Hindawi
Journal of Computer Networks and Communications
Volume 2022, Article ID 4952059, 12 pages
https://doi.org/10.1155/2022/4952059

Research Article

Towards Building Cyberphysical Systems with Agile Architecture

Alexander Vodyaho[1], **Nataly Zhukova**[2], **Yulia Schichkina**[1], **Saddam Abbas**[1] and **Vladimir Chernokulsky**[1,3]

[1]*Department of Computer Science and Engineering, Saint Petersburg Electrotechnical University, St. Petersburg 197376, Russia*
[2]*Laboratory of Big Data Technologies in Sociocyberphysical Systems, St. Petersburg Federal Research Centerof the Russian Academy of Sciences, St. Petersburg 199178, Russia*
[3]*Intelligent Systems Laboratory, St. Petersburg 197229, Russia*

Correspondence should be addressed to Nataly Zhukova; nazhukova@mail.ru

Received 13 July 2021; Accepted 18 January 2022; Published 20 February 2022

Academic Editor: Zhiyong Xu

The current stage of technology development is characterized by an increase in the complexity of the created anthropogenic systems, a constant expansion of the scope of information technologies, an increase in the intelligence level of the created systems, and the appearance of new paradigms for building information-oriented systems such as cyber-physical systems, the Internet of things, and cloud and fog systems. Modern information-oriented systems very often have dynamic structure, implement complex adaptive behavior, and can be considered as systems with agile architecture. The article discusses one of the possible approaches for building cyberphysical systems with agile architecture on fog platforms. The idea of the proposed approach is to accumulate knowledge about the current state of the observed cyberphysical systems in the form of knowledge graphs. As a model, it is proposed to use multilevel relatively finite state operating automaton at the upper level and knowledge graphs at the lower level. A distinctive feature of the developed approach is that models that describe the current state of the observed system can be built automatically.

1. Introduction

The progress of engineering and information technologies makes it possible to create very complex anthropogenic systems. The complexity of such new generation of systems results from the increasing of the number of elements, the number of internal and external links, the complexity of behaviour, and the variability of the structure and behaviour of such systems in time.

Modern anthropogenic systems can be defined as complex, multilevel, heterogeneous, distributed, network-centric and data-centric, and intelligent systems. The implementation of these qualities requires the presence of an information processing component, including large-volume information storage and powerful computers. Tallows consider modern anthropogenic systems as Software Intensive Systems (SwIS) [1].

Modern systems consist of elements of different nature, such as mechanical and electromechanical elements; it can also be natural entities, biological and human entities.

Nowadays, one can observe appearance of such new classes of systems as cyberphysical systems (CPS) and sociocyber systems[2, 3].

In other words, one can say that a significant part of complex anthropogenic systems has the property of changing their structure and behaviour depending on the external or internal contexts. Such systems can be defined as variability-intensive systems (VIS) [4, 5]. General trends in the development of anthropogenic systems allow assuming that the systems of next generation will be also VIS.

Solving the problem of variability management for complex, multilevel heterogeneous distributed systems is a nontrivial task, especially when reconfiguration decisions are made autonomously at several levels in different subsystems of a distributed system.

This article discusses one of the possible approaches for solving the problem of managing variability at the architectural level by using the agile architecture approach (AAA). The investigation is conducted in relation to CPS built on fog platforms [6], which are actively used in the construction of modern large-scale systems.

The proposed article includes 13 sections. Section 2 defines such concepts as variability, agility, agile architecture (AA), and their relationship. Section 3 contains the problem statement of the study. Section 4 discusses the current state of the agile architecture systems (AAS) construction technology. Section 5 describes the suggested approach. Section 6 discusses the problem of building an Agile Architecture CPS (AA CPS). Section 7 discusses the main tasks to be solved to support agility in CPS. In Section 8, it is proposed to use the automata representation to describe the mechanisms of agility. Section 9 describes the Agility Support Models (ASM). In Section 10, it is proposed to define 5 maturity levels when building AACPS. Section 11 discusses possible approaches for implementing AACPS. Section 12 provides an example of using AAA. Section 13 contains conclusions on the article.

2. Variability, Agility, and Agile Architecture

Nowadays, many different terms are used to describe the variability of the structure and behaviour of the systems. In relation to information system, for SwIS and for CPS, such terms as adaptive, flexible, reconfigurable, dynamic, and agile are widely used [4, 5, 7].

The term *agility* in Wikipedia is defined as "the ability to change the body's position efficiently" (https://en.wikipedia.org/wiki/Agility). In the IT sphere, this term is also actively used. In relation to SwIS, the term agility can be defined as the ability of a system to adapt to a certain context [5].

There are three main variants of the term agility use: (i) the use in a broad sense as a synonym of such terms as flexibility and adaptability [7], (ii) relation to the SwIS design process [8], and (3) relation to SwIS architectures [4, 5]. It should be noted that the concepts of Agile Process (AP) [8, 9] and AA are correlated to each other, although the mechanisms used are different. AP is a process that is focused on working with constantly changing requirements and belongs to the design stage. AAS are systems which are able to adapt to changes in the runtime context.

Recently, instead of the term agility, the term variability is sometimes used [4, 5, 9]. However, according to the authors of the article, these are different concepts. Variability can be defined as the ability of a system to change its state, and agility can be defined as the ability of a system to respond to a change in state or context.

It should be noted that majority of modern complex systems implement the mechanisms of agility [10, 11].

The subject of consideration in this article is AACPS implemented on fog platforms.

3. Problem Statement

There are several reasons why we deal with the variability of structure and behavior, the main of which are the following: (i) increase in performance by means of system scaling, (ii) increase in reliability (availability) indicators due to the implementation of the self-healing procedure, (iii) support for intelligent interfaces, (iv) restructuring in order to optimize the functioning, and (v) modernization of the system.

The problem of variability management can be formulated as follows. It is necessary to find mechanisms that allow determining the current state of a CPS, to present information in the required form to stakeholders and (or) to form reaction to a change in the structure, behavior of the observed (managed) system (OMS), or a change in the context.

The proposed article considers a possible approach for solving the problem of managing variability at the architectural level for multilevel complex distributed heterogeneous CPS.

4. The State-of-the-Art Agile Architecture Systems

Publications related to AAS can be divided into two groups: publications related to agile architecture systems themselves and publications related to the study of mechanisms that can be useful in implementing AAS. The first group includes early publications on dynamic computers such as [12], publications related to various aspects of adaptation, early publications related to AA [4], and recent publications on the topic [5, 11].

All these works are devoted to the consideration of the general principles of AAS building but contain few information about possible approaches to the agility mechanisms realization. The second group includes studies of general mechanisms that can be useful for AAS building: studies of adaptation mechanisms [7], variability [4, 5], synthesis of automata [13, 14], programs, [15–17] and business processes (BP) [18].

Despite the fact that the concept of AA itself was introduced quite a long time ago, the authors are not aware of works that offer AA models and frameworks.

