A Review of the Principles of Designing Smart Cyber-Physical Systems for Run-Time Adaptation

Learned Lessons and Open Issues

Tavčar, Jože ; Horvath, Imre

DOI
10.1109/TSMC.2018.2814539

Publication date
2019

Document Version
Accepted author manuscript

Published in
IEEE Transactions on Systems, Man, and Cybernetics: Systems

Citation (APA)
Tavčar, J., & Horvath, I. (2019). A Review of the Principles of Designing Smart Cyber-Physical Systems for Run-Time Adaptation: Learned Lessons and Open Issues. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(1), 145-158. Article 8329014. https://doi.org/10.1109/TSMC.2018.2814539

A Review of the Principles of Designing Smart Cyber-Physical Systems for Run-Time Adaptation: Learned Lessons and Open Issues

Jože Tavčar and Imre Horváth

***Abstract*—Smart cyber-physical systems (S-CPSs) are complex engineered systems empowered by cyber-physical computing and equipped with the capability of reasoning, learning, adapting, and evolving. As an outcome of data-driven dynamic computing, reasoning capabilities, and the run-time obtained own knowledge, nonlinear and emergent behavior of S-CPSs whilst in operation is an open issue, not experienced in the case of conventional technical systems. This paper analyzes the technical issues of run-time operation and emergent behavior of S-CPSs, reviews the current understanding and state of advancement in designing S-CPSs for run-time, explores the paradox, and issues of designing for run-time adaptation, and synthesizes some general principles that can be taken into consideration when addressing the challenges, first of all, in the context of advanced manufacturing systems. This paper introduces four levels of CPSs according to reasoning capabilities and adaptation freedom of systems, and recognizes the paradox that a system with a higher level of freedom requires a higher level of self-control and resource management according to the overall objective of operation. Specific and common design principles are presented and critically assessed for each advancement level of CPSs. The principles synthesized by the authors provide only a partial fulfillment of the generic need. The planned future research addresses these issues and proposes (largely implementation and application independent) genuine principles for system developers.**

***Index Terms*—Design principles, industry 4.0, run-time adaptation, self-adaptation, self-awareness, smart cyber-physical systems (S-CPSs).**

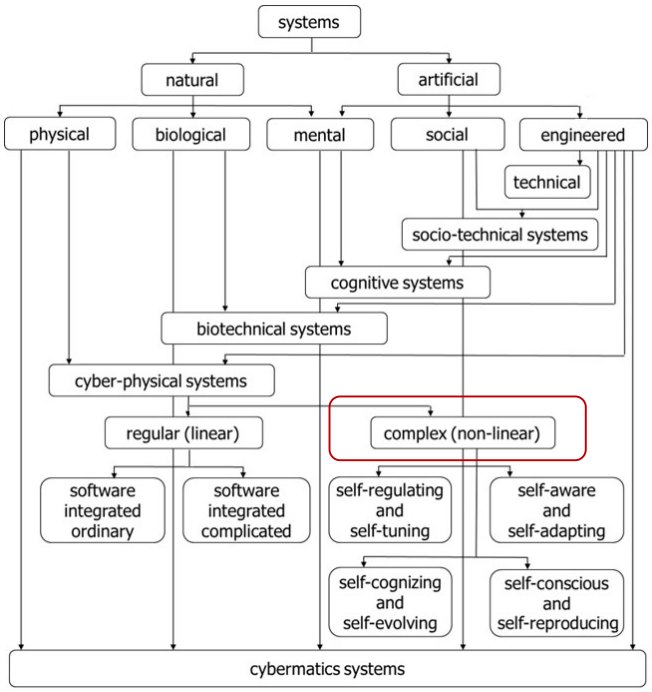


Fig. 1. Placement of S-CPSs on the landscape of systems. They are complex and nonlinear, and have specific intelligence and organizational features.

I. Introduction

A. Background of This Paper

THIS paper deals with smart cyber-physical systems (S-CPSs), which can be sorted into the category of complex nonlinear systems. They have a unique set of paradigmatic features such as self-awareness, self-adaptation, self-evolution, and self-reproduction (Fig. 1). They implement a higher level of integration of hardware, software, and cyberware technologies than any other type of systems ever before [1]. S-CPSs are typically deeply embedded-in, and having numerous internal/external relationships with, natural or engineered environments, and are operational status and context-aware systems which cognitively interact with humans and influence their social life. They are complex, partially autonomous systems, showing emergence, dynamics, nonlinearity, and other behaviors not present in the elements. Though S-CPSs are open systems from both architectural and operational viewpoints, they implement a high-level of synergy with regard to all dynamically changing functionalities and components. Their control is partially model-based and partially driven by run-time acquired data. The on-line feedback provided by S-CPSs strongly depends on the purpose of their application (informing, actuating, or both). Though many S-CPSs are distributed and decentralized networked systems, their operation is supposed to be real-time, and often show elements of autonomy [2].

