UBICOMM 2022

The Sixteenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies

ISBN: 978-1-61208-989-8

November 13 - 17, 2022

Valencia, Spain

UBICOMM 2022 Editors

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain

UBICOMM 2022

Forward

The Sixteenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2022), held November 13 and November 17, 2022, continued a series of evens meant to bring together researchers from the academia and practitioners from the industry in order to address fundamentals of ubiquitous systems and the new applications related to them.

The rapid advances in ubiquitous technologies make fruition of more than 35 years of research in distributed computing systems, and more than two decades of mobile computing. The ubiquity vision is becoming a reality. Hardware and software components evolved to deliver functionality under failure-prone environments with limited resources. The advent of web services and the progress on wearable devices, ambient components, user-generated content, mobile communications, and new business models generated new applications and services. The conference makes a bridge between issues with software and hardware challenges through mobile communications.

Advances in web services technologies along with their integration into mobility, online and new business models provide a technical infrastructure that enables the progress of mobile services and applications. These include dynamic and on-demand service, context-aware services, and mobile web services. While driving new business models and new online services, particular techniques must be developed for web service composition, web service-driven system design methodology, creation of web services, and on-demand web services.

As mobile and ubiquitous computing becomes a reality, more formal and informal learning will take pace out of the confines of the traditional classroom. Two trends converge to make this possible; increasingly powerful cell phones and PDAs, and improved access to wireless broadband. At the same time, due to the increasing complexity, modern learners will need tools that operate in an intuitive manner and are flexibly integrated in the surrounding learning environment.

Educational services will become more customized and personalized, and more frequently subjected to changes. Learning and teaching are now becoming less tied to physical locations, co-located members of a group, and co-presence in time. Learning and teaching increasingly take place in fluid combinations of virtual and "real" contexts, and fluid combinations of presence in time, space and participation in community. To the learner full access and abundance in communicative opportunities and information retrieval represents new challenges and affordances. Consequently, the educational challenges are numerous in the intersection of technology development, curriculum development, content development and educational infrastructure.

We take here the opportunity to warmly thank all the members of the UBICOMM 2022 technical program committee, as well as all the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to UBICOMM 2022. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

We also gratefully thank the members of the UBICOMM 2022 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope that UBICOMM 2022 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the field of mobile ubiquitous computing, systems, services and technologies. We hope that Valencia provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city

UBICOMM 2022 General Chair

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain

UBICOMM 2022 Steering Committee

Stéphane Galland, Belfort-Montbéliard University of Technology, France
Wladyslaw Homenda, Warsaw University of Technology, Poland
Chunguo Li, Southeast University, China
Dmitry Korzun, Petrozavodsk State University, Russia

UBICOMM 2022 Publicity Chair

Mar Parra, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

UBICOMM 2022

COMMITTEE

UBICOMM 2022 General Chair

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain

UBICOMM Steering Committee

Wladyslaw Homenda, Warsaw University of Technology, Poland
Stéphane Galland, Belfort-Montbéliard University of Technology, France
Chunguo Li, Southeast University, China
Dmitry Korzun, Petrozavodsk State University, Russia

UBICOMM 2022 Publicity Chair

Mar Parra, Universitat Politecnica de Valencia, Spain
Sandra Viciano Tudela, Universitat Politecnica de Valencia, Spain

UBICOMM 2022 Technical Program Committee

Afrand Agah, West Chester University of Pennsylvania, USA
Wafaa Ait-Cheik-Bihi, Itris Automation by Schneider Electric, France
Mehmet Akşit, TOBB ET University, Ankara, Turkey / University of Twente, The Netherlands
A. B. M. Alim Al Islam, Bangladesh University of Engineering and Technology, Bangladesh
Sadam Al-Azani, King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia
Mrim Alnfiai, Dalhousie University, Canada
Tahssin Altabbaa, Istanbul Gelisim University / Huawei Istanbul, Turkey
Nafisa Anzum, University of Waterloo, Canada
Paramasiven Appavoo, University of Mauritius, Mauritius
Mehran Asadi, Lincoln University, USA
F. Mzee Awuor, Kisii University, Kenya
Muhammed Ali Aydin, Istanbul University - Cerrahpasa, Turkey
Nebojsa Bacanin, Singidunum University, Serbia
Chiara Bachechi, University of Modena and Reggio Emilia, Italy
Matthias Baldauf, FHS St.Gallen, Switzerland
Anoud I Bani-hani, Zayed University, Dubai, UAE
Luca Bedogni, University of Modena and Reggio Emilia, Italy
Oladayo Bello, New Mexico State University, Las Cruces, USA
Imed Ben Dhaou, University of Turku, Finland
Imen Ben Lahmar, ISIM Sfax | ReDCAD laboratory | University of Sfax, Tunisia
Djamal Benslimane, Université Claude Bernard Lyon 1, France
Aurelio Bermúdez, Universidad de Castilla-La Mancha, Spain
Javier Berrocal, University of Extremadura, Spain
Nik Bessis, Edge Hill University, UK
Robert Bestak, Czech Technical University in Prague, Czech Republic

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

A Programming Model for Heterogeneous CPS from the Physical Point of View

Martin Richter, Theresa Werner, Matthias Werner
Operating Systems Group
Chemnitz University of Technology
09111 Chemnitz, Germany
email: {martin.richter, theresa.werner, matthias.werner}@informatik.tu-chemnitz.de

***Abstract*—The emergence of Cyber-Physical Systems leads to an integration of the digital and physical world through sensors and actuators. Programming such systems is error-prone and complex as a plethora of changing heterogeneous devices is involved. In existing approaches, the developer views the world from the digital point of view. He or she has to implicitly interpret digital values as sensor measurements of the environment or as control values, which influence the environment through actuators. This leads to an increase of complexity as the number of sensors and actuators in Cyber-Physical Systems is ever-increasing and different types of devices may become available during the runtime of the system. Additionally, the interactions between different types of distributed sensors and actuators have to be coordinated, which increases the likelihood of errors in the programmer's implicit interpretations of the digital values. Current approaches mainly focus on providing abstractions from the distribution and heterogeneity of the system, but fail to explicitly address the impact of digital calculations on the physical world. We present a programming model, which reverses the view of the developer on the system. It allows him or her, to take the perspective of the physical system of interest and to explicitly describe its desired behavior.**

Keywords*—*cyber-physical systems; programming model; context awareness, heterogeneity.

I. Introduction

Because of emerging trends like the Internet of Things [1], Smart Grid [2], automated warehouse logistics [3], and Industry 4.0 [4] an increasing number of devices are interconnected and have access to a multitude of different sensors and actuators in their environment. The emergence of such Cyber-Physical Systems (CPS) leads to an integration of digital computations and the physical world. This entanglement raises multiple challenges, which do not exist in classical distributed systems [5]. Apart from being distributed over space, each of the devices may be connected to a multitude of sensors and actuators. They possess varying capabilities, regarding what they measure and how they influence their environment. As the main goal of the developer is to monitor and control a physical system, he or she has to consider these capabilities when designing his or her application. In classic programming models for CPS, the developer implicitly converts sensor measurements to a digital representation of the physical phenomenon of interest (e.g., reading a value from a register of a sensor). Based on this digital representation, the programmer's application performs calculations of which the results are implicitly converted to impacts on the physical world (e.g., writing a value into a register of an actuator). This procedure increases the difficulty of designing applications. The semantics of controlling actuators and interpreting sensor readings are not always clear with respect to their influence on the physical and digital world, respectively. Our goal is to relieve the programmer from having to convert distributed measurements of physical phenomena to a digital representation and subsequently having to translate digital computations to a variety of actuator influences on the physical environment.

This paper presents a programming model for reducing the complexity of designing applications for heterogeneous CPS. To achieve this, we provide the developer a new view on the system. We reverse the programmer's perspective, such that he or she no longer directly controls the devices through digital computations. Instead, he or she describes the properties of the physical system of interest and how these properties should evolve over time to reach a target state. The developer is concerned with the CPS' effect on the environment (i.e., the desired state change) rather than the cause (i.e., the controlled actuators). As the programmer designs the application from the view of the physical system, he or she does not have to implicitly translate physical phenomena to digital representations and vice versa anymore. Rather, the Runtime-Environment (RTE) handles this conversion transparently by utilizing sensor and actuator specifications in addition to the programmer's physical system and target state descriptions. The RTE maintains a digital representation of the physical system by interpreting sensor measurements. Additionally, it takes advantage of a constraint solver to compute sufficient actuator inputs to reach a target state. These computations are based on the programmer's specification and the digital representation of the system. The RTE chooses a sufficient set of actuators and sensors at each point in time, based on the required physical inputs and outputs to control and observe the system. Hence, our programming model abstracts from complex conversions between digital computations and physical phenomena. Moreover, it provides transparency to the developer with respect to changing device configurations. It is intended to be used in applications utilizing a variable set of arbitrary sensors and actuators to measure and influence physical systems with well-understood properties and dynamics.

As a running example we use a set of robots interacting with a soccer ball. This example offers all the system traits that are of interest to us. There are different sensors and actuators attached to each robot and the system consists of multiple physical objects of interest (i.e., the robots and the ball).

This paper is structured as follows. Section II reviews the related work. Section III describes the concept of our

 ISBN: 978-1-61208-989-8

approach. It depicts the programmer's view on the system and the functionality of the RTE. Section IV supplies a conclusion and an outlook for future work.

II. Related Work

A CPS incorporates the digital and the physical world. The configuration of such heterogeneous distributed systems may change at any point in time due to device failures and the emergence of new types of sensors or actuators. Under such circumstances, programming errors are easily introduced as current solutions rely on the developer to handle the interpretation of physical phenomena through digital computations.

Approaches like *Aggregate Computing* [6] focus on convergence. They enable the developer to write an application for a set of computational nodes situated in a given region. The computations of each node take place on the basis of its local state and its neighbors states. Therefore, the behaviors of the nodes in a region converge over time. Such approaches abstract from the distribution of the system. Nevertheless, they are only suited for homogeneous CPS since a converging node behavior implies that the devices possess similar capabilities.

Other propositions like *Spatial Views* [7] or *Spatial Programming* [8] allow the programmer to control specifically, which part of the code is executed in which region. Additionally, the developer statically specifies, which sensors and actuators are required for the execution of the corresponding parts of the program. Thus, it is not possible for the programmer to take changing types of sensors and actuators into account.

Physical modeling languages like *Modelica* [9] or *Simulink* [10] enable the developer to describe the properties and the behavior of a physical system. These approaches are designed for the simulation of physical systems and for code generation purposes for non-distributed systems. Here, the developer explicitly handles the heterogeneity of the system. The main goal of physical modeling languages is to draw conclusions on the design of a system rather than controlling and observing it directly in a distributed fashion.

Approaches like *Regiment* [11], *Hovering Data Clouds* [12] or *Egocentric Programming* [13] provide mechanisms for the rule-based aggregation and dissemination of environmental data in a distributed CPS. The goal of these propositions is to monitor the environment, rather than to influence it. The programmer therefore has to utilize additional frameworks to describe the desired changes of the physical system state.

The presented solutions tackle challenges like providing distribution transparency or managing heterogeneity. The programmer's main concern still is the management of digital data, which obstructs him or her from focusing the main goal: influencing the physical environment. Our programming model reverses the developer's view on the system. He or she describes the properties and the desired behavior of the physical system from which the RTE deduces the required digital computations while managing a possibly changing set of heterogeneous devices.