This article is a further development and generalization of the results of the authors research in the fields of the synthesis of relatively finite automata, the synthesis of knowledge graphs [19, 20], and data acquisition systems in distributed heterogeneous CPS [21].

5. Suggested Approach

The developed AAA is based on the following principles.

(1) The proposed AAA assumes the construction of a multilevel system of models operating in discrete time in terms of discrete states. Each element of the model is a dynamic model of the corresponding architectural element of the model and can be

considered as digital shadow (DSH) of the OMS element.

(2) The proposed AAA is focused on use in VIS CPS.

(3) AAA is considered as one of the possible implementations of the Model-Based System Architecture (MBSA) approach [22].

(4) Models are considered as an independent element of the software architecture. The model is considered as an object with which any actions that can be performed with other objects are allowed. Models can be used at all stages of the life cycle. This article discusses the use of run time models.

(5) ASM can be implemented as objects, services, components, libraries, and loadable modules and in any other way.

(6) The AAA approach is based on the use of ASM, which are built and maintained up to date automatically.

(7) All requests to collect data or change the state of the OMS from potential stakeholders go to the ASM and not directly to the OMS. The model contains all the necessary Data, Information, and Knowledge (DIK) to answer all the requests of all potential stakeholders, which can include both users and subsystems, for example, the agility management subsystem.

The need to move the OMS to a new architectural state (Ast) may be required in the following cases: internal events can occur, such as changes in the structure because of the failure of individual subsystems, the appearance of new subsystems, or suboptimal functioning of the system, when it is necessary, for example, to perform load balancing.

AACPS can be considered as a Context Aware System (CAS) [23]. At the same time, it is necessary to distinguish between static and dynamic agilities. At the development stage, development time agility is implemented, and at the operation stage run time agility is realized.

It should be noted that the proposed approach is close to the Digital Twins approach [24]. It can also be considered as an implementation of the VIS [5] and evolutionary architecture concepts [25] applied to CPS.

6. Agile Architecture CPS

In recent years, CPS has found increasing practical usage and CPS can be considered as one of the possible areas of application of the proposed approach. It should be noted that CPS can be considered as a design paradigm that is based on the architectural approach to design [26]. To date, significant parts of the existing and projected SwIS are systems consisting of components of different physical nature, i.e., CPS. At the same time, it should be noted that the increase of the level of intelligence of created CPS and the increase of usage of ambient intelligence systems (AMIS) [27] are observed.

The Generalized Structure of the CPS. In the most general sense, modern CPS can be defined as multilevel (multilayer) systems, which are systems of the surrounding intelligence. It should be noted that modern CPS can include many thousands of elements.

The generalized structure of the CPS is shown in Figure 1.

In general case, CPS consists of 6 levels (layers): sensor level, fog computing level, cloud computing level, CPS level, CPS systems level, and AmIS. The sensor layer, fog layer, and cloud layer correspond to the levels of the IoT reference model [28]. Many CPS are built on the fog [18] platform, and, for CPS fog, the platform is represented as a set of services. The system can consist of an arbitrary number of CPS that can be integrated at different levels (data, applications, and user interfaces). At each level, the models are described using specialized dictionary. Stakeholders interact with the system through the distributed human machine interface subsystems, which support AMI interface [27]. It is necessary to mention that different groups of stakeholders use different Domain Specific Languages (DSL) [29] to interact with CPS and can have concerns correlated with different levels.

The Main Types of Variability that Occur in CPS. Obviously, when using models in run time mode, it is necessary to track changes in the CPS state. The main types of variability in structure, behavior, and context that may need to be monitored when using the model approach are shown in Figure 2. This variability classification can be conceded as adaptation of [30] to CPS and run time agility.

Variability can manifest itself both in the structure and in the behavior of an OMS. Changes in the structure and/or behavior can occur for a number of reasons. When using dynamic architectures, this is a normal process. Changes in the structure and behavior of the OMS can be associated with the implementation of contextual or content adaptation in order to optimize functioning. Changes in the structure may be due to malfunctions or modernization of the system, in particular, switching on or off the equipment.

Thus, we can assume that models can be used to solve tracking changes in the structure of the OMS, behavior changes, and tasks of tracking user activities.

7. Main Tasks to Be Solved to Support Agility in CPS

The implementation of the proposed approach is reduced for solving the following main tasks: (i) building and maintaining a multilevel model in an adequate state, (ii) building a script that implements the procedure for collecting data about the current state of an OMS, (iii) support DSLs using which different categories of stakeholders can communicate with OMS, and (iv) bringing the OMS in line with the model.

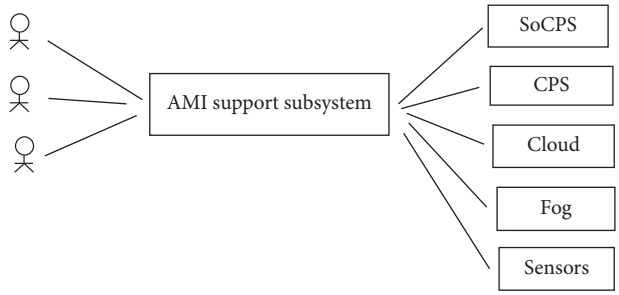

Figure 1: The generalized structure of the CPS.

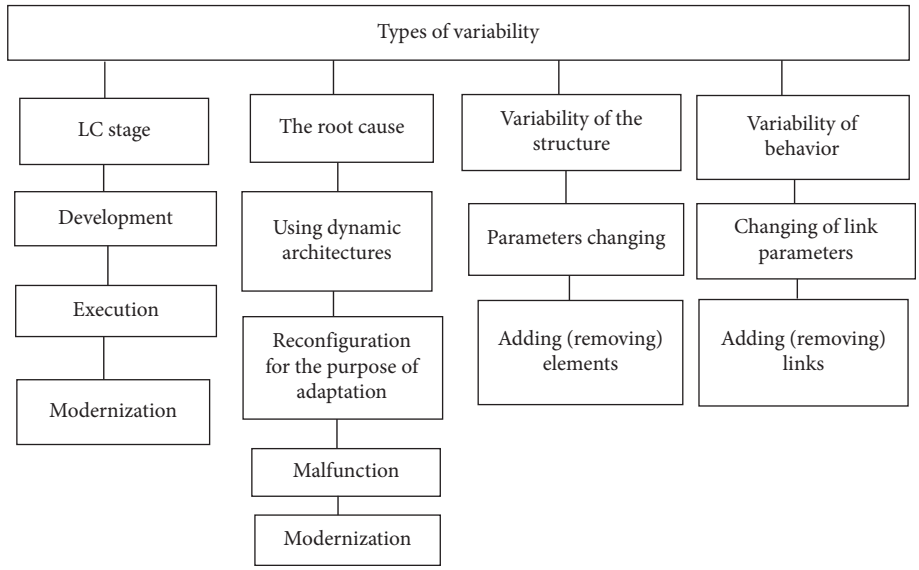

Figure 2: Main types of variability.