Manuscript received October 31, 2017; revised February 15, 2018 and February 24, 2018; accepted March 1, 2018. This work was supported by the Slovenian Research Agency under Contract P2-0265. This paper was recommended by Associate Editor A. Trappey. *(Corresponding author: Jože Tavčar.)*

J. Tavčar is with the Faculty of Mechanical Engineering, University of Ljubljana, Ljubljana SI-1000, Slovenia (e-mail: joze.tavcar@lecad.fs.uni-lj.si).

I. Horváth is with the Department of Design Engineering, Cyber-Physical Systems Design, Technical University of Delft, Delft, The Netherlands (e-mail: i.horvath@tudelft.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2018.2814539

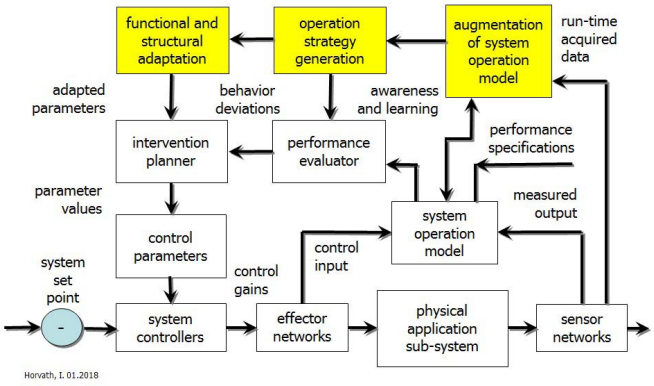


Fig. 2. Control regime of S-CPSs. It combines the loops of feedback control with those of data-driven reasoning, operational strategy generation, and structural adaptation.

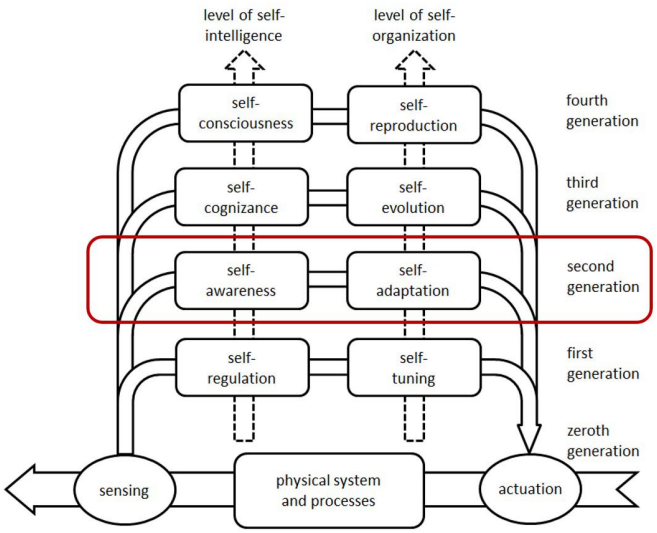


Fig. 3. Different generations of CPSs according to level of self-intelligence and self-organization [4] (2G-CPSs are referred to as smart CPSs).

The primary enabling asset of CPSs is cyber-physical computing (CPC). It pursues dynamic computing based on run-time obtained information [3]. From a computational perspective, it overwrites the von Neumann theory-based computing (i.e., making calculations in a predefined way). Owing to this, CPC makes both nonlinear operation and emergent behavior of hardware, software, and cyberware constituents possible in run-time. S-CPSs implement two recurrent and intertwined cycles of computing.

1) The basic control cycle, which comprises "sensing → monitoring → adjusting → actuating" activities
2) The (self-)enhancement cycle, which comprises "reasoning → learning → adapting → evolving" activities (Fig. 2).

Consequently, multiactor S-CPSs are not fully deterministic systems. They have some level of freedom in setting the objectives of system operations, as well as the capability of adapting themselves to varying tasks, situations, and contexts. This paper concentrates on what is known as the second generation of CPSs, which are able to implement self-awareness and self-adaptation whilst in operation (Fig. 3). Underpinning theories, computational methodologies, and enabling technologies for the fourth generation of CPSs are still in a premature stage nowadays. However, it is important to understand the trend of development and touches upon the essential characteristics and issues related to the third generation of CPSs. The classification of CPSs into four groups according to the adaptability capability level is an important paper contribution. This help to create the whole picture of CPSs run-time adaptation and in the next phase improve understanding of design principles for S-CPS.

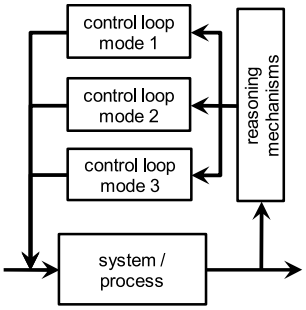
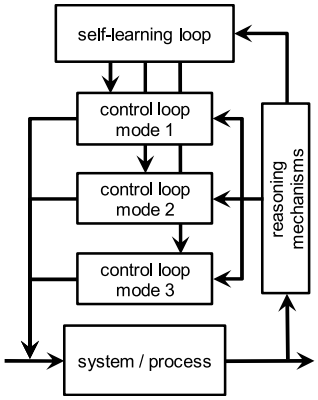
1) Level 1: First generation of CPSs with no changes in life span. The system structure and way of default operation is defined in the design phase, and it does not change throughout the system life span. The system has conventional control mechanisms and can regulate parameters to a known degree. In the case of a fault or changed circumstances in the environment, human intervention is expected. An example is a CNC machine where cutting speed and feed rates are in the first control loop. The production rate and machining cost can be additionally improved on the base real-time measurements of process variables, and the setting of optimal parameters during machining. The end-user can adjust the predefined adaptive control algorithms with some preselected parameters.