III. Concept

Our programming model provides means for the developer to define the properties of a physical system of interest, such that the RTE is able to transparently create a digital representation of it. Apart from that, the programmer is able to specify desired target states for the physical system. This allows the RTE to deduce required actuator actions to cause an appropriate state change. The following sections present a system model to introduce our definition of a physical system. Subsequently, we introduce the developer's view on the system, and an execution model for the RTE.

A. System Model

The programmer desires to influence a physical system through digital computations, such that a certain goal is reached. A physical system Σ is part of the environment and consists of a set of physical objects O. Each object $o \in O$ features a state $\vec{s}_o$, which comprises of multiple properties s_i:

$$\vec{s}_o(t) = \begin{bmatrix} s_1(t) & s_2(t) & \dots & s_n(t) \end{bmatrix}^T \tag{1}$$

Each property s_i is an interpretation g_i of one or more sensor measurements $\vec{m}$ of the environment at each point in time t:

$$s_i(t) = g_i(\vec{m}, t) \tag{2}$$

A measurement comprises of a value in a unit, which can be represented by a combination of possibly multiple SI base units. The set of all sensors makes up the input interface of the CPS. They allow the CPS to recognize physical objects through measurements of the environment. Subsequently, the RTE interprets these measurements to derive the objects' properties. For example, one property of a physical object may be its shape. One way to measure the shape of an object is to interpret the measurements of a digital camera. It transforms the reflected light of the environment (i.e., its wavelength or frequency) into a pixel array, which then can be interpreted to identify the shape of the object.

Properties of physical objects may change over time, which leads to a change of their state $\vec{s}'$. This change of state can be caused by internal dynamics (e.g., a rolling ball) or external influences $\vec{u}(t)$ (e.g., a ball being kicked). External influences are the impact of actuator actions on the properties of a physical object (e.g., a force acting on the ball during the kick). The change of state at each point in time is a function f of the object's state and the corresponding external influences.

$$\vec{s}'(t) = f(\vec{s}, \vec{u}, t) \tag{3}$$

An actuator takes a digital signal as input and transforms it into one or more actions that affect their environment. These actions have measurable impacts on the properties of physical objects. For example, a gripper arm performs the action of grabbing an object. This action can be measured as a force (in Newton) acting on the object from two directions. The set of all actuators makes up the output interface of the CPS. The external influences $\vec{u}_\Sigma$ on the physical system of interest are

the concatenation of the external influences on the different physical objects:

$$\vec{u}_\Sigma = \begin{bmatrix} \vec{u}_{o_1}(t) & \vec{u}_{o_2}(t) & \dots & \vec{u}_{o_m}(t) \end{bmatrix}^T \quad (4)$$

The state $\vec{s}_\Sigma$ of the system is a concatenation of the different physical object states $\vec{s}_{o_i}$:

$$\vec{s}_\Sigma(t) = \begin{bmatrix} \vec{s}_{o_1}(t) & \vec{s}_{o_2}(t) & \dots & \vec{s}_{o_m}(t) \end{bmatrix}^T \quad (5)$$

The change of the state of the physical system $\vec{s}'_\Sigma$ depends on the internal dynamics of the physical objects that populate the system and their external influences. The function f_Σ describes the system's state change:

$$\vec{s}'_\Sigma(t) = f_\Sigma(\vec{s}_\Sigma, \vec{u}_\Sigma, t) \quad (6)$$

We limit our approach to viewing actuator actions as external influences on physical objects. This stands in contrast to regarding any interactions between arbitrary physical objects as external influences. Considering all possible interactions between any physical objects would lead to an explosion in complexity, as there may be an arbitrary number of specified and unspecified physical objects. Instead, we treat interactions between objects as disturbances, which may or may not require counter measures by the CPS.

B. The Programmer's View

In our programming model, the developer views the system from a standpoint of physics. He or she provides specifications for the objects that populate the physical system. These specifications encompass information on the properties of the physical objects (i.e., their state) and a definition of their behavior, based on internal dynamics and external influences. The RTE requires these descriptions to determine, which sensors are necessary for observing the physical objects and how they react to given actuator inputs.

For the RTE to decide, which actions have to be taken by the actuators to reach a target state, a target state description is necessary. This description refers to the whole physical system rather than a single physical object, as relative relationships between physical objects may be of interest to the programmer. The target state description spans a state space, because different states may satisfy the goal of the developer. Table I summarizes the described requirements for the programming interface and for which of the RTE actions they are necessary.

1) Physical Object Specification: Since the physical system consists of possibly multiple physical objects of interest, each of which possess a designated state and behavior, the object-oriented programming paradigm fits the described requirements and system model. A class enables the developer to specify attributes (state) and methods (change of state) of a physical object. From such a class, the RTE creates a digital representation of a physical object whenever it recognizes the corresponding properties of the described object in the environment. If the RTE recognizes multiple objects of the same class, multiple instances are created. As a physical system may encompass a variety of physical objects, the programmer may have to provide multiple different class specifications.

TABLE I. REQUIREMENTS FOR RTE ACTIONS.

ID	Specification Requirement	RTE Actions
Req.1	Physical objects' properties	Recognizing objects and comparing the current system state with the target state space.
Req.2	Physical objects' internal dynamics	Estimate when objects reach the target state through internal dynamics.
Req.3	Physical objects' reactions to external influences	Estimate when objects reach the target state through external influences.
Req.4	Target state description	Calculate actuator actions to reach a target system state.

Through inheritance, a class may extend the state and behavioral descriptions of other classes. This simplifies the description of different types of objects, which partly have a similar state and behavior. For example, a car and a ball both possess the properties of moving objects (i.e., position, velocity, and acceleration) and they also have similar internal dynamics in the sense that their position changes with their velocity and their velocity changes with their acceleration. The specific differences in the behavior and properties of balls and cars are then described in their specific classes respectively, e.g., how external influences affect their positions, velocities and accelerations. Figure 1 shows an example of a ball, which extends the class of a moving object.

The state of an object of a given class is the vector of its attribute values. The value of an attribute is a digital representation of a physical object property, i.e., an interpretation

```
Class MovingObject extends PhysicalObject {
    MovingObject() {
        this.p = Position(?[m],?[m],?[m]);
        this.v = Velocity(?[m],?[m],?[m]);
        this.a = Acceleration(?[m],?[m],?[m]);
    }
    motion(ElapsedTime delta) {
        this.v = this.a + delta * this.a;
        this.p = this.p + delta * this.v;
    }
}

Class Ball extends MovingObject {
    Constructor Ball() {
        this.s = Sphere(radius==30[cm]);
        this.m = Mass(mass=0.3[kg]);
    }
    Requirement(Act(v) == Act(m) AND
                Act(v).position == this.p)
    kick(Velocity v, Mass m) {
        this.v = 1/(this.m + m) * (this.m * this.v +
             m * v + m * 0.8 (v - this.v));
    }
}
```

Figure 1. Example for physical object specifications.

of sensor measurements (see Section III-A). The type of an attribute describes which property it is (e.g., a shape). For a programmer to declare such attributes comfortably, a library provides commonly used classes. Further, the developer specifies the characteristics of the property for the given physical object in the constructor (e.g., the radius of a sphere). Such characteristics allow to distinguish multiple similar physical objects, if necessary. This allows the RTE to identify physical objects reliably and continuously through the available sensor measurements. In Figure 1, a ball is defined as a spherical object with a radius of exactly 30 centimeters. Its position, velocity, and acceleration are unknown and therefore have to be measured by sensors or computed by the RTE, which is syntactically indicated by the corresponding question marks.

The methods of a class describe the state change induced by the physical object's internal dynamics and its reactions to external influences. Each method has access to the object's state and describes a change of its state. A method's calculations may depend on parameters, which represent inputs from actuators. They affect the digital object's state and correspond to external influences on the physical object. For example, in Figure 1, the method `kick` takes the actuator's velocity and its mass as parameters, which influence the velocity of the ball after the impact.

Multiple actuators may provide the inputs to an object's method. This enables the RTE to coordinate a variety of actuator actions for better efficiency or to supply inputs, which a single actuator may not be able to provide. For example, if a building component has to be clamped, two forces on opposite sides of the component have to be at work towards its center. From a result perspective, it does not matter whether this is accomplished by one single actuator or two independent actuators.

Depending on the properties of the physical object and the method parameters, the programmer may have to specify requirements for the inputs from actuators. For example, an actuator has to be close to a component to exert a force on it. The actuator requirements may incorporate the following information:

- the origin of the inputs to the method (e.g., they have to be provided by the same actuator),
- the actuators' states (e.g., their positions), and
- the object's state (e.g., its position).

This allows the RTE to choose actuators capable of influencing the given object and of achieving the desired results.

Figure 1 depicts two methods for the classes `MovingObject` and `Ball`. Method `motion` describes the change of position and velocity, based on the object's velocity and acceleration, respectively. The `delta` parameter stands for the elapsed time between two evaluations of the method. Method `kick` takes two parameters `v` and `m`, which correspond to an actuators velocity and mass, respectively. For this method, requirements for the actuator inputs are given. They specify that both mass and velocity have to be provided by the same actuator and that the actuator has to be situated at the position of the ball. The method calculates the approximate velocity of a ball after being kicked by an actuator with a coefficient of restitution of 0.8.

2) Target State Description: We chose the concept of constraint programming [14] for describing the target state. It allows the developer to specify the properties of a solution to a problem rather than how to reach the solution. This fits our requirements, as the developer describes a desired physical system state and the RTE deduces the sufficient actuator inputs to the physical system. This approach abstracts from the individual actuators. Therefore, the developer is able to focus on the impact of the actuators' actions on the individual physical objects.

The programmer defines target states based on the overall system state, since relations between the different states of physical objects may be of interest. The RTE evaluates the set of constraints periodically in order to analyze whether a target state is reached and whether the system state develops correctly.

Figure 2 shows an example of a defending constraint for a game of robot soccer. The target state refers to the positions of the players of the own team with respect to the ball, goal, and enemy player positions. All players of the own team should be close to an enemy player (i.e., closer than one meter); there should always be one player between the ball and the own goal; there should always be one player between any enemy player and the ball. This positioning allows to intercept the ball, prevents undisturbed passes and enemy attempts to score. To reach this objective, the RTE has to coordinate the available actuators, such that the physical properties of the robots (i.e., their positions and velocities) are changed accordingly.

C. Runtime-Environment

The RTE maintains a set of physical object descriptions, which specify the digital representation of the physical system. It continuously evaluates sensor measurements of the CPS environment to determine the state of the physical objects populating the physical system. Moreover, the RTE continuously evaluates the constraint system for the target state, based on the current state of the physical system. In its evaluations the RTE takes into account the actuator requirements in addition to the available actuators, since they narrow down the available physical inputs.

```
Defense {
  double k, l;

  ∀player, enemy ∈ MyPlayer × EnemyPlayer:
    distance(player.p, enemy.p) <= 1.0[m];

  ∃player ∈ MyPlayer, ∀goal ∈ MyGoal, ∀ball ∈ Ball:
    goal.p + k * (goal.p - ball.p) == player.p;

  ∀ball ∈ Ball, ∀enemy ∈ EnemyPlayer, ∃player ∈ MyPlayer:
    ball.p + l * (ball.p - enemy.p) == player.p;
}
```

Figure 2. Examples of defensive positioning in robot soccer.