8. Automata Representation of AACPS

Formally, AACPS can be defined as: *AACPS* = <*OMS, OMSMM, SH, I1, I2, I3, I4*>, where *OMS* is an observable and managed system, *OMSMM* is agility management module (subsystem), and *SH* is a set of stakeholder groups (not necessarily people) and 4 interfaces. *I1* is the interface for maintaining the relevance of the OMS, *I2* is the interface for supporting the correspondence of the architectural state of the model and the OMS, *I3* is the interface for querying the state of the OMS, and *I4* is the interface for managing the OMS (Figure 3).

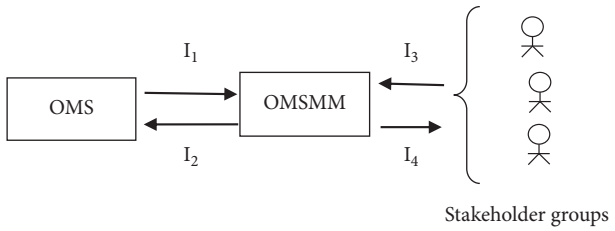

Figure 3: Agile management subsystem.

The following main special cases can be distinguished: (i) the structure and behavior of the OMS being static, (ii) when it is only necessary to monitor the state of the OMS, and (iii) when there is only one user group. In the case when the variability of the structure and behavior is absent or minimal, there is no need to use dynamic models. In this case, models can be built in statics at the design stage based on Model Driven Architecture (MDA) models.

If stakeholders only want to receive data about the state of the OMS, then we are dealing with a DSH [31]. In this case, the control can be carried out via other channels, for example, in manual mode. If there is only one group of stakeholders with the OMS, then models can be built in terms of the corresponding subject area. If there are several different groups of stakeholders who have different concerns, then the task of presenting the model becomes more complicated.

Formally, the interaction of OMS and OMSMM can be represented as interacting automata.

The OMS automaton can be defined as *OMSA* = <*InOMSA, OutOMSA, STOMSA, TROMSA*>, where *InOMSA* is a set of input signals, *OutOMSA* is a set of output signals, *STOMSA* is a set of internal states, and *TROMSA* is a function of transitions and outputs. *InOMSA* is information about events that initiate the transition to a new state. These can be internal events, external events, or commands to change the state (reconfiguration). *OutOMSA*

is, for the most part, logs. *STOMSA* is a set of states of the elements that make up the OMS, and these can be elements of different physical nature. *TROMSA* are the rules, according to which a decision is made to switch to a new state. The rules define FF and they can be of any complexity.

The OMSMM automaton can be described as *OMSMM* = <*InMMA, OutMMA, STMMA, TRMMA*>, where *INMMA* are logs coming from OMSA or commands from users. *OutMMA* is basically the results that are given out according to user requests. *STMMA* is a set of internal states of an abstract automaton. *TRMMA* is a function (table) describing transitions between OMSMM states. This function can be quite complex. The output signals can be associated with both states and transitions. The output signals represent the results on the DSL [29].

In general, OMSMM is a distributed system consisting of a set of interconnected nodes (OMSMMN), each of which can be described as *OMSMMN* = <*SE, AM, I, Srv*> where *SE* is the entity being modeled, *AM* is a model, *I* is an interface with the OMS, and *Srv* are model access services. Interfaces are intended for communication with the underlying layers, and services are intended for communication with the overlying layers.

The simulated entity can have any nature. With rare exceptions, the model is a virtual entity. *I* is interface for communication with the observed object; *Srv* are supported services. The interface *I* can be defined as *I* = <*RQ, ES*>, where *RQ* is a set of requests for data collection, and *ES* is event streams.

The typical structure of the agility support node is shown in Figure 4. The agility support node is a virtual machine that includes the following main elements: (i) Agility Support Models (AM) repository, (ii) AM processor, (iii) DSL processor, (iv) events processor, (v) log request processor, and (vi) control signals generation processor.

The AM repository is designed to store AM files that can be presented in different forms. AM processor is responsible for working with models. Events processor is responsible for processing the flow of events, which are presented in the form of logs (*L*). Log request processor is responsible for the formation and execution of scripts that implement the collection of data necessary for building an AM. This processor sends requests to receive logs (*L*). Control signals generation processor is responsible for generating management actions for OMS reconfiguration.

The functioning of the agility support node can be organized in different ways [21]: (i) the user request processing is implemented within a single process; with each request, the AM is built from scratch; (ii) there is a separate process responsible for keeping the AM up to date; for each specific moment of time there is an up-to-date AM; (iii) a mixed strategy is used; for example, at the upper level, the AM is built in the background mode, and, at the lower level, the model is completed when a specific request appears.

The third option is of real practical interest, since it is quite difficult to store and keep up-to-date complete model of a large CPS. Thus, two parallel processes are implemented in the agility support node: the process of keeping the model up to date and the process which is responsible to realize stakeholder requests.

The algorithm for keeping the model up to date:

(1) Start the OMS monitoring procedure

(2) Receive logs

(3) Clean and sort logs

(4) If AM correction is required, then continue; otherwise go to item 2

(5) AM correction and go to item 2

Algorithm for processing user requests:

(1) Wait for a request from a user

(2) Request acceptance and transformation DSL⟶MQL (Model Query Language, the language of the request to the model)

(3) Request to the model using MQL

(4) If the response is not received, then continue; otherwise MQL⟶ DS transformation and go to item 1

(5) Script synthesis

(6) Script execution

(7) Construction of the required AM of OMS

(8) Request for a new model

(9) MQL⟶ DSL and the transition of the item 1

The task of matching the state of the OMS with AM can be defined as the task of obtaining the OMS with the required ASt according to the model, which is AM. If we take into account that each ASt_i corresponds to exactly one AM_i, then the automaton describing the dynamic structure and behavior, which will be discussed below, can be conceded as a Digital Twins (DT). In this case, each transition is additionally loaded with the reference (Ctr) to the script that is to be executed for transition to this ASt. If for each architectural element of the OMS a kind of a set function is available that can set the element to any valid state and the order of execution does not matter, then the task becomes trivial.

9. Agility Support Models

As a system of models that meet the requirements for ASM, a two-level model *M* = <*MA, MSB*> can be proposed, *MA* is a model of the upper (architectural) level, and *MSB* is a model of the lower (structural-behavioral) level. *MA* describes the OMS in terms of architectural states and *MSB* in terms of structure and behavior. An *MSB* can contain an arbitrary number of nesting levels.

The Top-Level Model. This model describes the behavior of an OMS in terms of the change of ASt under the influence of external or internal events (Ev) and control actions (Ctr), which sets the OMS into this state. Using this model, it is possible to describe the AA [7, 8] in terms of changes in architectural states (Figure 5). The transitions between architectural states can be matched with Fitness Function (FF) [25].