2) Level 2: Second generation of CPSs with known modes of changes. The system is designed for alternative modes of control and selection of the optimal mode of control during run-time. Resources, modes of control, and a reasoning algorithm for selecting different modes are predefined in the design phase and do not change during the system life span (Table I). Data from system and environment awareness is used for control mechanisms in each mode of operation. An example is an air-fly control system that operates under several modes. There is at least a simple validated mode and the advanced mode with better performance [5]. The control system can select the optimal mode of operation in real-time according to measured data and reasoning logic. Some modes of operation are prepared for specific faults, such as if one aircraft engine does not work.

3) Level 3: Third generation of CPSs with quasi-known/unknown changes. Self-learning CPS has the ability to adapt predefined control algorithms during the exploitation period. The adaptation can be conducted only inside predefined limits or an envelope, and with constraint resources that are because of safety or any other reasons defined in the design phase (Table I). An example is a self-learning robot. The limits of the basic technical characteristics and resources are defined in the design phase: maximal arm acceleration and speed, weight of grippers, way of moving, and working area. Inside those predefined limits the robot is allowed to adapt its control algorithms for optimal control of movement [6]. The robot records the operating parameters and compares them with target function. On the basis of reinforcement-learning in the second loop, the robot can adapt its control algorithm throughout its lifetime.

4) Level 4: Fourth generation of CPSs with largely unknown changes. The system has the ability of control generation and in this way of constraining the application

TABLE I
LEVEL OF FREEDOM AND CONTROL MECHANISMS OF SECOND GENERATION OF CPSs

Level 2: Known modes of changes	**Level 3:** Quasi known/unknown modes of changes
 Selection of preprogrammed alternative control mechanisms for run time.	 Adaptation of predefined control according to learned conditions.
Level of freedom and self-control mechanisms	
Selection of alternatives with regards to known degrees. Regulation inside of each mode. System control keeps the system within preset limits. Safety loop.	Self-learning of how to adapt. Control of adaptation in each mode. System control software configuration within preset limits. Negotiation on the possible trajectory of evolution run-time. Run-time validation .
Role of humans	
Predefinition of all resources and operational alternatives.	Direct adjustment of resources and operational limits. Interventions in the case of faults.

and system structure. However, the field of adaptation is not restricted to rigid limits. Typically, human is involved in the control loop as supervisory controller. The manner in which the system adapts and limits itself depends on the applied self-learning algorithms. In the design phase it is not known in which way the system will adapt and what will be the final result. Two practical examples may be useful to clarify this. The first example is an S-CPS resembling implementation of a CNC machine and its work process. By accumulating data about the experiences with material removal by the different cutting tools and the achieved workpiece quality over the entire life cycle work of the CNC machine, deficiencies can be identified and adaptations can be introduced. This knowledge can be a rich source of further development of the cutting technology and optimal design of tool materials and morphologies. In addition, the various parameters of specific machining processes may be optimized, predicted, and adjusted dynamically [7]. The second example is a swarm of robots that are smartly coordinated in order to achieve an optimal task performance in a future production factory. In this context, de Lope *et al.* [8] proposed a distributed approach with autonomous techniques, where the robots or agents themselves are responsible for choosing a task. In an experimental scenario, a reinforcement learning algorithm was used that was based on ant colony optimization.

The CPSs of Level 1, and partially those of Level 2, operate exactly as their operation was specified in the design phase. The systems of Levels 1 and 2 can be context-aware in a growing extent, and they can tune or adapt themselves to new circumstances in the same way throughout the whole life cycle. Currently, for a safety-critical system like air-fly control, it is requested that they operate in the deterministic way they were validated. According to the proposed classification, only CPSs of Level 2 and above are treated as smart. CPSs belonging to level higher than 2 have the capability of self-evolving, change their behavior and performance throughout their life cycle toward some novel (but supervised) objectives, and have capability to adapt to new circumstances that were not known in the design phase. However, S-CPSs also need a self-control mechanism. As mentioned earlier, in this paper, we only address issues that are associated with the second generation CPSs since this generation of systems are becoming a daily reality in manufacturing environments already in our days.

Considering the above examples, the following principle can be extracted. If the system has a higher level of adaptation freedom, then it requires a higher level of self-control to the resource exploitation and setting the operation modes. A derived conclusion is that complex and self-adaptable systems need to be autonomous, system- and environment-aware, and self-controlled. The summary of control mechanisms is given in Table I. The design principles for designing the second generation of S-CPS need to be defined. In the next chapter a concise literature review of designing for run-time adaptation is given. Currently, there is information about a limited number of adaptive CPSs (reaching to or beyond Level 2), which are in daily operation. This explains why only, CPSs with known modes of changes were included in our research.