Figure 3 depicts the architecture of the RTE. It consists of four modules, which are executed in a distributed fashion: the interpreter, the observer, the controller, and the constraint solver. Furthermore, it facilitates drivers for sensors and actuators. They provide an interface for utilizing the devices and supply information on them to the other modules. The following paragraphs describe the functionalities of the RTE.

1) Interpreter: The interpreter offers an interface to the programmer for registering physical object specifications. It extracts three basic types of information from them:

(i) The state description of the physical object, i.e., what the properties of the object are and how it differs from other objects.
(ii) The behavioral description, i.e., how the object's state changes, based on internal dynamics and external influences.
(iii) The actuator requirements for providing input signals, i.e., what conditions have to be met for an actuator to be able to supply an input to the physical system.

The interpreter creates a vector of state variables $\vec{x}_c$ from the state description of a given class c. The constraint solver is later able to assign values to these variables. Each state variable stands for a physical object's property, i.e., a set of interpreted sensor measurements. The state variables are utilized in the state equation of the physical object. This equation is created from the set of methods M of the given class. Each method $m \in M$ describes a change of state $\vec{x}'_{c,m}$ for an object of the given class. A method may take parameters (external influences caused by actuators). The interpreter converts them to input variables $\vec{u}_m$. Each method describes a change of state, which depends on the state of the object, the specified internal dynamics and reactions to external influences. The function f_m describes this change of state:

$$\vec{x}'_{c,m}(t) = f_m(\vec{x}_c, \vec{u}_m, t) \tag{7}$$

If the function f_m is linear or linearized, the equation can be rewritten as a system of first order differential equations:

$$\vec{x}'_{c,m}(t) = A_m \vec{x}_c(t) + B_m \vec{u}_m(t) \tag{8}$$

If the class' overall behavior is linear or linearized, its state change can be described by the sum of all the methods state changes, as the principle of superposition holds:

$$\vec{x}'_c(t) = \sum_{m \in M} \vec{x}'_{c,m}(t) = \sum_{m \in M} (A_m \vec{x}_c(t) + B_m \vec{u}_m(t)) \tag{9}$$

Since the constraint solver evaluates the constraints periodically in discrete steps, the interpreter converts the described equation into a time discrete variant:

$$\vec{x}_c(k+1) = \sum_{m \in M} (A_m \vec{x}_c(k) + B_m \vec{u}_m(k)) \tag{10}$$

For each new class the interpreter appends the new classes state to the existing system state $\vec{\underline{x}}_\Sigma$. Therefore, a new system state $\vec{x}_\Sigma$ is created (see Section III-A):

$$\vec{x}_\Sigma(k) = \left[\vec{\underline{x}}_\Sigma(k) \quad \vec{x}_c(k)\right]^T \tag{11}$$

Hence, a similar approach is used for the state change equations. The new system state change equation is a concatenation of the old system state change equation and the new classes state change equation:

$$\vec{x}_\Sigma(k+1) = \left[\vec{\underline{x}}_\Sigma(k+1) \quad \vec{x}_c(k+1)\right]^T \tag{12}$$

Moreover, the interpreter creates constraints from the actuator requirements for each method of a class. These constraints allow allocating the available actuator inputs to the corresponding input variables. The target state description is added to the constraint system and restricts the possible system states and system state changes. From the given system of equations, requirements, and constraints, the constraint solver is able to compute a set of actuator inputs, which lead to a target state.

2) Observer: The observer module creates and maintains a digital representation of the state of the physical system. It gathers measurements from the available sensors of the system, similarly to the data aggregation and dissemination process described in [12]. This allows to gather and distribute data of the system, based on given rules (i.e., according to the object specification).

The programmer provides classes, which are specifications of the physical objects encompassing the physical system. The observer recognizes an object of a class, if the sensor measurements relate to the class attributes and the corresponding attribute description in the class' constructor.

The observer interprets the sensor measurements (see Section III-A) based on given rules. These rules describe the relation between physical object properties and the corresponding

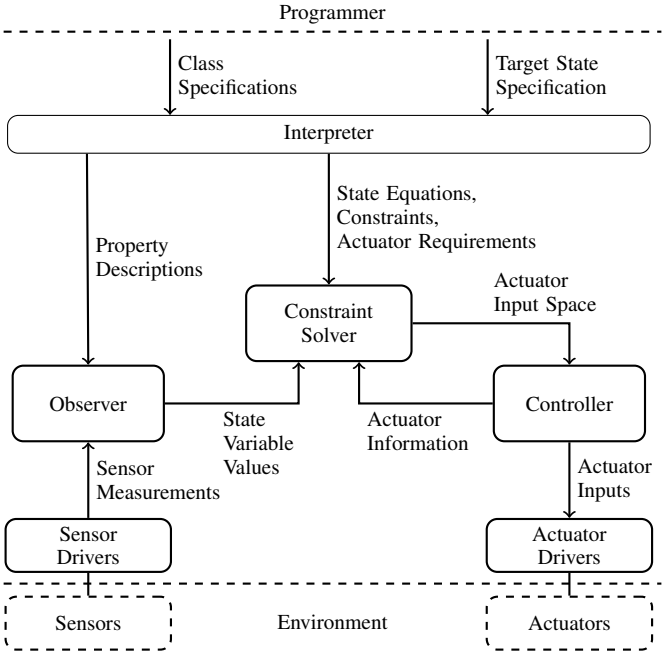

Figure 3. Architecture of the Runtime-Environment.

measurements. The observer maintains all the created object instances by updating their states. These updates are applied whenever new sensor values are available, which relate to the objects' properties. Moreover, the module populates the state variables of the constraint system with the corresponding attribute values (i.e., object states).

3) Constraint Solver: The constraint solver computes a set of sufficient actuator inputs to reach a state of the target state space. As inputs, it takes the system's state equation, the measured current state, the actuator requirements, and information about the currently available actuators. The solver evaluates the constraints periodically to check whether a target state is reached and to update the set of actuator inputs.

Mathematically, the constraint solver's task is to find a point in time t_1, which lies before a given deadline d, and a state trajectory for the system state. The trajectory depends on a set of actuator inputs between the current point in time t_0 and the chosen point in time t_1, such that all constraints hold for the state at time t_1:

$$\vec{x}_\Sigma(t_1) = \vec{x}_\Sigma(t_0) + \int_{t_0}^{t_1} \vec{x}'_\Sigma(t)dt,\ t_1 \leq d \tag{13}$$

4) Controller: The controller module manages the set of available actuators. For each actuator, the controller module maintains information about the actuator's state and which inputs it is able to provide. It offers this information to the constraint solver whenever an evaluation round starts. This enables the constraint solver to evaluate the actuator constraints for determining, which actuators are able to provide the required inputs.

The controller module uses the constraint solver's results (i.e., a set of sufficient inputs to reach a target state) and distributes it to the corresponding available actuators. As the module is executed in a distributed fashion, a consistent view on the available actuators and their information has to be maintained and a consensus for distributing the required inputs has to be found.

IV. Conclusion and Future Work

The presented programming model allows the developer to focus on the description of a physical system and its target state. It allows him or her to specify explicitly what a desired state for a physical system is and how this state changes, based on given inputs and internal dynamics. This abstracts from the need to manage a changing set of actuators and sensors directly, as the required information by the programmer is reduced to defining the influences of actuators on the system and specifying properties of physical objects.

We present a RTE, which encompasses an interpreter, an observer module, a controller module, and a constraint solver. The observer module maintains a digital representation of the physical system's state, based on the physical object descriptions. The interpreter translates the programmer's system specification to a set of constraints and equations such that the constraint solver is able to utilize them. The constraint solver derives target states and required actuator inputs for the physical system from the programmer's specification and the current state of the system. The constraint solver's results are passed to the controller module. It utilizes this data to control the corresponding actuators in order to reach a target state.

The presented programming model abstracts from implicit conversions between digital computations and physical phenomena. Therefore, the physical semantics of a program are made explicit. They are easier to understand and errors in the translation between digital and physical quantities are prevented. Additionally, the RTE transparently handles a set of changing devices as the programmer is concerned with the influences on the physical system of interest, rather than their cause.

For future work, we intend to provide a formal description of sensor and actuator specifications, which allows deducing their properties with regard to how they observe and influence their environment. Further research will be focused on describing the interactions between arbitrary physical objects, which are currently viewed as disturbances. To test the described approach, we will create a prototypical implementation of the RTE. In this regard, we intend to provide verifications of the real-time capabilities of the RTE. Additional research will concentrate on implementing consensus and consistency algorithms for the RTE, as a consistent view of the environment and an optimal utilization of the devices have to be ensured.

References

[1] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Trans. on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] X. Yu and Y. Xue, "Smart grids: A cyber–physical systems perspective," in *Proc. of the IEEE*, vol. 104, 2016, pp. 1058–1070.

[3] F. Basile, P. Chiacchio, and J. Coppola, "A cyber-physical view of automated warehouse systems," in *2016 IEEE Int. Conf. on Automat. Science and Eng. (CASE)*, 2016, pp. 407–412.

[4] N. Jazdi, "Cyber Physical Systems in the Context of Industry 4.0," in *IEEE Int. Conf. on Automat., Quality and Testing, Robotics*, 2014, pp. 103–105.

[5] E. A. Lee, "Cyber physical systems: Design challenges," in *11th IEEE symposium on Object Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2008, pp. 363–369.

[6] M. Viroli *et al.*, "From distributed coordination to field calculus and aggregate computing," *Journal of Logical and Algebraic Methods in Programming*, vol. 109, no. 100486, pp. 1–29, 2019.

[7] Y. Ni, U. Kremer, and L. Iftode, "Spatial views: Space-aware programming for networks of embedded systems," in *Lang.s and Compilers for Parallel Comput.* Springer, 2004, pp. 258–272.

[8] C. Borcea, C. Intanagonwiwat, P. Kang, U. Kremer, and L. Iftode, "Spatial programming using smart messages: design and implementation," in *24th Int. Conf. on Distrib. Comput. Syst.*, 2004, pp. 690–699.

[9] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3*, 2nd ed. John Wiley & Sons, 2014.

[10] E. Hossain, *MATLAB and Simulink Crash Course for Engineers*. Springer, 2022, ch. Introduction to Simulink, pp. 317–359.

[11] R. Newton, G. Morrisett, and M. Welsh, "The regiment macroprogramming system," in *2007 6th Int. Symp. on Inf. Process. in Sensor Networks*, 2007, pp. 489–498.

[12] S. Ebers *et al.*, "Hovering data clouds for organic computing," in *Organic Comput. — A Paradigm Shift for Complex Syst.*, 2011, pp. 221–234.

[13] C. Julien and G.-C. Roman, "Egocentric context-aware programming in ad hoc mobile environments," in *Proc. of the 10th ACM SIGSOFT Symp. on Found.s of Softw. Eng.* ACM, 2002, pp. 21–30.

[14] J. Jaffar and M. J. Maher, "Constraint logic programming: A survey," in *The Journal of Logic Programming*. Elsevier Science Inc., 1994, pp. 503–581.