The concept of agility assumes that for one system there are several architectures; in the other words, AA system may

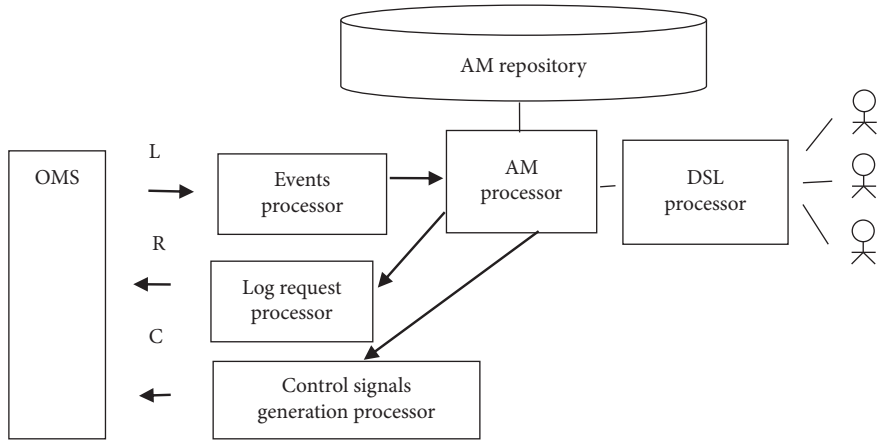

Figure 4: Agility support node.

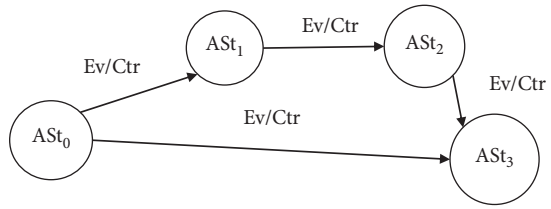

Figure 5: Transitions between ASt.

have several architectural states. The transition between ASt occurs under the influence of both internal and external factors. In the first case, we can talk about the implementation of self ∗ mechanisms (self-testing, self-healing, etc.), and, in the second case, we can talk about CAS [23].

As a model describing the functioning, an AA system it is proposed to use a multilevel relative finite state operational automaton (MLRFSA) [15, 16], which describes the implementation of agility mechanisms in terms of transitions between architectural states (In the Figure 5, Ev denotes events and Ctr denotes the reconfiguration control signals).

The event can be either internal or external. It should be noted that only in the simplest cases the structure of the considered automaton is fully known; most often there is some incomplete knowledge about the structure of the automaton. In this case, it is necessary to solve the problem of an automaton constructing (synthesis). Algorithms of MLRFSA synthesis are described in sufficient detail in the publications of the authors [15, 16].

The automaton according to Figure 5 is a MLRFSA operating in discrete space and discrete time [13, 32]. This is a class of automata in which the sets of acceptable parameters are, in general, finite only at the interval of one step of behavior. At the same time, it is possible to change the set of valid input, internal, and output states of the automaton, as well as the set of valid functions of transitions and outputs of the automaton; i.e., completely rebuild the automaton.

A MLRFSA state can be described by 10 parameters

$$MLRFS_r = \left\{d_{a_r}, d_{b_r}, d_{c_r}, F_r^b, F_r^c, DA\left(d_{b_{r-1}}\right), DB\left(d_{b_{r-1}}\right), DC\left(d_{b_{r-1}}\right), FB\left(d_{b_{r-1}}\right), FC\left(d_{b_{r-1}}\right)\right\}, \quad (1)$$

where d_{a_r} is an input parameter vector; d_{b_r} is an internal state parameter vector; d_{c_r} is an output parameter vector. Functions F_r^b are the functions that describe conditions of the changing automaton internal state and F_r^c are the output functions. The automaton behavior can be described as follows:

$$\begin{aligned} d_{b_{r+1}} &= F_r^b\left(d_{a_r}, d_{b_r}\right), \\ d_{c_r} &= F_r^c\left(d_{a_r}, d_{b_r}\right). \end{aligned} \quad (2)$$

States d_{b_r}, d_{c_r}, and d_{a_r} and functions F_r^b and F_r^c describe automaton at the r-th moment of time and must satisfy the following conditions:

$$\begin{aligned} d_{a_r} &\in DA\left(d_{b_{r-1}}\right), \\ d_{b_r} &\in DB\left(d_{b_{r-1}}\right), \\ d_{c_r} &\in DC\left(d_{b_{r-1}}\right), \\ F_r^b &\in FB\left(d_{b_{r-1}}\right), \\ F_r^c &\in FC\left(d_{b_{r-1}}\right). \end{aligned} \quad (3)$$

The state of the inputs of the automaton at the r-th moment of time is limited by the set $DA(d_{b_{r-1}})$ of valid states defined relatively to the $r-1$ moment of time. The internal state of the automaton at the r-th moment of time must refer to the set $DB(d_{b_{r-1}})$ of its permissible internal states. The possible states of the outputs of the automaton must relate to the set $DC(d_{b_{r-1}})$. The transition function F_r^b implemented by the automaton at the r-th moment must be included in the set of valid functions defined with respect to the $r-1$ moment of time. The set of transition functions $FB(d_{b_{r-1}})$ reflects the system of valid transitions of the automaton at the r-th moment of time. The function F_r^c of outputs at the r-th moment of time must belong to the set $FC(d_{b_{r-1}})$ of valid functions that are active relatively to the $r-1$ moment of time.

The transition from the automaton MLRFSAr to the automaton MLRFSA_{r+1} at $r+1$ moment can be defined as

$$F_r^b:\ OKA_r, d_{a_r} \longrightarrow OKA_{r+1}. \tag{4}$$

The functions of the automaton transitions from one state to another can be also represented as

$$F_r^b\left(d_{a_r}, d_{b_r}\right) \longrightarrow d_{b_{r+1}}. \tag{5}$$

If many functions are used in the transitions, then the transitions are described as follows:

$$F_{zv}\left(d_{zv_e}; e = \overline{1, E_z}\right) \longrightarrow d_{zv_a}; z = \overline{1, Z}; v = \overline{1, V_z}, \tag{6}$$

where z and v define the type of functions used in transitions.

To each of the conditions, their statuses can be assigned: main condition, precondition, and postcondition. For preconditions, numbers of key terms under which they are true can be determined, and postconditions define terms on which they are false. If you set all the elements in the conditions to their corresponding predicates $P_{zv0}(F_{zv0}(g))$ and $P_{zv1}(d_{zv1}), \ldots, P_{zva}(d_{zva})$, which take the value 1 when the variables are defined (true) and 0 otherwise, taking into account the considered relations, the automaton model of the observed object can be described in the form of a structure:

$$MLRFS_r = \left\{F_{zv}\left(d_{zv_e}; e = \overline{1, E_z}\right) \longrightarrow d_{zv_a}; \{d_S\}; \{d_w\}; z = \overline{1, Z}; v = \overline{1, V_z}\right\}, \tag{7}$$

linking $\{d_S\}$ and $\{d_w\}$.

