org (Accessed in Marth 2021).

Pritsker A. (1995) Introduction to Simulation and SLAM II. Wiley; 4th edition, 839 p.

Pritschow G. (2001). Open Controller Architecture – Past, Present and Future. CIRP Annals – Manufacturing Technology50(2), -pp. 463–470.

Qin S. J., Badgwell T. A. (1997) An overview of industrial model predictive control technology / In J.C. Kantor, C.E. Garcia, and B. Carnahan, editors, AIChE Symposium Series: Fifth Int. Conf. on Chemical Process Control. Vol. 316. pp. 232–256.

QNX Software Development Platform 6.5.0: Release Notes. (2017) (Accessed http://www.qnx.com/developers/articles/rel_4222_10.html)

Rajashekaran S., Vijayalksmi G. A. (2004) Neural Networks, Fuzzy Logic and Genetic Algorithms. Publisher: Prentice-Hall of India Pvt. Ltd, 456 p.

Razem B. SAP R3 & SAP S/4 HANA. SAP, (2020) (Accessed https://answers.sap.com/questions/13087191/sap-r3-sap-s4-hana.html)

Ravn A. P., Rischel H., Hansen K. M. (1993) Specifying and Verifying Requirements of Real–Time Systems // IEEE Trans. Softw. Eng. – Vol. 19, No. 1, pp. 41–55.

Reising W. (1985) Petri Nets: An Introduction. Springer-Verlag, 161 p.

Romanovs, A., Pichkalov, I., Sabanovic, E., Skirelis, J. (2019). Industry 4.0: Methodologies, Tools and Applications. In: 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream 2019): Proceedings, Lithuania, Vilnius, 25-25 April, 2019. Piscataway: IEEE, 2019, pp.7-10. Available from: doi:10.1109/eStream.2019.8732150.

Sawada C., Akira O. (1997). Open controller architecture OSEC-II: architecture overview and prototype systems // IEEE 6th International Conference on Emerging Technologies and Factory Automation Proceedings, EFTA '97.

Scholten B. (2007) The Road to Integration: A Guide to Applying the ISA-95 Standard in Manufacturing. ISA.

Schulze, Klaus-Rainer. (2007) Electron Beam Technologies. DVS Media, Düsseldorf.

Schweichhart K. (2018) Reference Architectural Model Industrie 4.0 (RAMI4.0), An Introduction, [Online]. Available: https://scholar.google.lv/scholar?q=Reference+Architectural+Model+Industrie+4.0&hl=en&as_sdt=0&as_vis=1&oi=scholart. Accessed on September 2020.

Schyn A., Palanque P., Nedel L.P. (2003) Formal description of a multimodal interaction technique in an immersive virtual reality application, Proceedings of the 15th Conference on l'Interaction Homme-Machine, pp. 150–157. https://dl.acm.org/doi/abs/10.1145/1063669.1063690.

Sheng Z., Mahapatra C., Zhu C., and Leung V. C. M. (2015) Recent Advances in Industrial Wireless Sensor Networks Toward Efficient Management in IoT, IEEE Access, Vol. 3, pp. 622–637.

Silbert N., Hawkins R. (2016) A tutorial on General Recognition Theory. Journal of Mathematical Psychology. Vol. 73, – pp. 94–109.

Simanta S., Morris E., Lewis G., Smith D. (2010) Engineering Lessons for Systems of Systems Learned from Service-Oriented Systems, in. Proc. of IEEE Systems Conference, 4th Annual IEEE.

Skyttner L. (2001) General Systems Theory: Ideas and Applications. University of Gävle, Sweden, 472 p.

Smid P. (2007) CNC Programming Handbook, Third Edition. Industrial Press, Inc., 600 pages.

Staggs K. and et.al., ISA 62443-4-2 security for industrial automation and control systems technical security requirements for IACS components, https://www.isa.org. Accessed on September 2020.

Sultanovs, E., Skorobogatjko, A., Romanovs, A. (2016) Centralized Healthcare Cyber-Physical System's Architecture Development. In: Proceedings of the 2016 57th International Scientific Conference on Power and Electrical Engineering of Riga Technical University, Latvia, Riga, 13–14 October, 2016. Riga: RTU Press, pp. 153–158. Available from: doi:10.1109/RTUCON.2016.7763155

Staggs K. et.al. ISA 62443-4-2 security for industrial automation and control systems technical security requirements for IACS components, https://www.isa.org. Accessed on September 2020.