 ISBN: 978-1-61208-989-8

A Fog Computing Application Using Knowledge-Based Collaborative Sensors - Noise Annoyance Monitoring and Control of a Sun Tracker Use Cases

Joaquin Canada-Bago, Jose-Angel Fernandez-Prieto
Department of Telecommunication Engineering
University of Jaen
Linares, Spain
e-mails: jcbago@ujaen.es, jan@ujaen.es

Ulrich Birkel
Department of Electrical Engineering and Information Technology
Technische Hochschule Mittelhessen
Gießen, Germany
e-mail: ulrich.birkel@ei.thm.de

***Abstract*—Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP) protocols are widely used allowing communication between Internet of Things (IoT) devices and platforms as well as device-to-device communication. However, when they are used in real applications based on Cloud Computing, different problems are observed, such as data loss, lack of security or long communication times between sensors. In this sense, Fog Computing can improve the performance of these applications. The objective of this work is to propose an application based on Fog Computing using knowledge-based devices for two real scenarios: a) control of a solar tracker and b) noise annoyance monitoring. Several experiments have been carried out in order to verify if the application and MQTT and CoAP protocols are appropriate in the system communications of both use cases. The results show that, in the case of noise annoyance monitoring application, this architecture allows reducing the errors in a satisfactory way. However, in control applications, where a communication time between sensors of less than 10 ms is required, the use of these protocols may not be adequate. For this case, an additional client-server software to the Fog Computing system has been developed to be executed in the IoT devices. Although it has lower performance than the widely used protocols, it allows the transmission time to be reduced, and can be satisfactorily applied to the control of systems, such as the control of a sun tracker.**

Keywords; Knowledge-based sensors; fog computing.

I. INTRODUCTION

The Internet of Things (IoT) [1] [2] concept was introduced by Kevin Ashton in 1999 as a system where the physical world is connected to the Internet through ubiquitous sensors. Nowadays, IoT refers to obtaining data and acting in the real world by means of devices with information processing capacity, communications that allow the storage of data on servers located in the cloud and subsequent analysis of stored data.

IoT typical devices have constrained-resources, sensor and actuator capacity, local information process, and are able to communicate data with servers on the Internet cloud. A first classification divides them into devices without an operating system (e.g., Arduino, WaspMote) or with it (e.g., Raspberry).

Regarding IoT communications, data networks (e.g., IEEE 802.15.4, ZigBee, Sigfox, Wireless Sensor Networks (WSN) [3]), network protocols (e.g., LoRa, LoRaWAN [4]), and application protocols (e.g., Message Queuing Telemetry Transport (MQTT) [5], Constrained Application Protocol (CoAP) [6]) have been specifically designed. Although MQTT is classified as a Machine-to-Machine (M2M) protocol, it uses an intermediate server (broker), which introduces a delay that could not be tolerable in distributed control applications.

Currently, the development of IoT is causing that a large number of devices provides huge amounts of data, which has to be stored in servers located in the cloud. Cloud Computing [7] [8] is a technology that allows large-scale computing with the following advantages: virtualized resources, parallel processing, service integration and data storage. In this context, cloud computing systems such as OpenStack [9] and server virtualization environments such as Proxmox Virtual Environment [10] have been proposed. These cloud servers (also called platforms) provide developers Application Programming Interfaces (APIs) and Software Development Kits (SDKs) so that it is possible to establish communication between the IoT devices and the cloud platform. The protocols most commonly used by the platforms are MQTT and HTTP.

Fog Computing [11][12] is a new paradigm, which is based on the transfer of computer and network services from the cloud to the Internet periphery. In this way, IoT devices will use fog servers located very close to them. After that, data are transmitted to cloud servers by fog servers. Some benefits of fog computing are the following:

- Distributed data storage on fog servers.
- Hierarchical data processing in fog servers.
- Quality of Service (QoS) in the data, in order to prioritize the sending of data from delay sensitive applications.
- Performing complex tasks on servers in the fog.
- Uninterrupted services, intermittent access to the cloud would not affect the application.
- Latency reduction since, on the one hand, communications between devices in the fog are faster and, on the other, the volume of data to be sent to the cloud is smaller.

 ISBN: 978-1-61208-989-8

The use of IoT smart devices is widely referenced in the European Commission documents relating to IoT [13] [14] [15]. These documents present devices (smart things) in which algorithms can be executed for intelligent decision based on real-time measurements.

Fuzzy Rule-Based Systems (FRBS) [16] are based on the use of Fuzzy Logic (FL) [17] and express the knowledge through a set of linguistic rules, which are grouped into a Knowledge Base (KB). These systems are correctly adapted to problems in which there is a high degree of uncertainty and imprecision and can be applied to control problems (Fuzzy Logic Controller (FLC)) in which the control algorithm is expressed as rules of action.

A collaborative approach based on FRBS for constrained resource devices is shown in [18][19]. In this approach, each device is able to share its data (e.g., the value of a local variable) with another sensor, with a group of sensors or with all the elements of the system. Therefore, it is conceivable that a device may have local and remote data: local data obtained directly by the device and remote data obtained by other devices and subsequently transmitted to the given device.

The objective of this work is to propose a fog computing collaborative application for the control of a system (sun tracker) and for the noise annoyance monitoring, as well as verify that MQTT and CoAP protocols can be used in the collaborative system communications.

The rest of this paper is organized as follows. Section II describes the fog computing application. Experimental results are provided in Section III. Some conclusions and future work are presented in Section IV. The acknowledgement closes the article.

II. FOG COMPUTING APPLICATION

Fig. 1 shows the proposed system, which is composed of the following components: FRBS collaborative sensors or actuators, direct communication between sensors, communication between sensors and fog computing server, fog computing server, communication between fog and cloud servers and a cloud server.

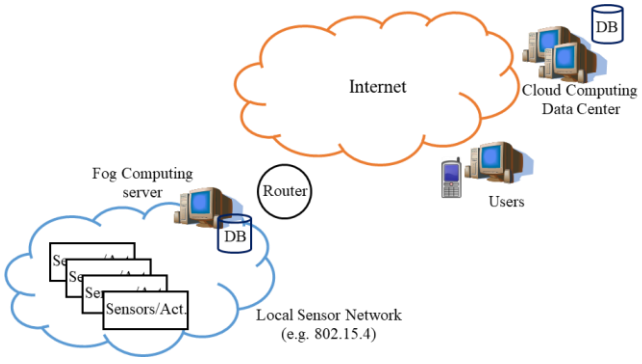

Figure 1. System proposed.

The sensors and actuators used (Fig. 2) are based on a FRBS system and a communications module. The FRBS system allows the sensor to infer the output using a knowledge base.

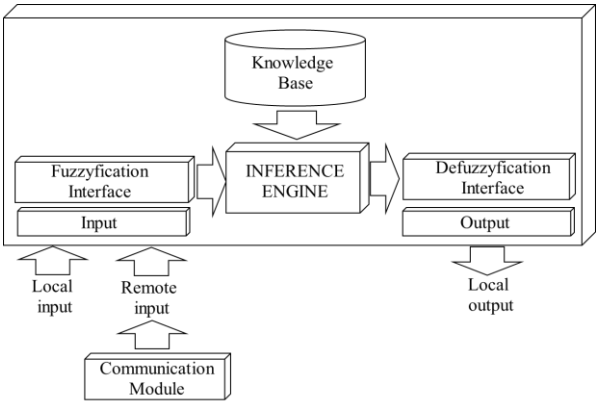

Figure 2. System proposed.

The knowledge-based sensor is based on the structure of the Mandani [20] FRBS, which consists of the following components: a fuzzification interface, a KB, an inference engine and a defuzzification interface.

The fuzzification interface transfers the value of the inputs variable to the fuzzy system. The KB contains the definition of the controller input and output variables, the fuzzy sets defined in the variables, and a set of IF-THEN rules that relate these variables. The inferences engine is responsible for inferring the fuzzy output of the system from the input variables and the knowledge base. Finally, the defuzzification interface adapts the value of the fuzzy output to a real output value.

On the other hand, the communications module allows the sensor to use remote variables obtained by other sensors. Two types of communications are used in this application: direct communication between sensors (Fig. 3) and communication through the fog computing server. Direct communication between sensors allows the sensors to request and obtain the value of a variable obtained by a remote sensor in the fastest way. In this sense, each sensor has a small server that allows the collaborative approach.

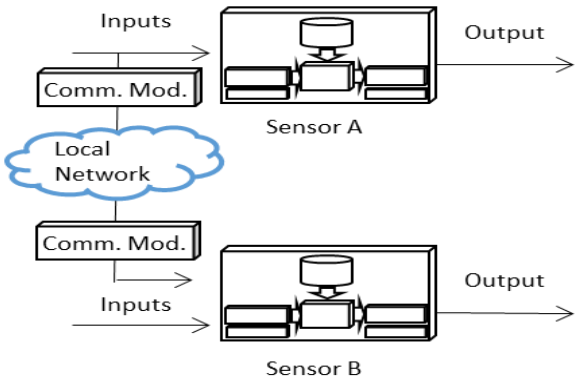

Figure 3. Sensor direct communication.

The main functions of the fog server are: a) local storage of data obtained by all the sensors, b) data communication to

sensors, c) data communication to the cloud platform, and d) increased security of the sensor network. In this way, the measurements and inferences made by the sensors are sent to the fog server and stored in a database. Sensors that need these values can request them from the fog server, although the procedure is slower. Finally, a cloud platform is used for data storage and for Internet users to view and analyze the data obtained.

Note that although the fog computing platform could perform preliminary data analysis processes, in this work, it has not been developed.

III. RESULTS

Several experiments have been carried out in order to verify that the application can be used in both use cases. In the case of the control of the solar tracker, which have the following requirements: a) controllers reaction time must be less than 10 ms; b) five sensors (of which two must be collaborative FRBS controllers) are necessary; c) knowledge bases of up to five variables and d) 15 rules of action per KB.

Fig. 4 shows the test bed installed for the experiments. It consists of a set of Arduino DUE devices with Ethernet shield, an IEEE 802.3 local area network based on a QoS switch, a fog server based on a Raspberry Pi 3B, MySQL database and a cloud platform with HTTP protocol. The software of the Collaborative FRBS sensors, TCP server in sensors, fog server and communications have been developed in C language.

First, the computation time of the tracker control based on a FRBS has been calculated. Using a test KB composed of 5 variables and 15 rules, the microcontroller computes more than 500 inferences per second, which is equivalent to a reaction time of 2 ms. The KB has been modeled through expert knowledge. One of our previous works [21] provides more details about the FRBS system and some inference examples can be observed.

Figure 4. Photograph test bed.

Secondly, the communication time of variables between sensors has been measured using the MQTT and CoAP protocols. Regarding MQTT, the "Arduino Client for MQTT" library and the Mosquitto broker on Raspberry PI have been used. Several tests have been carried out with five devices as QoS1 clients of the broker transmitting simultaneously in two topics. The shortest communication time between sensors has been 67ms. In case of CoAP protocol, the CoAP-simple-library has been used. When performing the experiment under similar conditions to the previous one, a communication time of 30 ms was obtained.

Subsequently, various experiments have been performed using a small TCP server on each device to carry out direct communications between sensors. The server has less functionality than the MQTT and CoAP protocols, allowing exclusively the communication of variables between sensors. In this case, the measured communication times are 5 ms.

Regarding the use of the fog server for the communication of data between sensors, the measured communication times have been much longer due mainly to access to the database.

In the case of the noise annoyance monitoring, in a previous work [22] we presented the design and implementation of a complete low-cost system, composed of nine sensors nodes, for a Wireless Acoustic Sensor Network (WASN) deployed in the city of Linares (Jaén), Spain. The network topology for the proposed complete system is shown in Fig. 5.