At the same time, our analysis covered some representative examples of Level 1 for the sake of comparison with designing a conventional technical system. The recognized design principles used at Level 2 were critically assessed if they can be applied for designing S-CPS with quasi-known changes through the life span (Level 3). The rest of this paper is organized as follows. Section II will provide a concise overview of the literature of designing for run-time adaptation. Section III will summarize technical and open issues of run-time adaptation. Finally, Section IV discusses emergent behavior of S-CPSs, practical implications of the design principles, and Section V presents the final conclusions.

B. Smart CPSs and Industry 4.0

In the manufacturing environment, businesses will establish global networks that incorporate production facilities, machinery, and warehousing systems in the shape of cyber-physical systems (CPSs) [9]. Self-organizing manufacturing, context-/situation-aware control, and symbiotic human–robot collaboration will have paramount importance. The unique features of CPS in networking, communication, and integrated device control point to the smartness and intelligence of manufacturing in the horizon. The CPS technology must find transitional technologies through which the truly novel ideas may be gradually introduced on the shop-floor level,

without incurring major investments [10]. There are several initiatives that accommodate CPS development, including the "Advanced Manufacturing Partnership 2.0 (AMP, 2014)" and "Industrial Internet (IIC, 2014)" in the USA, "Industry 4.0" [9] in Germany, "Factories of the Future" in the EU, the "Made in China 2025" strategy alongside "Internet Plus" in China, and several others [11]. Industry 4.0 is focused on creating smart products, procedures, and processes [12], [13]. By connecting people, things, and data, new ways of organizing and conducting industrial processes emerge. Smart factories are able to manufacture goods more efficiently, are less prone to disruption, and capable of managing complexity. Within a smart factory, self-organizing value chains can be optimized in real-time. This will require an appropriate regulatory framework, as well as standardized interfaces and harmonized business processes [9]. The building blocks for the development of S-CPSs originate in several disciplines, such as: software engineering, control engineering, IT, AI, embedded systems, Internet of Things (IoT), advanced robotics, and sensor and actuator technology. The development toward a network of smart autonomous objects is already happening in the case of IoT [14]–[16].

Wang *et al.* [17] are focused on a vertical integration that enables a highly flexible and reconfigurable smart factory. The physical artifacts form a self-organized and autonomous manufacturing system based on big data, an industrial network, and an intelligent negotiation mechanism. The smart factory is a specific implementation of CPS [17]. Nowadays, decisions on process adaptions are, in most cases, made by humans. In the future the decision process will be supported by knowledgeable and self-optimizing manufacturing systems [18], [19]. In the context of intelligent manufacturing there is a promising predictive diagnosis based on industrial big data [20], [21]. Nowadays 95.1% of publications related to Industry 4.0 present a kind of laboratory experiment, while only 4.9% of them present an industrial application [11]. However, the number of contributions is steadily rising [10], [13], [22] such as smart welding [23], self-organizing techniques for multirobot system [8], or intelligent diagnosis on the basis of big data mining [24].

The first publications concerning the paradigm and realization of Industry 4.0 appeared in 2011. Two years later Lu found 11 related articles, and in 2014 there were already 29 [13]. This is also in connection with a gradual increase in the number of conferences related to Industry 4.0 from 2013 (5 conferences) to 2015 (63 conferences) [11]. Cyber-physical production systems partly break with the traditional pyramid, because a more decentralized way of functioning is characteristic for higher levels of the hierarchy [25]. One of the distinguishing features of S-CPSs is their smart reasoning capability. CPSs equipped with this capability are believed and expected to offer new opportunities for business innovation in the service-orientated manufacturing industry [26].

II. Reviewing the Literature on Designing for Run-Time Adaptation

Our ambition was to conduct a systematic exploration of design principles that could be used in designing S-CPS for run-time adaptation. It is a specific focus that does not cover all aspects of intelligence and organization of CPSs. A systematic literature research was done concerning journals, conference papers, books, and other Internet sources of information. The key words applied at searching in Web of Science, Science Direct, Scopus, and Google Scholar were: CPS, smart CPS, self-control, run-time control, context-aware, self-awareness, self-evolving, self-healing, self-organization, self-adaptive, self-learning, Industry 4.0, robust, resilient, complex system, fault tolerance, knowledge management, system modeling, and simulation. Additionally, some of the references found in selected publications have also been checked (also for avoiding cross-referencing). Altogether, more than 500 publications have been analyzed, from which 259 were journal papers. A detailed overview of the statistical composition is presented in Table II. The analysis was conducted manually and the relevance and adequacy of the contents of the publication for S-CPS was the main selection criterion.

TABLE II
Structure of Analyzed Publications

Type of publications	Number of analyzed publications		Included in the review	
	Total	IEEE	total	IEEE
Journal papers	259	49	68	15
Conference papers	223	91	30	11
Book chapters, reports	56	n.a.	27	n.a.
Dissertations	9	n.a.	1	n.a.