For practical implementation, described MLRFSA are required to have relevant information about the current state of the system, the changes taking place in the structure of the system, and the context; on the basis of this information, a decision on the transition of the system to another architectural state is made.

The MLRFSA synthesis algorithms are considered in detail in [13, 14, 32].

Lower-Level Models. The lower-level models are domain-oriented architectural models and can include several points of view [33]. The type of model used is determined by what DIK stakeholders want to get about the OMS. It should be noted that large CPS are characterized by the presence of several groups of stakeholders with different concerns.

The functioning of SwIS is usually described in terms of structure and behavior [22]. The Structural-Functional Model (SFM) can be used for this purpose. This model is designed to generate models describing the OMS in terms of the structure and implemented behavior in the form of BP. The proposed SFM is a bipartite colored graph and is defined as

$$SPR = \ < OP, L, COL, GO, SEMO, STRM, tk, TK0, CSG, RMS > , \tag{8}$$

where OP is the set of operators, L is the set of arcs, COL is the coloring of arcs, GO are the rules for starting operators, $SEMO$ is the semantics of executing operators, $STRM$ is the strategy mask, tk is the set of tokens, $TK0$ is the initial markup, CSG is the control signal generator, and RMS is the resource monitoring system.

OP operators are operators of any level of complexity; the operators are connected by arcs L, which can be colored in one of 4 colors: arcs, through which Ld data is transmitted, Lc arcs, through which control signals are transmitted that allow the execution of OP, Lz arcs, through which requests for the execution of operators are transmitted, signals, and Lr arcs, through which signals about the availability of resources are distributed.

GO are rules for launching operators:

$$\begin{aligned} GO &= TK\ d\hat{T}Kc\hat{T}Kz\hat{T}Kr, \\ TKi &= tki\hat{S}TRMi, \end{aligned} \tag{9}$$

where tki is the presence of a type i token at the operator input and $STRMi$ is the i-th bit of the $STRM$ mask.

Thus, depending on the mask used, there are 16 different ways to check the readiness of operators for execution. This model can be used as a metamodel. On its basis, it is possible to build various kinds of private models that can describe the OMS in terms of both structure and behavior. A more detailed description of this model and how to build it based on log files is contained in [21]. For automatic construction

of the SFM model, one can use modified Process Mining algorithms [18]. This model is one of the possible models. It should be noted that different models can be used at different levels.

The top-level models are models in terms of which the agility mechanism is implemented, and the lower-level models are, on the one hand, a way to store information about the current state of the OMS and, on the other hand, a source of information for the formation of SFM.

10. AAA Maturity Levels

When solving real life problems, the proposed AAA is not necessarily used in full. One can define 4 levels of AAA maturity.

Level 0. Systems with a Fixed Architecture. The OMS has a fixed architecture, which is created at the design stage. Reconfiguration at the architectural level is not planned at the design stage.

Level 1. Manual (External) Management of OMS Variability. At the design stage, the possibility of reconfiguring the OMS at the architectural level in manual mode is laid. A small number of types of variability are supported.

Level 2. Adaptive Architectures. The OMS is designed according to the principle of a family of architectures. The transition between ASt can be carried out automatically. However, all the ASt are known at the design stage. An architectural automaton can be built in statics at the design stage.

Level 3. Evolutionary Architectures. The OMS is designed according to the principle of an evolutionary architecture. The transition between ASts can be carried out automatically. At the design stage, all the ASts are not known. In this case, both ASts themselves and individual elements may be unknown. The architectural automaton (OMSMM) is built in dynamics.

11. Possible Approaches to Implementation

In general, a large-scale CPS model is a system of model systems (SoM), which can be considered as system of systems [22]. Depending on the type of OMS, models can be organized into the SoM in different ways. Different ways of representing ASM can be used at different levels.

Figure 6 shows two edge variants of building a SoM. Figure 6 shows a variant when all interactions between the system elements are implemented through the use of ASM, and Figure 6, b shows a variant when deferred processing is implemented and the OMS and ASM interact through a telemetric communication channel and through a database (DB). In reality, there may be a large number of intermediate variants that are most often used in practice.

In Figure 6, the following designations are adopted: P is a physical entity, Li is the i-th level of the OMS, Mi is the i-th level model, and DSL is the interpreter of a domain-oriented language.

One can define three main approaches to the implementation of AM: (i) the implementation of the model, i.e., using JAVA in the form of an object model, (ii) based on using ontologies [34], and (iii) based on using knowledge graphs [19, 20, 35, 36]. Using JAVA to build models allows get minimal delays, but it is very expensive in terms of programming. The use of ontologies and knowledge graphs gives slower solutions but allows using existing tools, such as SPARQL [37]. Since the models are quite complex, it is advisable to implement them as cloud services. Individual domain-oriented model fragments can be placed in the fog layer.

Nowadays, it is preferable to build ASM in terms of knowledge, in particular, using ontologies and knowledge graphs. The issues of automatic ASM construction in terms of knowledge graphs are considered in [19, 20].

12. Case Study

As an example, which illustrates the possibilities of using the proposed AAA in practice, one can consider the crane complexes, which is an element of a flexible production system of an automated assembly and welding site. This system, in turn, is a part of a manufacturing enterprise with a high level of automation, planning to implement the concept of Industry 4.0 [38–41].

The existing system is designed to collect and process data from crane systems and is designed for use on production sites, each of which consists of several shop-floors, in which bridge and semicrane cranes work. The cranes are equipped with analog and digital sensors. Analog sensors are used to measure distances between cranes, between trolleys (for double-bed cranes), and between the trolley and the edge of the crane, weight of the load, ambient temperature, motors, temperature of frequency converters, voltage, and current of the power supply network. The discrete sensors include sensors that determine the extreme upper and lower positions of the hook, the condition of the repair gates, the position of the hatch in the cabin, and all fuses in the switch cabinet. Data is collected from all sensors at 250 ms intervals. The total number of measured parameters on each tap is about 560. The total amount of data received from each tap per month is approximately 2 TB.

Based on the data received from the sensors, the tasks of assessing the condition of the cranes and ensuring their operability are solved. In operating mode, different data such as data on the state of the fuses must be collected at a frequency of at least 2 GHz, from the load mass sensors of at least 2 Hz, and from the engine temperature sensors of at least 1 Hz. In the idle mode of the crane, it is necessary to collect information only about the state of the power supply and the ambient temperature. When collecting data, it is necessary to ensure that a delay in data transmission from the cranes to the workplace of the chief mechanic is no more than 5 seconds.