Tajima K. (1996) Genetic algorithms and their practical application. FUJITSU Sci. Tech. Journ. – Vol. 32, No. 2. pp. 271–286.

Taylor C. et al. (1998) Open, Modular Architecture Controls at GM Powertrain – Definition of OMAC Concept in GMPTG. Control Engineering, https://www.controleng.com/articles/open-modular-architecture-controls-at-gm-powertrain-definition-of-omac-concept-in-gmptg/. Accessed in 2021.

Thiebaux S. et al. (2006) Decision-Theoretic Planning with non-Markovian Rewards. Journal of Artificial Intelligence Research 25, pp. 17–74.

Travica B. (1997) The Design of the Virtual Organization: A Research Model in Gupta, Jatinder N. D., Association for Information Systems Proceedings of the Americas Conference on Information Systems, August 15–17, 1997, Indianapolis, IN, pp. 417–19.

Ungerer G. (2005) uClinux -- Micro-Controller Linux. https://elinux.org/images/b/bb/Uclinux.pdf () (Accessed in Marth 2021).

Vafeiadis T., Ioannidis D., Ziazios C., Metaxa I., and Tzovaras D. (2017) Towards Robust Early Stage Data Knowledge-based Inference Engine to Support Zero-defect Strategies in Manufacturing Environment," Procedia Manufacturing, Vol. 11, pp. 679–685.

Valiant L. G. (1990) A bridging model for parallel computation. Commun. ACM. – Vol. 33, No. 8. pp. 103–111.

Visser W., Pasareanu C., Khurshid S. (2004) Test Input Generation with Java PathFinder. Proc. ISSTA 2004: Int'l SymP. on Software Testing and Analysis. – Boston, MA, Vol. 29, No. 4. pp. 97–107.

Voropayev V., Gelrud Y. (2013) Cyclic stochastic alternative network models for project management. PM World Journal. Vol. II. Issue VIII, pp. 1–18.

Wang F. (1996) Parametric Timing Analysis for Real–Time Systems // Information and Computation, 130(2): 131–150.

Wang Q., Spronck P., Tracht R. (2003). An overview of genetic algorithms applied to control engineering problems // Proceedings of the Second International Conference on Machine Learning and Cybernetics, Xi'an, 2–5 November 2003. pp. 1651–1656.

Wardy M. Y., Wolper P. (1994) Reasoning about infinite computation. Information and Computation. No. 115. pp. 1–37.

Wiest J., Levy F. (2011) Management Guide To Pert/Cpm, A, With Gert/Pdm/Dcpm And Other Networks. Prentice hall.

Wiesner, S., Marilungo E., and Thoben K. D. (2017). Cyber-physical product-service systems — challenges for requirements engineering. International Journal of Automation Technology, 11(1), 17–28. doi: 10.20965/ijat.2017.p0017.

Wooldridge M. (2002). An Introduction to Multiagent Systems. John Wiley & Sons Ltd, 2002. 349 p.

Zabasta, A., Peksa, J., Kondratjevs, K., Kunicina N. (2017). MQTT Enabled Service Broker for Implementation Arrowhead Core Systems for Automation of Control of Utility' Systems. In: 2017 5th IEEE Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE'2017). Piscataway: IEEE, 2017, PP.1-6. doi:10.1109/AIEEE.2017.8270543.

Zabasta, A., Kuņicina N., Kondratjevs K., Patlins A., Ribickis L., Delsing J. (2018). MQTT Service Broker for Enabling the Interoperability of Smart City Systems. International Conference on Energy and Sustainability in Small Developing Economies – ES2DE18. Funchal, Spain, 9 – 11 July 2018. Publisher: Institute of Electrical and Electronics Engineers Inc. pp. 81–87, DOI: 10.1109/ES2DE.2018.8494341.

Zheng P. et al. (2018) Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives, Frontiers of Mechanical Engineering, Review, Vol. 13, No. 2, pp. 137–150.

Zhou, K., Liu T., and Zhou L. (2015) Industry 4.0: Towards future industrial opportunities and challenges. In 2015 12th Int. Conf. Fuzzy Systems and Knowledge Discovery (FSKD), pp. 2147–2152, IEEE.

Zlatanov N. (2016) ARM Architecture and RISC Applications.

Zuburek W. M. (1980) Timed Petri Nets and Preliminary Performance Evaluation. Proc. EEE 7th. Annual. Symp. On Computer Architecture, pp. 88–95.