One of the methodological conclusions of the research was that only a very limited number of publications reported on really smart CPS according to our strict definition. The majority of publications were related to first generation CPSs, which are not characterized by any form of self-evolving capability. With some exceptions, the publications dealing with issues of this generation of CPSs have not been included in the review. A large number of the analyzed publications were related to pure software systems. Nonstop operation, fault-tolerance, artificial intelligence, and run-time control of software has been a research focus for two decades already. However, software is a key component of any CPS, therefore some of design principles from the software solutions can also be the inspiration for the S-CPSs. There were several obstacles reported that hinder the proliferation of second and third generation S-CPSs in safety critical applications. For instance, there are complexity- and dependability-implied safety issues and the validation protocols were laid down in the time of deterministic technical systems.

Eventually, the search results have proved that there was a place and need for reviewing the knowledge and advancement concerning the principles of designing S-CPSs. In particular, the design challenges and principles related to anticipating human design and run-time self-design of second generation CPSs did not receive sufficient attention yet. In overall, there were many more issues discussed concerning CPSs with known modes of changes, that concerning CPSs with quasi-known changes. This explains why the intention of the authors was to contribute to the awareness of design

principles of second generation CPSs. The demarcating categorization and systematic analysis applied in this paper serve two purposes: 1) showing the whole picture and the local details related to CPSs and 2) enabling a better understanding of the requirements, possibilities, and limitations of S-CPSs. This paper also shows the way toward the third generation of CPSs in several details.

The problems of designing a complex system for run-time adaptation are broadly addressed in the contemporary literature. Adaptive systems represent a special class of nonlinear systems that intellectualize, organize and measure their own performance, operating environment, and the operating condition of their components [27]. Ding *et al.* [28] introduced an adaptive Petri net model for a self-adaptive software system. Brun *et al.* [29] proposed a hierarchy of self-* properties. They differentiated: a primitive level (which includes self-awareness and context awareness), a major level (which includes self-configuring, self-healing, self-optimizing, and self-protecting), and a general level (which includes self-adaptiveness). In their road-mapping paper, Cheng *et al.* [30] summarized the state-of-the-art and identified critical challenges for systematic software engineering of self-adaptive systems and addressed supporting factors of self-adaptation.

The literature advises us that self-adaptation may be a useful capability of complex systems to achieve the objective and the operational or behavioral requirements. As Weyns *et al.* [31] recognized, there are different communities behind these notional descriptions, as well as different vocabularies. Healthcare CPSs have already found their way to advanced applications [32]. However, there is no clear view on how self-adaptation actually contributes to tackling the challenges of engineering and managing complex software systems. Current literature claims that self-adaptive CPSs should be capable of adjusting or changing their structure, functionality, and behavior during run-time as a response to emerging requirements, changing objectives, environments, and contexts that may be unknown at design-time. Not only the physical subsystem does need to be adaptive (responding to changing conditions), but also the software and the cyber subsystems, which should in addition be even predictive (anticipating changes in the physical processes).

A. System Architecture From Control Point of View

System regulation and adaptation is enabled with several levels of control loop, as already presented in Fig. 2. Software architecture has dominant influence on the way of implementation and the whole system structure. Therefore, several important references come from the software domain. IBM defined an "Autonomic Computing" program [33]. The general concept was in the following decade developed to several architectures that are applicable for S-CPSs. A self-managed software architecture is one in which components automatically configure their interaction and achieves the goals of the system. A three-layer reference model with goal management, change management, and component control was defined [34].

There are several activities related to Industry 4.0 on the level of vision, architecture and reference models, learning factories, and the specific solution in laboratory environment. In 2011, the first self-organizing assembly system was demonstrated at FESTO in Germany, this approach was based on Agent-oriented Architectures [10]. Wang *et al.* [17] proposed an intelligent negotiation mechanism for agents to cooperate with each other, and to enable distributed self-decision making with big data-based feedback assistance. Civerchia *et al.* [35] reported on the capabilities and performance of an IoT system in real industrial environments. The learning factory that integrates the shop floor and top floor via suitable cloud services supports holistic, problem-based learning, and evaluation of research projects [36]. Wang *et al.* argued that the smart factory framework consists of four layers: 1) the physical resource; 2) industrial network; 3) cloud; and 4) supervisory control layers. Big data is collected in the cloud from the smart things in the physical layer and interacts with people through supervisory control [17]. Hermann *et al.* [37] identified some generic design principles for Industry 4.0: interconnection, decentralized decisions, information transparency, and technical assistance.

In a second step, scenarios and a specification book for implementation of the smart factory was prepared [37]. An ambitious effort of the swarm approaches is the UC Berkeley led TerraSwarm project. The Smart Factory initiative has realized a multivendor and highly modular production system as reference for Industry 4.0 [38]. Trappey *et al.* [39], [40] provided a consolidate review of CPS literature and a review of international standards and patents related to CPS and IoT. Lee *et al.* [41] presented the 5C architecture for implementation of advanced CPSs as representatives of manufacturing systems in the context of Industry 4.0. This 5C architecture starts with involving smart sensors on level I, while level II brings in self-awareness to machines, level III (cyber level) acts as the central information hub, and while there is a decision support system on level IV. Level V (the configuration level) feedback is provided from cyber space to the physical space. This layer provides supervisory control to make machines self-configurable and self-adaptive [41]. The presented 5C architecture can also be applied in the case of S-CPSs, if the top level is supported by the capability of self-evolution and self-control. The first generation of CPSs presented in this paper extends at least to the first three levels of the 5C architecture.