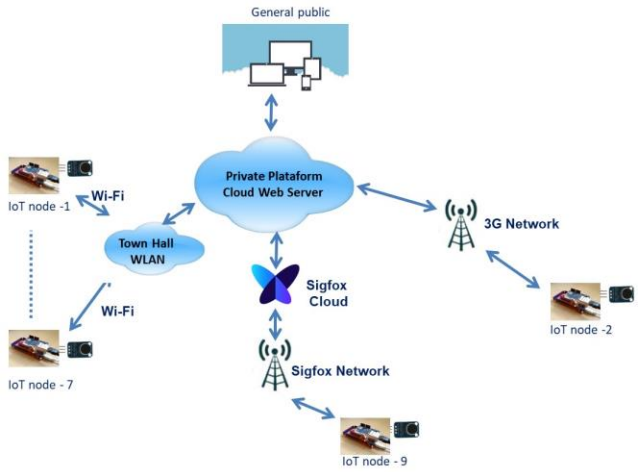

Figure 5. Network topology used for the WASN deployed in the city.

In this real system, a cloud system was used and several problems were encountered, such as: data loss due to the lack of a connection to the cloud, processing of a large volume of data on the server, inability to collect data at certain locations, and inability to use complex knowledge-based systems. To solve these kinds of problems, we have designed and incorporated a fog computing platform between the sensor nodes and the cloud. The fog servers are only in charge of sensed data store and retransmission to the cloud server. A previous work [23] provides more details on how the KB and the FRBS are implemented in the sensors. The final network topology used is shown in Fig. 6.

 ISBN: 978-1-61208-989-8

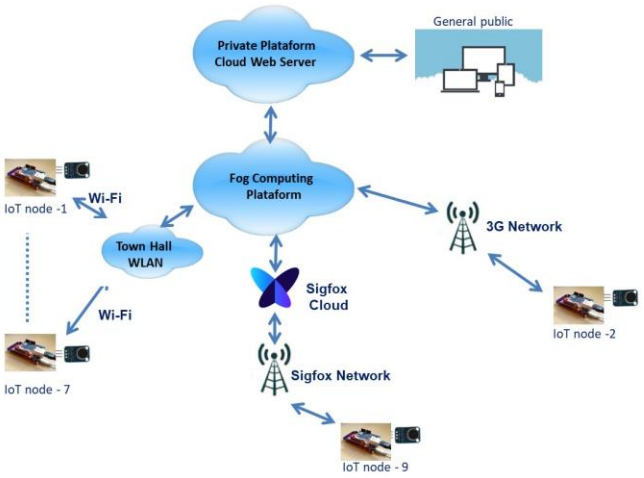

Figure 6. Network topology incorporating the fog server in the WASN deployed in the city.

The fog server has stored the data received from the sensors and retransmitted it to the cloud server. Various tests have been performed with intermittent internet access. In this sense, when recovering communications over the Internet, the data are retransmitted without any loss.

IV. CONCLUSIONS

The device used has sufficient processing capacity to infer the output values using a typical KB to control a sun tracker. The use of the implementations of MQTT protocols and CoAP exceed the maximum reaction time (10 ms) for sun tracker control. Nevertheless, the reaction time is shorter than the maximum by means of the small TCP server and direct communications between sensors.

In the case of the noise annoyance monitoring, since the sensor nodes calculate the sound pressure level every second, and every 30s they send the noise annoyance to the fog server, communication times are not critical in this scenario.

With regard to future work, the following actions are proposed: a) implement the application for tracker control, b) perform complex tasks (which IoT devices cannot execute) on servers in the fog, which increases the capabilities of these applications.

ACKNOWLEDGMENT

This work was partially funded under the Programa Operativo FEDER-Consejería de Economía y Conocimiento de la Junta de Andalucía, Spain, Project Ref. 1380677.

REFERENCES

[1] S. Madakam, R. Ramaswamy, and S. Tripathi, “Internet of Things (IoT): A Literature Review” Journal of Computer and Communicatios, vol. 3, pp. 164-173, 2015.

[2] R. Buyya and A. Dastjerdi, Internet of Thing. Principles and Paradigms. Elsevier, 2016.

[3] H. Karl and A. Willing, Protocols and Architectures for Wireless Sensor Networks. Chichester, West Sussex: Willey, 2007.

[4] Lora Aliance. [Online]. Available from: https://lora-alliance.org/ 2022.08.29

[5] MQTT. [Online]. Available from: http://mqtt.org/ 2022.08.29

[6] IETF. *The Constrained Application Protocol (CoAP)*. [Online]. Available from: https://tools.ietf.org/html/rfc7252 2022.08.29

[7] W. Stallings, Foundations of Modern Networking: SDN, NFV; QoE, IoT and Cloud. Addison-Wesley, 2015.

[8] I. Targio Hashem et al., “The rise of big data on cloud computing: review and open research issues," Information Systems, vol. 47, pp. 98-115, 2015, doi: 10.1016/j.is.2014.07.006.

[9] Openstack. [Online]. Available from: https://www.openstack.org/ 2022.08.29

[10] Proxmox. [Online]. Available from: https://www.proxmox.com/en/ 2022.08.29

[11] H. Freeman and T. Zhang, “The Emerging Era of Fog Computing and Networking,” IEEE Communications Magazine, vol. 4, pp. 4-5, June 2016, doi: 10.1109/MCOM.2016.7497757.

[12] Z. Hao, E. Novak, S. Yi, and Q. Li, “Challenges and Software Architecture for Fog Computing,” IEEE Internet Computing, vol. 21, pp. 44-53, Apr. 2017, doi: 10.1109/MIC.2017.26.

[13] O. Vermesan and P. Friess, Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems. River Publishers, 2013.

[14] O. Vermesan et al. *Internet of Things. Strategic Research Roadmap*. [Online]. Available from: http://www.internet-of-things-research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2009.pdf 2022.08.29

[15] CERP-IoT. *Vision and Challenges for Realising the Internet of Things*. [Online]. Available from: http://www.internet-of-things-research.eu/pdf/IoT_Clusterbook_March_2010.pdf 2022.08.29

[16] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, Genetic Fuzzy Systems: Evolutionary Tunning and Learning of Fuzzy Knowledge Bases, Advances in Fuzzy Systems — Applications and Theory: Volume 19. World Scientific, 2001.

[17] A. Zadeh, “Fuzzy Sets,” Inf. Control, no. 8, pp. 338-353, 1965.

[18] J. Canada-Bago, J. A. Fernandez-Prieto, M. Gadeo-Martos, and J. Velasco, “A New Collaborative Knowledge-Based Approach for Wireless Sensor Networks,” Sensors, vol. 10, no. 6, pp. 6044-6062, 2010, doi:10.3390/s100606044.

[19] M. Gadeo-Martos, J. Fernandez-Prieto, J. Canada-Bago, and J. Velasco, “An Architecture for Performance Optimization in a Collaborative Knowledge-Based Approach for Wireless Sensor Networks," Sensors, vol. 11, pp. 9136-9159, 2011, doi:10.3390/s111009136.

[20] E. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” International Journal of Man-Machine Studies, vol. 7, pp. 1–13, 1975, doi:10.1016/S0020-7373(75)80002-2.

[21] J. Canada-Bago, J. A. Fernandez-Prieto, M. A. Gadeo-Martos, and P. Perez-Higueras, “Knowledge-Based Sensors for Controlling A High-Concentration Photovoltaic Tracker,” Sensors, no. 5: 1315, Feb 2020, doi:10.3390/s20051315.

[22] J. A. Fernandez-Prieto, J. Canada-Bago, and M. A. Gadeo-Martos, “Wireless Acoustic Sensor Nodes for Noise Monitoring in the City of Linares (Jaén),” Sensors, no. 1: 124, Dec 2019, doi:10.3390/s20010124.

[23] J. A. Mariscal-Ramirez, J. A. Fernandez-Prieto, M. A. Gadeo-Martos, and J. Canada-Bago, “Knowledge-based wireless sensors using sound pressure level for noise pollution monitoring,” 11th International Conference on Intelligent Systems Design and Applications, IEEE Press, Nov. 2011, pp. 1032-1037, doi:.10.1109/ISDA.2011.6121794.

 ISBN: 978-1-61208-989-8

Smart Monitoring in Tactile Cyber-Physical Systems

Dmitry Korzun
Department of Computer Science
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
e-mail: dkorzun@cs.karelia.ru

Sergei Marchenkov
Department of Computer Science
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
e-mail: marchenk@cs.petrsu.ru

***Abstract*—We consider the development problem of smart monitoring systems for Internet of Things (IoT) environments. Such systems form a special class of Tactile Cyber-Physical Systems (TCPS) with the essential role of sensorics and artificial intelligence (AI). Sensorics enables the touching sense when a remote object is monitored in Tactile Internet (TI). IoT and AI technologies support near real-time data processing and feedback to perceive and control the object. The applicability of strain gauges is discussed for the needs of emerging applications of smart monitoring in manufacturing, building construction, and vehicle operation. We introduce a multi-layer architecture of TCPS that focuses on the bigdata and smart interaction requirements of industrial monitoring systems. The architecture takes into account the Information&Communication Technologies (ICT) that have already shown efficiency in the industrial settings.**

***Index Terms*—Smart Monitoring, Sensorics, Tactile Internet, Cyber-Physical System, Bigdata Processing.**

I. Introduction

We consider the development problem of smart monitoring systems for Internet of Things (IoT). In particular, such systems are rapidly developed in Industrial Internet or Industrial IoT (IIoT) for the case of machinery equipment monitoring, e.g., see [1]. The data come from multiple sensors that surround the equipment (and its assembly parts—nodes). Monitoring implements functions for recognition of technical state and utilization condition.

Smart monitoring assumes consideration of the following requirements to the system development.

1) *The bigdata requirement (R_{BD}):* Data processing is based on Artificial Intelligence (AI) with advanced methods of Machine Learning (ML) and recognition for Bigdata analytics [2], [3].
2) *The smart interaction requirement (R_{SI}):* System components act as smart IoT objects that interact in IoT environment to construct services using Ambient Intelligence (AmI) [4], [5].

The progress in the IoT/IIoT technology leads to shortening the distance between the human and monitored objects. The results of analytics can be delivered to the user in near real-time. As a result, a person can extend her/his sensory system despite the physical distance (five senses: eyesight, hearing, taste, touch, and smell). In particular, the emerging technology of Tactile Internet (TI) introduced haptic data related to the human perception of objects through its sensory nervous system [6]. This human perception property is considered in Ambient Intelligence (AmI) when human is in a digital environment (e.g., IoT environment) and surrounding devices construct various recognition services by monitoring the physical, informational, and social worlds [4], [7], [8].

The bigdata and smart interaction requirements can be implemented using the concept of Cyber-Physical System (CPS) [9]. CPS components act as data producers (sensing) and consumers (processing) from the physical, informational, and social worlds [4]. Integration of physical and information processes in an AmI environment provides the ability to perceive the environment and its participants based on analysis of the sensed data.

We limit this study with the tactile sense, aiming at monitoring systems that are based on sensing deformations and mechanical stresses. Production machinery (under monitoring) is equipped with various strain gauges, which we elaborated in our previous work [10]. Our target case is Tactile Cyber-Physical System (TCPS).

In this paper, we consider the applicability of strain sensorics in development of IoT monitoring systems, such as manufacturing, building construction, robot movement, wearable sensorics, and vehicle operation. TCPS implements regular measurements (in real-time) of various deformations and mechanical stresses. Application examples and possible strain gauges can be found in [11]–[13]. In particular, magnetic strain sensors and their use are considered in [14].