The structure of a preexisting CPS is shown in Figure 7. It is a distributed system that includes equipment installed on cranes, workshop equipment, and main (plant) server. The equipment installed on the cranes includes sensors and actuators (S&A). For data acquisition (DA) subsystem

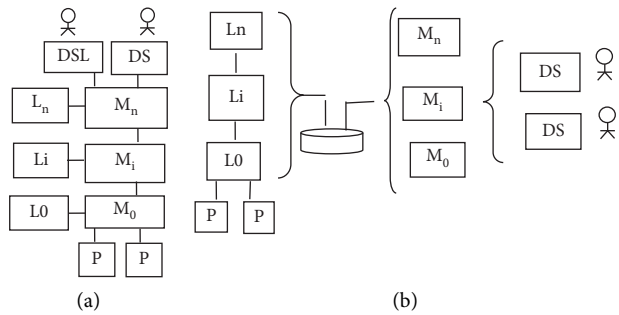

Figure 6: Edge variants of building a SoM.

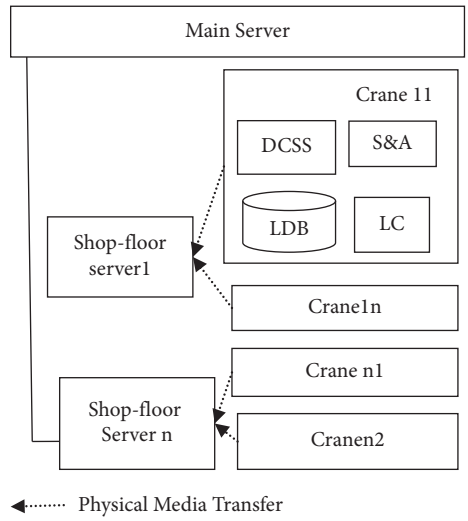

Figure 7: Preexisting system.

(DASS), a local controller (LC) and a local database (LDB) are used. The data acquisition process is divided into two stages: data collection from sensors, which is carried out on the crane, and data transfer from the crane to the central server. The list of data to be collected is fixed.

Data are collected from the sensors in real time. MITSUBISHI MELSEC-Q series industrial controllers installed on the crane are used, providing control of the crane, as well as being a buffer for data collected from sensors. The internal memory of the controllers is accessed via the Seamless Message Protocol. The crane also hosts a single-board ODROID computer for data transfer from LC to ODROID; an Ethernet communication channel with a bandwidth of 100 Mbit/s is used. This channel also transmits commands to the control mechanisms of the crane. The channel provides high quality communication but has limited bandwidth. Overloading the channel can lead to the loss of control commands and cause failures in the operation of the crane. The data are collected by theODROIDC3 single-board computer. ODROID C3 has sufficient computing power to poll the registers at the required frequency. The collected data are placed in the LDB installed on the crane.

Data transfer from the LDB database to the central server can be carried out via a Wi-Fi communication channel or by an operator. The use of the Wi-Fi channel is problematic due to the high level of electromagnetic interference from welding and AC sources installed in the shop-floor. On average, the ratio of the number of lost and transmitted packets is 1 : 5. Data collection by the operator involves the use of the operator's tablet. The software installed on the tablet allows receiving data from the LDB over the radio channel and caching it on the tablet. The range of the radio channels does not exceed 20 meters, which requires the operator to be located near the crane.

This system can be attributed to the maturity level 0, since it is actually a system with a fixed architecture.

12.1. Problems. This CPS was in trial operation for some time, according to the results of which the owners formulated the following problems: (i) insufficient amount of collected data, (ii) long delays in obtaining the data by decision-makers, (iii) low data quality, and (iv) high total cost of ownership (TCO).

The enterprise owners have plans to implement a system that meets the industry 4.0 standards, including usage Digital Twins (DT) technologies. But, nowadays, the list of features which are to be realized is absent. The analysis showed that in order to realize Industry 4.0 ideas it is necessary to achieve at least agility level 3.

In addition, the owners of the system have plans for building and using the corporate knowledge graph, which is also needed to be taken into account when working on the modernization of the system.

A more detailed analysis of these problems showed the following.

The insufficient amount of data is collected due to the following reasons: (i) user requirements for the composition and quality of the data collected are constantly changing; (ii) users cannot say with certainty what data they need in the future.

The reasons for the three remaining problems are the very high level of electromagnetic interference caused by the presence of powerful welding equipment, which does not allow realizing an effective data exchange between the mobile crane and the workshop server. In the existing system, an operator with a tablet goes to the crane and takes data via Wi-Fi. Using this mode leads to unpleasant consequences: (i) there will be large delays in receiving data, while some of the data may become outdated, and (ii) a part of data may be lost due to buffer overflows if the operator with the tablet does not have time to pick up them.

If you want to increase the amount of data being captured, it is necessary to increase the buffer size. The requirement to have a human operator responsible for data collection increases the TCO.

The analyses of these problems have shown that the root cause of the problems is a lack of agility.

Thus, the solution of the problem of improving the efficiency of the system operation was reduced to solving 2 problems: (i) the problem of an agile system organization and (ii) the problem of implementing effective interaction between the crane controller and the workshop server.

To solve these problems, the following main decisions were made: (i) to increase the level of agility, it is proposed to use the model approach described above, (ii) it is proposed to modify the structure of this system and use the concept of edge (fog) computing, and (iii) to solve the problem of excluding the human operator from the data transfer process with a tablet, it is proposed to use clustering mechanisms.

12.2. Suggested Solution. For DA on the crane, an additional intermediate link (fog node) "ARM CPU" was developed, which provides data collection management (Figure 7). The crane additionally houses a controller based on the APM processor, which performs the functions of a fog node. In the repair shop, controllers are also placed, which perform the functions of fog nodes. In order to reduce the volume of collected and stored data, fog nodes use a priori information from the low-level model (LLM) about the object from which the information is collected, the types of sensors are installed on it, and the dynamics of changes in the measured values, i.e., models of the observed objects, are stored. We can assume that the controllers placed on the cranes form the mist level and the controllers placed in the repair areas form the fog level.

The initial scheme assumed measurement of parameter values with a certain sampling step, representation of data in binary form, and their storage in the database. To reduce the data stored in the database, the new scheme detects changes in values, identifies ascending and descending edges, and records the time when these events occur. When collecting data using analog sensors, dynamic changes in the sampling rate are provided depending on the state of the crane. The sampling rate is reduced after the system analyses the rate of change of the measured values over a fixed period of time.

12.3. A Cluster-Oriented Model for Collecting Data from Cranes. To solve the problem of reducing the response time, it is proposed to use the cluster model [42].

The traditional data collection scheme involves direct data transmission using Wi-Fi. The Wi-Fi communication channel in the workshops has low reliability due to the high level of interference. The data transfer rate is inversely proportional to the distance to the access point. Data transfer to the server starts when the threshold value of the accumulated data is reached. With this scheme, the data transfer rate is calculated as $\gamma\ (l) = k/\mathrm{l}$, where k is the gain-transfer rate.