The monitor–analyze–plan–execute over a shared knowledge (MAPE-K) feedback loop is the most influential reference control model for autonomic and self-adaptive systems [42]. A verification technique enables discovering unwanted interferences between MAPE-K loops in the early stages of system design [42]. A dynamic software product line (DSPL) extends the MAPE model by learning and adaptation [43]. Intraloop coordination allows the execution of multiple subloops within one control loop, and allows the MAPE computations of different loops to coordinate the various phases of adaptations [44]. The DEECo system applies dynamic architecture abstractions of autonomous components and component ensembles that

provide a straightforward reflection of operational goals in the application architecture [45]. Capodieci *et al.* [46] proposed a conceptual framework for the design of systems that is based on inspiration from the natural immune system.

The SEAMS community has recognized the control objectives manager, the adaptation controller, and the context monitoring system as the key subsystems for the design of effective context-driven self-adaptation [31], [47].

A step forward at specifying a self-adaptive system is done with patterns and templates. Ramirez and Cheng [48] documented a set of 12 patterns that focus on the software design level, and they aim to facilitate the design and construction of self-adaptive systems. Weyns *et al.* [49] presented a set of five patterns for decentralized control in self-adaptive systems that were derived from different studies. The templates provided reusable design knowledge that in turn allows rigorous modeling and verification of the behaviors of MAPE-based feedback loops for distributed applications in which self-adaptation is used for managing resources [50]. The system architecture needs an additional control loop for each additional level of freedom. On Level 2, reasoning mechanisms for mode selection is needed, and on Level 3 a self-learning loop is necessary [44], [51]. The key concerns are knowledge needed for implementation and integration of several self-control-loops.

B. Run-Time Control and Middleware

In the run-time self-control, a key capability of autonomous systems and the related theories and methodologies focus on issues such as robustness, fault-tolerance, etc. Most of the solutions for adaptive CPSs come from software engineering. Run-time monitoring is supported and encouraged as a fundamental principle for building reliable systems [52], [53]. The four meta-models (DSPL, context, reasoning, and architecture) mean the main data manipulated by the run-time infrastructure responsible for dynamically adapting component-based applications at run-time [54]. A conceptual reference model for M@RT contains three levels of adaptation mechanisms: 1) level M1 includes the run-time models; 2) M2 level determines the syntax and semantics of these models; and 3) the top level M3 is the meta-metamodeling level [55].

It is still an open question how to take advantage of S-CPS and fulfill safety criteria. One solution is based on the concept of fault isolation and recovery at the service level [56]–[58]. The Simplex architecture, which supports using simplicity to control complexity, is a common approach for keeping a complex system reliable. That means the existence of a simple and reliable core component that ensures the system's critical functions despite the failure of non-core software components [59]. A fault-tolerant control system is designed to mitigate the effects of system component failures [60]. The model-based fault detection, isolation, and reconfiguration method is divided into the fault detection and isolation step, and the controller reconfiguration step [61]. The "Golden Rules" to design a fault-tolerant aircraft include: hardware redundancy, monitoring of flight control elements in real-time, reconfiguration in case of failure, dissimilarity, installation segregation, and robustness of software and system equipment [62], [63].

The protection scheme, called "Run-time Enhancement of Trusted Computing (RETC)," enhances trust in CPSs which contain untrusted software and hardware. RETC is complementary to design-time verification approaches. Interface guards enable in-line monitoring and enforcement of critical system computations at run-time [64].

Real-time properties of CPSs are the key characteristic in safety critical situations [65]. Hardware architectures can significantly contribute to the reliability of systems. Middleware platforms provide novel approaches for faster and more reliable development and operation of complex CPS applications, where distributed and heterogeneous components interact in various ways [66].

Timely operation is a key issue in CPS because of the inherent distributed nature. In CPSs some degree of control over the resources is expected to assure timeliness. The proposed solution is a middleware model, typically referred to by the acronym OMA-Cy, which serves as a general reference architecture to design and implement middleware technology for CPS domains. OMA-Cy is short name for Overarching Middleware Architecture design model for CPSs [67]. Cucinotta *et al.* [68] presented service-oriented architectures that support real-time and quality-of-service (QoS) capabilities for industrial automation. They define hard, real-time activity that has to be completed in a specified time frame; like sense-compute-actuate control loops need to be in a range below 10 ms. [68].