The key contribution of this paper is the proposal of a multi-layer TCPS architecture for sensed data processing, either in batch mode or near real-time mode. The proposed architecture is based on the latest technologies from IoT, TI, AmI, AI, and Bigdata. The technologies are selected based on our review of existing industrial solutions. Although the architecture is oriented to the specific characteristics of data sensed from various strain gauges, other sensors can be added to TCPS. Easy addition is supported by the layered structure.

The rest of the paper is organized as follows. Section II introduces the problem of applying TCPS in industry, where strain gauges are used to implement the required tactile sense. Section III proposes our multi-layer TCPS architecture for data processing of sensed data, either in batch mode or real-time. Section IV summarizes the key findings of this study.

 ISBN: 978-1-61208-989-8

II. Smart Monitoring with the Tactile Sense

Smart monitoring is widely used in AmI environments, where CPS components act as data producers and consumers from the physical, informational, and social worlds [4]. Integration of physical and information processes in an AmI environment provides the ability to perceive the environment and its participants based on analysis of the sensed data.

In the context of Industry 4.0 and IIoT, a given CPS implements monitoring functions related to recognize equipment technical state and utilization condition, diagnostics and prognosis, making recommendations for maintenance, and selecting optimal operating modes [15]. Such monitoring functions supports analytics of sensed data for diagnostics, prediction, and prescriptive maintenance. In this study, we focus on the monitoring problem machinery equipment.

A monitoring system implements of iterations for manipulating objects (sending commands and receiving feedback). The iterations support construction of the informational and control services. An example of the informational service is predicting technical state of nodes in the machinery equipment (e.g., bearing and their defects [16]). Examples of control service are manipulating effects with feedback in real-time to achieve the desired results [17].

For monitoring and manipulation with real objects, a system is needed for collecting and processing the data from sensors. We focus on tactile sensors, which enhance such a human sense as "touch". The monitoring CPS acts as TCPS [18]. TCPS extend a set of applications and services by combining machine-to-machine and human-to-machine interactions. The following properties are typical for sensorics in a TCPS [11], [12].

- Digitalization of the primary results of measurements.
- Use of many sensors and sensor nodes for monitoring the state of one object as well as processing of the data obtained in parallel from many sensors.
- Correction for noninformative factors (e.g., the influence of temperature on strain sensors).
- Recognition of failures of nodes or communication lines and built-in fault-tolerance capability.
- The sensors used for monitoring are by themselves smart and able to function as nodes of IIoT.
- Wireless connection of the components of the system.
- The ability of the components of the system to communicate with each other in real-time mode.
- High-level characterization of the state of the object under monitoring (e.g., normal or dangerous).
- Recognition of abnormal behavior of the object and making decisions on this base.
- The use of the machine learning (ML) methods for classification of the states of the object under monitoring.
- Flexibility of the system, i.e., possibility to re-configure when necessary.

An example of a TCPS application is remote manipulation of real or virtual robotic industrial equipment in inaccessible and dangerous conditions. In TCPS, an operator remotely controls a robot (e.g., a manipulator) to perform production operations using robotic equipment. Even though the sense of presence can be provided through the exchange of audio/video information, complete immersion is impossible without the exchange of haptic information.

The haptic feedback gives the operator a sense of force, movement, vibration, etc. For example, the operator can adjust the position and grip of the manipulator. The exchange of tactile sense includes commands to the object and feedback from the object. The network round-trip latency in such a loop cannot exceed a few milliseconds to solve the problem of delayed and asynchronous feedback [19]. Reaching the boundary values of delay imposes additional requirements on the development of hardware and software architecture, algorithms, and protocols for TCPS.

Almost any TCPS is characterized by multi-source multi-type data sensing and information exchange followed by data fusing. Strain gauges can serve as an additional source of information on the technical state and utilization condition of production machineries (metalworking machines, gas turbine equipment, presses, pumps, etc.). Multiple data sources provide "redundancy" in measurements. The redundancy can be used to improve the accuracy, reliability, objectivity and validity of technical state assessment and operating conditions. In this regard, measurements of elastic deformations can be based on non-destructive testing methods [20]. This kind of measurements of physical parameters can be used to improve the accuracy of solving a wide class of promising production problems, as we summarized in Table I.

III. Multi-Layer Architecture

We propose the multi-layer TCPS architecture for data processing of sensed data, either in batch mode or near real-time mode. The model is shown in Figure 1. The proposed architecture combines a number of the well-proven technologies used in the digitalization of manufacturing industry. In particular, the bigdata technologies aims at storing and processing huge (in most cases, redundant) sets of continuously arriving sensor data with the possibility of horizontal scaling [3].

A related concept for TCPS is Digital Shadow (the basic component of Digital Twin) [21]. The architecture maintains connections and dependencies (rules) that describe the behavior of a real object under normal operating conditions. Also, any digital object is augmented with additional data collected from the corresponding real object using the IIoT technology [22].

The architecture is based on the data life cycle model "data-information-knowledge-decisions" [2]. The following concept layers are used: (1) physical layer; (2) edge layer; (3) network layer; (4) gateway layer; (5) storage layer; (6) computation layer; (7) analytics layer; (8) service layer.

The physical layer (together with the edge layer) implements the hardware and software enclosure for measuring control around a monitoring and manipulation object—industrial equipment. The measuring control enclosure is created without making any structural changes and does not require taking the

 ISBN: 978-1-61208-989-8

TABLE I
TCPS APPLICATIONS IN INDUSTRY USING STRAIN GAUGES

Application	Use of tactile sensors
1. Remote manipulation of real or virtual objects in inaccessible and dangerous conditions.	Tracking movement and position of human body parts by flexible strain sensors.
2. Monitoring the state of transport vehicles, ship hulls and airframes, wind turbines, railway lines, dams, oil drilling platforms, structural components of bridges and buildings.	Detection of early structural damage based on the analysis of strain measurements; data source in wireless telemetry system; measurement of mechanical resonance frequencies of structures.
3. Design and exploitation of aerospace and aircraft technologies.	Comparison of deformations arising under the action of various forces with the results of CAD (Computer Aided Design) and FEA (Finite Element Analysis) simulations; monitoring the actual stresses in mechanical parts during flight to ensure that it is safe.
4. The control of deformations of parts during processing to adjust the pressing forces by robotic metalworking equipment.	Strain measuring of the part during machine processing by the pressure of the cutter (e.g., during drilling).
5. Measurement of the torque applied by a motor, turbine, etc. to generators, wheels, etc. for optimization of the regime of the equipment	The torque is calculated from the measured strain and the rotational speed on a shaft.
6. Manufacturing of weight and pressure measuring devices for the creation of robotic systems for industrial production.	Strain sensors are the basic (sensing) elements of load cells.

industrial equipment out of operation. The physical layer contains many heterogeneous high intensity sensors and actuators. Industrial automation sensor equipment is used as the main sources of information on the technical state, operation, and operating conditions.

Stresses and strains are the main parameters for monitoring and manipulating the state of objects (including industrial ones), which must withstand dynamic loads. To ensure redundancy of information and coverage of most application scenarios in TCPS, strain measurement should be performed together with such physical parameters as vibration, current, rpm (revolutions per minute), and temperature. For example, using vibration and strain data in integration, one can determine the critical stretch of material and reduce vibration so that this limit is not exceeded. There is also a relationship between shock events and deformation values.

The edge layer ensures that a significant part of the data processing computations is performed close to the data sources and the object under monitoring. Preprocessing data on edge devices increases the performance of upstream digital diagnostics and predictive analytics algorithms by reducing the amount of streaming data and network latency, as well as by distributing the load across edge compute nodes. For edge devices, algorithms must be not only mathematically simple, but also energy efficient to execute them on microcontrollers.

Raw data are collected and presented in a summary form for further time-domain statistics [23]. The summary form for an individual strain data processing stream is determined by a set of such statistical metrics as Root Mean Square (RMS), max, min, crest factor, and kurtosis within a given time window for providing initial, approximate information about faults. We follow the model of [24], where such computation is implemented by so-called sensor computing modules (SCM—data acquisition system instance, DAQ), which can collect data from high-resolution sensors with high-speed measurement. The sensor data are digitized with high precision, preprocessed using basic mathematical transform operations (e.g., Fast Fourier Transform—FFT). SCM uses an external ADC with 24-bit resolution operating (ADS127L01) and a maximum sampling rate of 512 kSPS (Samples Per Second).

A single SCM can connect from 1 to 15 sensors of different types. The sampling rates and duration of sampling are customized along with algorithms to establish an appropriate value for them. In the monitoring system using 10 modules, the daily total volume for the continuous flow of raw and preprocessing data can reach 1.236 TB. In this regard, TCPS faces with Bigdata challenges and specialized technologies and architecture patterns are required to organize storage, stream synchronization, and data processing when designing the overlying architecture layers [3].

Due to the significant requirements for computing resources and the possibility of horizontal scaling, systems for working with Bigdata are developed mainly as distributed systems that implement parallel processing of large data sets [25]. The architecture is with Massive Parallel Processing (MPP) [26]. Many independent computing nodes are connected by a high-bandwidth LAN. A local dedicated server is deployed at the edge. The nodes provide initial data to the local server. The server transmits the processed data further to a data center.

The network layer connects the edge layer and the Bigdata-oriented layers, providing an environment for communication over wireless or wired network channels (Wi-Fi, ZigBee, Ethernet, Bluetooth, RS-485, CAN and similar network protocols). Such protocols as MQTT, CoAP, AMQP, and DDS provides standardized data transfer based on IoT solutions [27]. Nevertheless, proprietary protocols can also be developed to provide lightweight options for polling the DAQ system, requesting a one-time fetch of data from a sensor, and requesting a continuous data acquisition (subscription) [24]. For secure access to both external data sources and a private data center, a virtual private network is used with the creation of dedicated secure circuits and with tunneling protocols [28].

When working with large data sets, the execution is time-consuming for queries to implement the application functions. Many queries cannot be executed in real-time, since they require the execution of algorithms that work with distributed