The structure of fog DCS is shown in Figure 8. To ensure fast and reliable DA without the involvement of an operator, a cluster scheme for collecting data from cranes was developed. In each shop-floor, Wi-Fi access points are installed in the repair areas, which act as relay nodes. When a certain amount of collected data is reached on the tap (250 MB), data is transferred from the tap to the head element of the cluster (the crane having the minimum distance to the access point). If there is no possibility of communication with the head element of the cluster (no line of sight), data transmission is carried out through nodes (cranes) located in the visibility zone. Access points are connected to the in-shop network via Ethernet. When the crane is moved to the repair area, the crane data is transmitted to the relay node (Wi-Fi access point). High-level models are realized as distributed knowledge graphs and are located on repair areas, on workshop servers, and on the main server.

As a result of the use of the model approach and the transition to the fog structure, the following problems were solved: (i) providing the ability of expanding the list of collected parameters through the use of high-level model and (ii) the use of low-level model which has halved the amount of data collected.

The use of clustering mechanisms allows solving two problems: (i) reducing the response time for data collection from tens of minutes to units of seconds and (ii) excluding a human operator from the DA process.

The use of these solutions allowed achieving level 2 of the maturity model because only a limited number of models are available.

In the described above example, the benefits obtained are the reduced data access time (reduced response time) and the reduced TCO, since the transition to new equipment is simplified and opportunities for the transition to systems corresponding to the ideas of Industry 4.0 open up.

The further direction of development is associated with the transition to DT technologies [40, 41], while it is planned

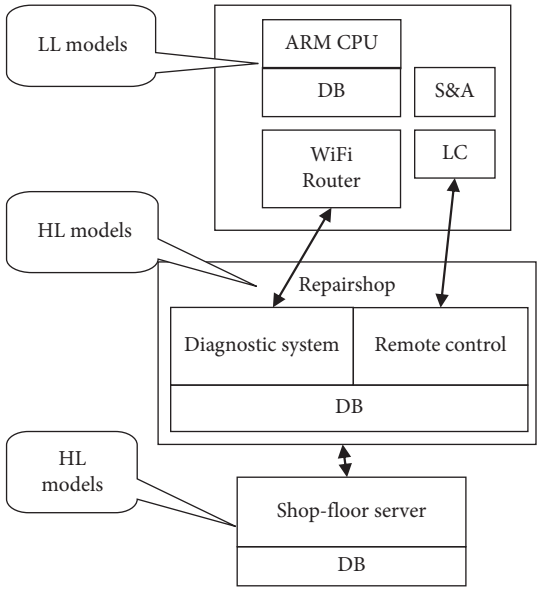

Figure 8: Fog DCS structure.

to actively use the created models. It should be noted that the joint use of Industrial Internet of Things paradigms leads to the need to create dynamic DT, but it is a subject of separate investigations.

13. Conclusion

Using the proposed AAA to CPS implementation allows solving a number of important problems, such as reaching a new level of complexity of the created anthropogenic systems and solving problems of increasing the level of service availability. It should be noted that the paper considers the application of AAA to the construction of the large-scale VIS CPS, while this class of systems is considered as an example of complex heterogeneous systems.

The idea of the proposed approach is that the system has several architectures (architectural states) and it is possible to switch between architectural states. This is a core idea of all agile architecture approaches. Our contribution is that we show how it can be done.

Our approach can also be considered as one of the possible implementations of the DevOps approach to design. Thus, there is an opportunity to get benefits in the TCO terms. Performance gains can be obtained if we consider agility as an approach to reconfiguration. In this case, the gain can be obtained through contextual and (or) content adaptation. The possibility of implementing adaptation mechanisms makes it possible to increase the level of reliability.

A necessary and sufficient condition for using the proposed model approach is a sufficiently high complexity of the structure and behaviour of OMS that have high structural dynamics and adaptive behaviour.

Under these conditions, the use of the model approach reduces the response time to a request for the status of the OMS. In addition, the presence of model knowledge about the past states of the system allows determining the root causes of events and predicting future states of the system to implement proactive management. In addition, the availability of knowledge about past states allows using learning mechanisms.

As the main subject of our future R&D, we see the problem of agile architecture cyberphysical systems security. Nowadays, it is not possible to state unequivocally that agile architecture systems provide a higher level of security compared to traditional architectures. In principle, it is possible to consider the possibility of transition to an architectural state that provides a higher level of security, but this requires assessing the level of security in terms of architecture and building a security model for each architectural state. The authors failed to find adequate approaches in the available sources. We plan to investigate the agile architecture system security and develop architectural tactics for improving security.

Data Availability

The data are gathered from crane complexes, which are an element of a flexible production system of an automated assembly and welding site. Provided data files contain volume of data in GB produced by each of the cranes. The data used to support the findings of this study have been deposited in the repository (https://zenodo.org/record/5109526#.YPfeXXUzY5k).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Ministry of Science and Higher Education of the Russian Federation (agreement no. 075-15-2020-933) dated 13.11.2020 on the provision of a grant in the form of subsidies from the federal budget for the implementation of state support for the establishment and development of the world-class scientific center Pavlov center «Integrative physiology for medicine, high-tech healthcare, and stress-resilience technologies».

References

[1] J. Lattanze Anthony, *Architecting Software Intensive Systems: Practitioner's Guide*, p. 453, Taylor & Francis Group, LLC, Boca Raton, FL, USA, 2009.

[2] R. G. Sanfelice, "Analysis and design of cyber-physical systems. a hybrid control systems approach," in *Cyber-Physical Systems: From Theory to Practice*, D. Rawat, J. Rodrigues, and I. Stojmenovic, Eds., CRC Press, Boca Raton, FL, USA, 2016.

[3] R. C. Calinescu, J. Camara Moreno, and C. Paterson, "Socio-cyber-physical systems: models, opportunities, open challenges," in *Proceedings of the 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems*, Montreal, QC, Canada., May 2019.

[4] R. Capilla, J. Bosch, and K.-C. Kan, *Systems and Software Variability Management*, Springer-Verlag Berlin Heidelberg, Berlin, Germany, 2013.

[5] I. Mistrik, M. Galster, B. Maxim, and B. Tekinerdoga, *Knowledge Management in the Development of Data-Intensive Systems*, Taylor & Francis Group, LLC, Boca Raton, FL, USA, 2021.

[6] "Open fog reference architecture for fog computing," 2021, https://iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf.

[7] P. Tarvainen, "Adaptability evaluation at software architecture level," *The Open Software Engineering Journal*, vol. 2, pp. 1–30, 2008.

[8] "Principles behind the Agile Manifesto," 2021, http://agilemanifesto.org/principles.html.

[9] A. Rasheed, B. Zafar, T. Shehryar et al., "Requirement Engineering Challenges in Agile Software Development," 2021, https://www.hindawi.com/journals/mpe/2021/6696695/.