The strong real-time constraints of many CPSs introduce new challenges. Run-time monitoring and self-healing has been an open topic for decades in software engineering [69]. In the context of CPSs, requests are stricter because the physical system cannot be simply restarted, and unstable critical operation can cause irreparable damages [70]. Faults and unpredictable events in the environment are part of each CPS operation. In the case of the second generation of CPSs, smart run-time monitoring is used to assure fault-tolerant operation [61]. An important difference is that, at first, generation of CPS central control dominates, because complex systems of second generation distributed control has advantages [71].

C. Reasoning Mechanisms (Self-Learning, Context-Awareness)

The capability for reasoning, adapting, and self-learning are the main characteristics of the second generation of CPS. Reasoning is the process of drawing conclusions by utilizing human beings' problem-solving strategies. It is reasoning with facts and knowledge with given steps, using sets of inferences, or reasoning strategies. Reasoning strategies for S-CPSs extract and combine data, information, and collect and assess raw data acquired from a dynamic, noisy, and uncertain physical environment, and may convert them into useful cyberware (useable knowledge) in real-time.

Four main functions have been identified for the implementation of a control loop in IBM's autonomic element

management scheme [72]. Håkansson *et al.* [73] presented reasoning strategies for S-CPSs that can extract and combine data, information, and knowledge. DIANNE is an incremental learning system that processes inputs in real-time, and the temporal learning algorithm executes continuously in a life-long manner [74]. The implementation of model-based reasoning approach can be easily reused [75]. Reinforcement learning is a computational approach to understanding and automating goal-directed learning and decision-making. It contains four main subelements: 1) a policy; 2) a reward function; 3) a value function; and 4) optionally, a model of the environment [6], [76].

D. Context-Awareness

System and environment awareness is the input information for the reasoning and learning processes. There are three different approaches on how to acquire contextual information: 1) direct sensor access; 2) middleware infrastructure for eases extensibility; and 3) context server permits multiple clients access to remote data sources [77]. Cognitive context information is the key information to provide context-aware computing services. The physical context information represents the low-level context, and the high-level context is inferred from the low-level one [78]. Situation-awareness agents are able to assess their own abilities and behave proactively by learning from each other [79]. Context-awareness has already become a service and context management has become an essential functionality in software systems [80]. The awareness process is cyclic with many iterations and different classes of awareness [81].

E. Building Blocks Level (Complexity and Self-Awareness)

The investigations of system components extend to analog and digital hardware, system and application software, and active knowledge and media cyberware. An autonomic element consists of a closed control loop that can control a system without external intervention. A quantitative measure of the degree of autonomy of technical application systems is presented—static and dynamic degree of autonomy is introduced [82]. With increasing complexity and heterogeneity of the systems-on-chip (SoCs) platform architectures, there is an extra need for self-awareness of these SoCs and for performing in an adaptive manner. Self-aware SoCs require a combination of ubiquitous sensing and actuation, health-monitoring, and statistical model-building to enable the adaptation of SoCs over time and space [83]. Subagdja and Tan [84] presented a framework for developing an application based on smart autonomous components that collaborate with the developer or user to realize the entire system. Filho *et al.* [12] presented importance and criteria for a self-aware smart product in the context of smart 4.0 production environment.

Ye *et al.* [85] surveyed the field of self-organizing multiagent systems. In many cases, the need for scalability of complex systems arises from the fact that most of the relevant processes are performed locally in self-organized and adaptive way [71].

One key to achieving dependability at a reasonable cost is commitment to simplicity of critical functions and simplicity in system interactions. Designing for simplicity principle means that artifacts can be analyzed by simple models at different levels of abstraction [86]. Alexopoulos *et al.* [87] presented a context-aware manufacturing information system that is based on several advanced technologies, including: near field communication, radio-frequency identification, Web services, and related standards.

Context-awareness is one of fundamental necessities to drive configuration and adaptation, that is, appropriate for actual conditions [88]. The building blocks of the system have to be autonomous and context-aware [14]. They can have complex internal structure, but from the point of view of the system designer they need to have representation at high abstract level. That means system designer do not need to know all technical details to use them; the system can be reconfigured in a simple way, new components can be added, old ones can be replaced. The basic standards for interconnection and communication need to be common accepted and reliable through longer time frame [83], [86]. Modular product structure and use of standard, replaceable components is preferred already in conventional technical systems. At the building block level there is no sharp boundary between requests for the second or third generations of CPS. But with increasing system complexity and unpredictability of behavior, it is a must to have autonomous and context-aware building blocks.

F. Modeling and Simulations

A top-down system design starts with a conceptualization of the intended high-level behavior of the planned system. This system model is refined at multiple levels until a representation that can be executed on the selected hardware platform has been generated [86]. The classic reductionist tools are no longer adequate. Complex, nonlinear systems cannot be modeled by linking together a fragmented collection of linear models. A new, multidisciplinary toolkit is needed to model complex, adaptive, and unpredictable systems [2]. CPS constitutes a new engineering discipline that demands its own models and methods. CPSs combine deterministic models in such a way that determinism is not preserved. Lee [89] argued that deterministic CPS models with faithful physical realizations are possible and practical.