 ISBN: 978-1-61208-989-8

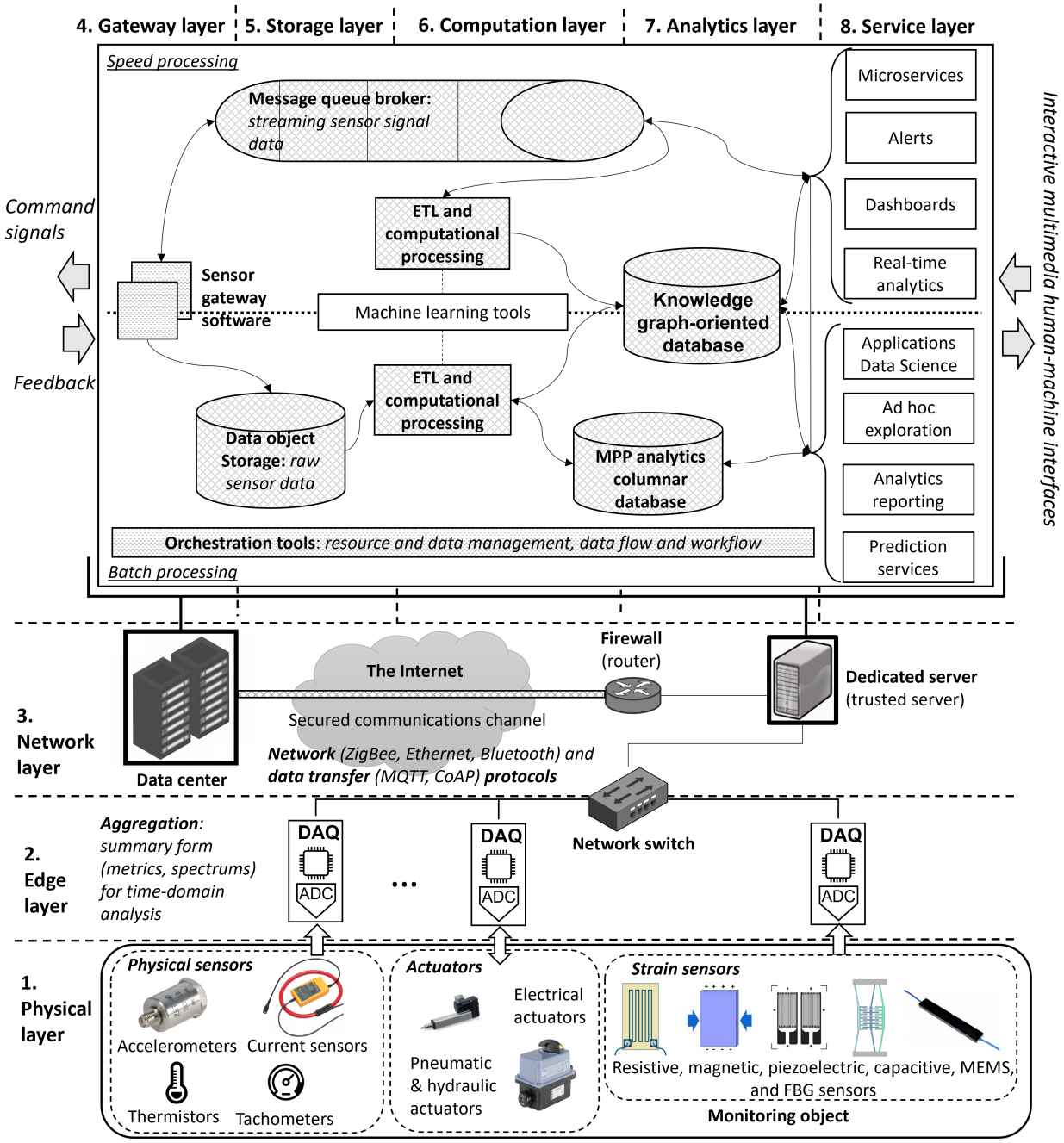

Fig. 1. The multi-layer TCPS architecture model for near real-time intelligent analysis of sensor data.

storages on several nodes of the data center, processing data in parallel. The processing can take several hours, making the results irrelevant. Specialized architectural patterns are needed for obtaining the results in near real-time (likely, with some loss of accuracy) and for complementing them based on executed slow queries over a large set of retrospective data [29].

In particular, the Lambda architecture design pattern separates real-time and batch data processing [30]. The Lambda pattern is applied in many practical industrial applications. Our TCPS architecture (in Figure 1) follows the basic principles of the Lambda pattern. Our model combines the batch processing path (bottom) and the speed processing path (top), so providing a unified, merged view to the service layer. Moreover, the users are not interested to consider these two data streams. The users simply need the analytics results. In particular, if real-time analytics is needed then the accuracy becomes lower, since the result comes from the speed path. Nevertheless, after the completion of complex distributed queries on the batch path, the results obtained by users can be enriched or updated with more accurate information after long-term, deep analytics.

The batch processing path performs heavy-weight, time-

 ISBN: 978-1-61208-989-8

consuming, and resource-intensive queries with no real-time requirements. Free time slots can be used to perform data processing (depending on the schedule, e.g., at night). In this case, reports and various statistics are built only on retrospective data (e.g., for the last day, months, etc.). The result is completed after some time (e.g., tomorrow). However, such results are significantly more accurate and reliable. Note that all incoming data on the batch path is always appended to the existing one, i.e., previous data are not overwritten (by the "log" storage type), allowing storing the history for deeper analytics.

The speed processing path implements near real-time processing. The trade-off is the loss of accuracy with respect to fast ready-made results. On this path, some small data processing functions (e.g., average, some aggregation) operate on individual records in a sensor data stream or in a sliding window mode.

The gateway layer is used to receive, transform, and store information from edge devices and external sources. Control of edge devices is possible by sending configurations to adjust the intensity and content of data streams (depending on the needs of the data processing). According to the data source and destination, the sensed data are forwarded to the batch path (for retrospective data, at rest) or to the speed path (for streaming data, in motion).

For processing streaming data, time synchronization is needed as many streams from various sources. A stream can be aggregated (by types of data sources with reference to a specific equipment node) to provide services in real-time (e.g., diagnostics). Message brokers are used for distributed streaming and data processing (e.g., Apache Kafka, RabbitMQ) [31]. A broker has low latency, since all data processing is executed in-memory, with no slow disk access.

The data-driven interaction between data producers and consumers follows the publish/subscribe model using a message broker [32]. Another equally important task is the creation of a retrospective storage of raw data of physical measurements (batch data) for the further execution of analytical computations and training of neural networks on the results. Software tools designed to automate the transfer of large data batches from data sources to TCPS use the concept of "extract, transform, load" (ETL), e.g., Apache NiFi, Sqoop [25].

The storage layer is critical for the Bigdata requirement in TCPS. Storage and processing of accumulated data from sensors and recognized knowledge are the most resource-intensive operations that uses distributed computing [21], [33]. A MPP architecture is commonly applied to data lakes and databases (or data warehouse) [26], [34]. Data processing is performed by multiple computing nodes. Each node has its own storage and computing resources to resolve a part of the overall data query. Depending on the volume, structure, variety, and variability of data, a storage location is selected.

A data lake is an object storage designed to store various data (structured, semi-structured, or unstructured) in its original, raw form (only the file object and the path to it) without using schemas, types, and data models. A data lake involves storing data in a distributed file system. Hadoop Distributed File System (HDFS) is fault-tolerant and low-cost object storage with low performance when working with small data amounts (file block size from 128MB) [35]. HDFS is suitable for storing samples of raw physical measurements, grouped in large files, e.g., by sampling period (file size depends on the length of the period and the sampling frequency).

A database is suitable for semi-structured and structured data with support for data queries. Compared to a data lake, a database uses a data model that defines how to store, organize, and process data. There are three classes of databases [25], [34], [36]: relational database management systems (RDBMS), NoSQL, and NewSQL databases. RDBMS provide operation of transactional systems using the "online transaction processing" (OLTP) approach (e.g., MySQL, Oracle).

NoSQL databases emerged as an alternative to traditional relational databases with a tabular data organization format (e.g., MongoDB, Neo4j). Depending on the problem being solved, NoSQL databases offer the following set of fundamental data structures: wide column, document, key-value pair, or graph.

NewSQL databases emerged as a combination of the NoSQL and RDBMS advantages. They solve the ACID problems (atomicity, consistency, isolation, durability) with horizontal scalability of OLTP databases (e.g., Clustrix, NuoDB). Therefore, RDBMS and NewSQL databases focus on transactional loads. They are rarely used in TCPS, since the key issue is the analytics of large sets of time series from physical parameters measurements and predicting the technical state of machinery equipment [25].

Nevertheless, columnar NoSQL databases are traditionally designed to support business analytics (e.g., Vertica, ClickHouse) [37]. Typical use is a large data warehouse solving analytical problems using the "online analytical processing" (OLAP) approach. An analytical database is at a level higher than a data lake. Data are further processed (partitioning, compression) to make analytics easy and fast. In turn, graph-oriented NoSQL databases [38] can be used for digital virtualization of the monitoring and manipulation object (sensors, actuators, nodes) and the surrounding context (employee profiles, operating conditions). Detection of composite events is possible to understand the nature of cause-and-effect chains and the simultaneity of a set of basic events for decision making (a knowledge base is created).

Fusing (linking) data in a graph model is the process of combining data sets obtained from heterogeneous sources to form a unique consistent view and to reduce the uncertainty of multi-source information. "Reducing uncertainty" means moving to a new level of abstraction, with a more reliable and accurate way to identify events occurring within TCPS [39].

The computation layer is core in the Bigdata architecture. Data sources are heterogeneous both in structure and content. As we considered above, the two computing modes are used: batch processing and stream processing. In batch processing, a large data block (batch) is received at the input processed in certain time period. In streaming data processing is not limited with beginning and end points. Data processing acts

 ISBN: 978-1-61208-989-8

in a sliding window or as individual records. It is necessary to process data on the fly, i.e., in near real time. Stream processing is easily scalable by creating new handlers on a given stream. Spark is a versatile large-scale batch and stream computing engine suitable for industrial applications [40]. Generally, the computation layer also handles data preparation, aggregation, fusing, and cleansing.

The analytics layer is designed to extract information and knowledge for decision making from the collected Bigdata. Two areas of analytics can be distinguished [39]: 1) online analytical processing (OLAP) using analytical queries, 2) data mining and machine learning algorithms (e.g., decision trees, convolutional artificial neural networks, regression, support vector machines). A toolkit is used depending on the requirements of the system. Spark SQL is used in analytics over structured batch data [40]. Spark Streaming is used for stream analytics [41]. MLlib and TensorFlow are used for machine and deep learning [42].

OLAP databases provide their set of tools in the form of a query language (usually SQL) for advanced analytics and business intelligence (e.g., Greenplum, Teradata). Analyzing streaming and retrospective data, analytical tools can recognize knowledge for decision making in TCPS (e.g., statistical metrics and spectral images for vibration diagnostics of rotary equipment, events about equipment deviations from normal operation modes, residual equipment life).

Data-driven extrapolation requires strain measurements to be made in all states of interest over a representative time period [43]. The collected strain data can be used to train the extrapolation algorithm. If there are no data for training, the missing data (beyond the range) are generated by simulation. In particular, the correlation between deformation and applied loads is non-linear under extreme stress conditions, so calculating and predicting deformation of equipment assemblies is difficult with traditional numerical methods. A back propagation neural network (improved by particle swarm optimization) can be used for determining the non-linear relationship between strain and load [44].

The service layer is on the top of our architecture. The layer is based on interactive multimedia human-machine interfaces. They provide users with a set of information and analytical services. The provision is a merged, seamless view, while hiding advanced analytical algorithms and differences in streaming and batch processing from the users. This view visualizes the results of the underlying layers using dashboards, reports, and plots. The service layer combines business and artificial intelligence with visualization to help in interacting the users and machines, in making decisions based on collected data, analytics, and expert evaluation [45].

On this layer, an interactive situational center is constructed to implement the "data–information–recommendation–evaluation–decision" cycle. Decisions are made based on the representation of the object's state. In particular, a decision is on timely equipment maintenance and the feedback is monitored. Recommendation services provide an evaluation of remaining life of machinery equipment components. Such decision making reduces maintenance costs and improves overall production reliability.

Reports are generated (according to plan and online) with information about the residual resource, recent and predicted state of the machinery equipment. Managing workflows for business, data processes, and resources requires orchestration tools that are used on all layers of the architecture, especially along the batch processing path [46]. A data workflow is a set of interrelated time steps that trigger specific jobs, such as Spark job or SQL query. Apache NiFi can be used as an orchestration tool for creating data streams and integrating data with the interactive service layer.

IV. Conclusion

This paper studied the use of TCPS to smart monitoring. We considered the two requirements of the system development: the bigdata requirement (R_{BD}) and the smart interaction requirement (R_{SI}). The role of the requirements was shown with respect to the tactile sense. A particular application area in demand is monitoring of various production machineries in real-time. We analyzed the properties from practical application problems and existing technologies for industrial data processing. We proposed the multi-layer TCPS architecture for effective processing of sensed data, either in batch mode or near real-time mode. Elements of the TCPS architecture have been already implemented in several monitoring applications. Our plan is to continue the development of smart monitoring IoT/IIoT systems based on the proposed generic architecture. We expect that the role of the tactile property becomes increasing in manufacturing, building construction, vehicle operation, robotics, and mobile healthcare.