[10] M. Ali Babar, A. W. Brown, and I. Mistrik, *Agile Software Architecture Aligning Agile Processes and Software Architectures*, Morgan Kaufmann, Burlington, MA, USA, 2014.

[11] J. Bloomberg, *The Agile Architecture Revolution: How Cloud Computing, REST-Based SOA, and Mobile Computing Are Changing Enterprise IT*, Wiley & Sons, Inc., Hoboken, NJ, USA, 2013.

[12] V. Glushkov, M. Ignatyev, V. Miasnikov, and V. Torgashev, "Recursive machines and computing technology," in *Proceedings of the IFIP-74, Computer Hardware and Architecture*, pp. 65–70, Stockholm, Sweden, August 1974.

[13] V. Osipov, E. Stankova, A. Vodyaho, M. Lushnov, Y. Shichkina, and N. Zhukova, "Automatic synthesis of multilevel automata models of biological objects," in *Proceedings of the International Conference on Computational Science and its Applications*, pp. 441–456, Springer Nature AG, Cham, Switzerland, 2019.

[14] V. Y. Osipov, A. I. Vodyaho, N. A. Zhukova, M. Tianxing, and S. Lebedev, "Distributed technical object model synthesis based on monitoring data," *International Journal of Knowledge and Systems Science (IJKSS)*, vol. 10, no. 3, pp. 27–43, 2019.

[15] S. Y. Maslov, *Teoria Deduktivnykh System I Eeprimeneniya (Theory of Deductive Systems and its Applications)*, Radio I Svyaz', Moscow, Russia, in Russian, 1986.

[16] E. K. Tyugu and M. Y. Kharf, "Algorithms for structural synthesis of programs," *Programmirovanie*, vol. 4, pp. 3–13, 1980.

[17] S. Kumaran and M. Quinn, "Towards architecture-adaptable parallel programming," *Scientific Programming*, vol. 6, no. 2, pp. 163–186, https://www.hindawi.com/journals/sp/1997/586912/.

[18] W. Van der Aalst, *Process Mining Data Science in Action*, Springer, Heidelberg. Germany, 2nd edition, 2016.

[19] K. Krinkin, A. Vodyaho, I. Kulikov, and N. Zhukova, "Models of telecommunications network monitoring based on knowledge graphs," in *Proceedings of the 2020 9th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–7, Budva, Montenegro, June 2020.

[20] I. Kulikov, G. Wohlgenannt, Y. Shichkina, and N. Zhukova, "An analytical computing infrastructure for monitoring dynamic networks based on knowledge graphs," in *Computational Science and its Applications–ICCSA 2020. ICCSA 2020. Lecture Notes in Computer Science*, O. Gervasi, Ed., vol. 12254, Cham, Switzerland, Springer, 2020.

[21] A. Vodyaho, V. Osipov, N. Zhukova, and V. Chernokulsky, "Data collection technology for ambient intelligence systems in internet of things," *Electronics*, vol. 9, no. 11, p. 1846.

[22] T. Weilkiens, J. Lamm, S. Roth, and M. Walker, *Model-based System Architecture*, 375 pages, John Wiley & Sons, Hoboken, NJ, USA, 2016.

[23] G. Vert, S. Iyengar, and V. Phoha, *Introduction to Contextual Processing Theory and Applications*, Taylor and Francis Group, LLC, Boca Raton, FL, USA, 2011.

[24] F. Tao, A. Liu, T. Hu, and A. Y. C. Nee, *Digital Twin Driven Smart Design*, Elsevier Inc., London, UK, 2019.

[25] N. Ford, R. Parsons, and P. Kua, *Building Evolutionary Architectures*, O'Reilly Media, Sebastopol, CA, USA, 2017.

[26] P. Marwedel, *Embedded System Design: Embedded Systems, Foundations of Cyber-Physical Systems, and the Internet of Things*, Springer International Publishing, Berlin, Germany, 2018.

[27] Z. Mahmood, *Guide to Ambient Intelligence in the IoT Environment Principles, Technologies and Application*, Springer International Publishing AG, Cham, Switzerland, 2019.

[28] S.-L. Peng, S. Pal, and L. Huang, *Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*, Springer Nature Switzerland AG, Cham, Switzerland, 2020.

[29] M. Fowler, *Domain-Specific Languages*, Addison-Wesley, Upper-Saddle River, NJ, USA, 2014.

[30] M. Galster, D. Weyns, D. Tofan, B. Michalik, and P. Avgeriou, "Variability in software systems—a systematic literature review," *IEEE Transactions on Software Engineering*, vol. 40, no. 3, pp. 282–306, 2014.

[31] M. Grieves, "Digital twin: manufacturing excellence through virtual factory replication," 2014, https://web.archive.org/web/20170517031855/http://innovate.fit.edu/plm/documents/doc_mgr/912/1411.0_Digital_Twin_White_Paper_Dr_Grieves.pdf.

[32] V. Osipov, M. Lushnov, E. Stankova, A. Vodyaho, and N. Zukova, "Inductive synthesis of the models of biological systems according to clinical trials," in *Computational Science and its Applications–ICCSA 2017. ICCSA 2017. Lecture Notes in Computer Science*, O. Gervasi, Ed., vol. 10404, Cham, Switzerland, Springer, 2017.

[33] M. Richards and N. Ford, *Fundamentals of Software Architecture an Engineering Approach*, O'Reilly Media, Inc., Sebastopol, CA, USA.

[34] J. Orozco, *Applied Ontology Engineering in Cloud Services, Networks and Management Systems*, Springer Science+-Business Media, LLC, 2012.

[35] A. Blumauer and H. Nagy, *The Knowledge Graphs Cookbook*, Semantic Web Company, Vienna Austria, 2020.

[36] D. Fensel, U. Şimşek, and K. Angele, *Knowledge Graphs Methodology, Tools and Selected Use Cases*, Springer Nature, Cham, Switzerland, 2020.

[37] B. DuCharme, *Learning SPARQL Querying and Updating with SPARQL 1.1*, O'Reilly Media, Sebastopol, CA, USA, 2013.

[38] S. Patnaik, *New Paradigm of Industry 4.0 Internet of Things, Big Data & Cyber Physical Systems*, Springer Nature, Cham, Switzerland, 2020.

[39] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): an analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, 2018.

[40] F. Tao, M. Zhang, and A. Y. C. Nee, *Digital Twin Driven Smart Manufacturing*, Elsevier Inc., London, UK, 2019.

[41] M. Farsi, A. Daneshkhah, A. Hosseinian-Far, and H. Jahankhani, *Digital Twin Technologies and Smart Cities*, Springer Nature, Cham, Switzerland, 2020.

[42] K. Suwandhada and K. Panyim, "ALEACH-plus: an energy efficient cluster head based routing protocol for wireless sensor network," in *Proceedings of the 2019 7th International Electrical Engineering Congress (iEECON)*, pp. 1–4, IEEE, HuaHin, Thailand, March 2019.