Acknowledgment

The research is implemented with financial support by Russian Science Foundation, project no. 22-11-20040 (https://rscf.ru/en/project/22-11-20040/) jointly with Republic of Karelia and funding from Venture Investment Fund of Republic of Karelia (VIF RK).

References

[1] O. Petrina, S. Marchenkov, and D. Korzun, "A semantic space-time event representation model in production equipment monitoring of technical state and utilization condition," in 2022 8th International Conference on Control, Decision and Information Technologies (CoDIT), 2022, pp. 1362–1367.

[2] M. El Arass and N. Souissi, "Data lifecycle: From Big Data to SmartData," in 2018 IEEE 5th International Congress on Information Science and Technology (CiSt), 2018, pp. 80–87.

[3] H.-N. Dai, H. Wang, G. Xu, J. Wan, and M. Imran, "Big data analytics for manufacturing internet of things: opportunities, challenges and enabling technologies," Enterprise Information Systems, vol. 14, no. 9-10, 2020, pp. 1279–1303.

[4] D. Korzun, E. Balandina, A. Kashevnik, S. Balandin, and F. Viola, Ambient Intelligence Services in IoT Environments: Emerging Research and Opportunities. IGI Global, 2019.

[5] J. Zhang and D. Tao, "Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things," IEEE Internet of Things Journal, vol. 8, no. 10, 2021, pp. 7789–7817.

[6] T. Ali-Yahiya, "Introduction to Tactile Internet," in The Tactile Internet. Wiley-ISTE, 2021, pp. 1–11.

 ISBN: 978-1-61208-989-8

[7] V. Kostakos, J. Rogstadius, D. Ferreira, S. Hosio, and J. Goncalves, Human Sensors. Springer International Publishing, 2017, pp. 69–92.

[8] S. E. Navarro et al. "Proximity perception in human-centered robotics: A survey on sensing systems and applications," IEEE Transactions on Robotics, vol. 38, no. 3, 2022, pp. 1599–1620.

[9] S. J. Oks et al. "Cyber-physical systems in the context of industry 4.0: A review, categorization and outlook," Information Systems Frontiers, 2022, pp. 1–42.

[10] D. Korzun, S. Marchenkov, V. Ignakhin, and I. Sekirin, "Strain sensors in smart applications of tactile cyber-physical systems: Opportunities and recommendations," Jul. 2022, keynote Speech on The 2022 IARIA Annual Congress on Frontiers in Science, Technology, Services, and Applications (IARIA Congress 2022). Nice, Saint-Laurent-du-Var, France. [Online]. Available: https://www.iaria.org/conferences2022/ProgramIARIACongress22.html[accessed:2022-07-30]

[11] J.-H. Low, P.-S. Chee, E.-H. Lim, and V. Ganesan, "Design of a wireless smart insole using stretchable microfluidic sensor for gait monitoring," Smart Materials and Structures, vol. 29, no. 6, apr 2020, p. 065003.

[12] S. L. Ullo and G. R. Sinha, "Advances in smart environment monitoring systems using IoT and sensors," Sensors, vol. 20, 2020, pp. 1–18.

[13] F. L. M. dos Santos, B. Peeters, J. Lau, W. Desmet, and L. C. S. Goes, "The use of strain gauges in vibration-based damage detection," in Journal of Physics: Conference Series, Volume 628, 11th International Conference on Damage Assessment of Structures (DAMAS 2015) 24-26 August 2015, Ghent, Belgium, 2015, pp. 1–8.

[14] I. Sekirin and V. Ignakhin, "Sensors of mechanical stresses and deformations based on magnetic phenomena," in 2020 27th Conference of Open Innovations Association (FRUCT), 2020, pp. 207–213.

[15] Y. Lu, "Cyber physical system (cps)-based industry 4.0: A survey," Journal of Industrial Integration and Management, vol. 02, no. 03, 2017, p. 1750014.

[16] V. Perminov, V. Ermakov, and D. Korzun, "Fault diagnosis and prognosis for industrial rotary machinery based on edge computing and neural networking," in Proc. 14th Int'l Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM). IARIA XPS Press, Oct. 2020, pp. 1–6.

[17] A. Aijaz and M. Sooriyabandara, "The tactile internet for industries: A review," Proceedings of the IEEE, vol. 107, no. 2, 2019, pp. 414–435.

[18] N. Promwongsa et al. "A comprehensive survey of the tactile internet: State-of-the-art and research directions," IEEE Communications Surveys & Tutorials, vol. 23, no. 1, 2021, pp. 472–523.

[19] M. Maier, M. Chowdhury, B. P. Rimal, and D. P. Van, "The Tactile Internet: vision, recent progress, and open challenges," IEEE Communications Magazine, vol. 54, no. 5, 2016, pp. 138–145.

[20] O. Atalay, A. Atalay, J. Gafford, H. Wang, R. Wood, and C. Walsh, "A highly stretchable capacitive-based strain sensor based on metal deposition and laser rastering," Advanced Materials Technologies, vol. 2, no. 9, 2017, p. 1700081.

[21] A. Ladj, Z. Wang, O. Meski, F. Belkadi, M. Ritou, and C. Da Cunha, "A knowledge-based digital shadow for machining industry in a Digital Twin perspective," Journal of Manufacturing Systems, vol. 58, 2021, pp. 168–179, digital Twin towards Smart Manufacturing and Industry 4.0.

[22] W. Khan, M. Rehman, H. Zangoti, M. Afzal, N. Armi, and K. Salah, "Industrial internet of things: Recent advances, enabling technologies and open challenges," Computers & Electrical Engineering, vol. 81, 2020, p. 106522.

[23] F. Ballo, M. Gobbi, G. Mastinu, and G. Previati, "Advances in force and moments measurements by an innovative six-axis load cell," Experimental Mechanics, vol. 54, 2014, pp. 571–592.

[24] D. A. Kirienko, P. V. Lunkov, V. V. Putrolaynen, S. I. Aryashev, and M. A. Belyaev, "Modular hardware platform for the development of IoT devices implemented using multi-chip packaging technology," in IOP Conference Series: Materials Science and Engineering, Volume 862, Mechanical and Automation Engineering for Industry, 2020, p. 032012.

[25] Y. Cui, S. Kara, and K. C. Chan, "Manufacturing big data ecosystem: A systematic literature review," Robotics and Computer-Integrated Manufacturing, vol. 62, 2020, pp. 1–20.

[26] V. Saravanan, A. Alagan, and I. Woungang, "Big data in massive parallel processing: A multi-core processors perspective," in Handbook of Research on Big Data Storage and Visualization Techniques. IGI Global, 2018, pp. 276–302.

[27] T. M. Tukade, R. Banakar, and R. Banakar, "Data transfer protocols in iot—an overview," International Journal of Pure and Applied Mathematics, vol. 118, no. 16, 2018, pp. 121–138.

[28] M. Juma, A. A. Monem, and K. Shaalan, "Hybrid end-to-end VPN security approach for smart IoT objects," Journal of Network and Computer Applications, vol. 158, 2020, pp. 1–14.

[29] T. Dubuc, F. Stahl, and E. B. Roesch, "Mapping the big data landscape: Technologies, platforms and paradigms for real-time analytics of data streams," IEEE Access, vol. 9, 2021, pp. 15 351–15 374.

[30] F. Cerezo, C. E. Cuesta, J. C. Moreno-Herranz, and B. Vela, "Deconstructing the lambda architecture: An experience report," in 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), 2019, pp. 196–201.

[31] S. Srinivas and V. R. Karna, "A survey on various message brokers for real-time big data," in Sustainable Communication Networks and Application, P. Karrupusamy, J. Chen, and Y. Shi, Eds. Springer International Publishing, 2020, pp. 164–172.

[32] D. G. Korzun, S. I. Balandin, A. M. Kashevnik, A. V. Smirnov, and A. V. Gurtov, "Smart spaces-based application development: M3 architecture, design principles, use cases, and evaluation," International Journal of Embedded and Real-Time Communication Systems (IJERTCS), vol. 8, no. 2, 2017, pp. 66–100.

[33] N. G. Ugur and A. H. Turan, "Understanding big data," in Research Anthology on Big Data Analytics, Architectures, and Applications, I. R. M. Association, Ed. IGI Global, 2022, pp. 1–21.

[34] T. R. Rao, P. Mitra, R. Bhatt, and A. Goswami, "The big data system, components, tools, and technologies: a survey," Knowledge and Information Systems, vol. 60, 2019, pp. 1165–1245.

[35] M. Hajeer and D. Dasgupta, "Handling big data using a data-aware hdfs and evolutionary clustering technique," IEEE Transactions on Big Data, vol. 5, no. 2, 2019, pp. 134–147.

[36] P. Raj, "Chapter one—a detailed analysis of NoSQL and NewSQL databases for bigdata analytics and distributed computing," in A Deep Dive into NoSQL Databases: The Use Cases and Applications, ser. Advances in Computers, P. Raj and G. C. Deka, Eds. Elsevier, 2018, vol. 109, pp. 1–48.

[37] K. T. Sridhar, "Modern column stores for big data processing," in Big Data Analytics, ser. Lecture Notes in Computer Science, P. K. Reddy, A. Sureka, S. Chakravarthy, and S. Bhalla, Eds., vol. 10721. Springer International Publishing, 2017, pp. 113–125.

[38] I. Comyn-Wattiau and J. Akoka, "Model driven reverse engineering of NoSQL property graph databases: The case of Neo4j," in 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 453–458.

[39] R. H. Hariri, E. M. Fredericks, and K. M. Bowers, "Uncertainty in big data analytics: survey, opportunities, and challenges," Journal of Big Data, vol. 6, no. 44, 2019, pp. 1–16.

[40] M. Zaharia et al. "Apache spark: A unified engine for big data processing," Communications of the ACM, vol. 59, no. 11, oct 2016, pp. 56–65.

[41] B. Zhou et al. "Online internet traffic monitoring system using spark streaming," Big Data Mining and Analytics, vol. 1, no. 1, 2018, pp. 47–56.

[42] M. Assefi, E. Behravesh, G. Liu, and A. P. Tafti, "Big data machine learning using apache spark mllib," in 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 3492–3498.

[43] L. Ziegler, N. Cosack, A. Kolios, and M. Muskulus, "Structural monitoring for lifetime extension of offshore wind monopiles: Verification of strain-based load extrapolation algorithm," Marine Structures, vol. 66, 2019, pp. 154–163.

[44] X. Liu, Z. Liu, Z. Liang, S.-P. Zhu, J. A. F. O. Correia, and A. M. P. D. Jesus, "PSO-BP neural network-based strain prediction of wind turbine blades," Materials (Basel), vol. 12, no. 12, 2019, pp. 1–15.

[45] Y. Lu, J. S. Adrados, S. S. Chand, and L. Wang, "Humans are not machines—anthropocentric human-machine symbiosis for ultra-flexible smart manufacturing," Engineering, vol. 7, no. 6, 2021, pp. 734–737.

[46] Y. Cui, S. Kara, and K. C. Chan, "Monitoring and control of unstructured manufacturing big data," in 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2020, pp. 928–932.

 ISBN: 978-1-61208-989-8