Gürdür *et al.* [90] introduced a specific visualization approach to make interoperability of tool chains visible for a better understanding of needs in CPS development. There is no guarantee that an optimal design in terms of mechanics will also exhibit a good closed-loop performance. In order to achieve the mechatronic optimum, the closed-loop nature of the system has to be considered from the start of the design [91]. The implementation of second generation of CPSs requires various resources that are supposed to enable a holistic system operation together. The conventional separation of computation (software) from physicality (hardware) does not work for CPSs. Instead, a more holistic approach that integrates the mentioned parts into a composition is needed. The physical (analogue) part of a generic CPS is complemented by

five other computing resources: 1) netware; 2) (digital) hardware; 3) software; 4) firmware; and 5) knowledgeware. The composition platform synchronizes the various subplatforms and facilitates an integral consideration [1].

The concept of system manifestation features has been developed to support modeling and simulation of complex, heterogeneous, and adaptive CPSs [92]. A reference model for self-adaptive systems where adaptation goals and monitoring requirements change dynamically, DYNAMICO enables faster software development [47]. The method presented by Canedo *et al.* [93] automatically generates industrial CPS simulation models from functional models. In model-based development, there is one system model that is the central artefact of the whole design process. The system model is built at the earliest possible time, and transformed and upgraded such that parts of the model can be executed. A virtual prototype can be derived from this initial system model, which is used to simulate and validate the behavior of the target system, even before it physically exists [94].

Modeling and simulation is a helpful design tool for all three levels of CPS. However, requests for modeling are growing with complexity and in nonlinear system behavior. An integrated model with the possibility for simulation is a must for a complex system with several interdependent components. The system developer needs to get feedback information on system behavior as soon as possible. An open issue is how to model self-adapting systems? In the design phase it is not known the system final structure and how reasoning mechanisms can be adapted during system life span. Self-adapting systems need reliable and advanced self-awareness and a self-control mechanism. Robustness and functionality of control mechanisms has to be precisely modeled and systematically tested. The summary of recognized methods and principles for designing of S-CPSs is presented in Fig. 4. The platform and formal methods for CPSs development are recognized as background. Specific methods and approaches in the next phase help achieve self-awareness, robustness, autonomy, and self-adaptation of S-CPSs. In Table IV, the concepts included in Fig. 4 are supplemented with references to the key sources.

III. Technical and Open Issues of Run-Time Adaptation

A. Design Process

Self-adaptive behavior implies that certain development and change activities are shifted from development-time to run-time, while reassigning the responsibility for these activities from engineers to the system itself [95]. Models need to continue to live during run-time and adapt as changes occur while the software is running. To ensure dependability, analysis that the updated system models continue to satisfy the goals must be performed by continuous verification [95], [96]. The challenge is to infuse a systematic understanding of the alternatives for adaptive control into the design process.

B. Uncertainty, Reliability, and Safety

How can the openness-based uncertainty in system adaption be balanced with the need to control system adaption with respect to dependability issues [97]? Robustness, reliability, and safety, among others, are critical challenges because of uncertainties in the environment, such as security attacks and errors in physical devices. The design and implementation of networked control in CPS pose several challenges: guarantee of mission-critical QoS over wireless networks, tradeoff between control law design, and real-time computation constraints [98]. In the design and analysis of secure control algorithms we need to introduce a trust analysis of the CPS architecture, and realistic and rational adversary models [70]. How to trust into the complex system that is built from invalidated components? Shall we replace conventional specify-design-validate methodologies with a provide-smartness-and-set-objectives paradigm? Lee [99] emphasized the importance of security, reliability, and real-time assurance in CPSs.

C. Resource Management, Dynamic Composition, and Timing Requests

Time-limited operation or real-time properties are a mandatary requirement for several advanced CPSs. Lee [65] argued that beside the desired functionality and behavior of CPSs, it is even more important to model and analyze timing properties as part of their semantic correctness. García-Valls *et al.* [100] proposed a middleware architecture that supports time-deterministic configuration in distributed soft real-time environments with a software model based on services. The middleware contributes by including time-limited reconfiguration and service-based composition algorithms that are built on top of real-time resource management. As system complexity grows, the space of scenarios, modes, and conditions that must be tested grows exponentially. Certification is estimated to consume more than 50% of the resources required to develop new, safety-critical systems in the aviation industry. For the autonomous vehicle scenario to become reality, the human monitor must be replaced with a certified limiting algorithm that is capable of providing absolute guarantees about the vehicle's safety in the highly dynamic environment, such as urban streets [101].

A system with self-awareness, self-constraint generation, and run-time validation is a promising way forward [102]. S-CPSs have to adapt the software and behavior at run-time caused by the evolution of the system. García-Valls *et al.* [103] suggested the generation of tentative configurations at run-time. In the end, a new model is created that supports the new adapted situation [103]. Bersani and García-Valls [104] focused on online adaptation to support dynamic changes during run-time. They are using an autonomic manager (OLIVE) that performs online verification for a specific application. The verification manager is based on MAPE-K and uses reasoning on the architectural model [104].

D. Knowledge Management

Knowledge presentation is an open issue. How to connect abstract logic with real-world meanings in reliable and flexible

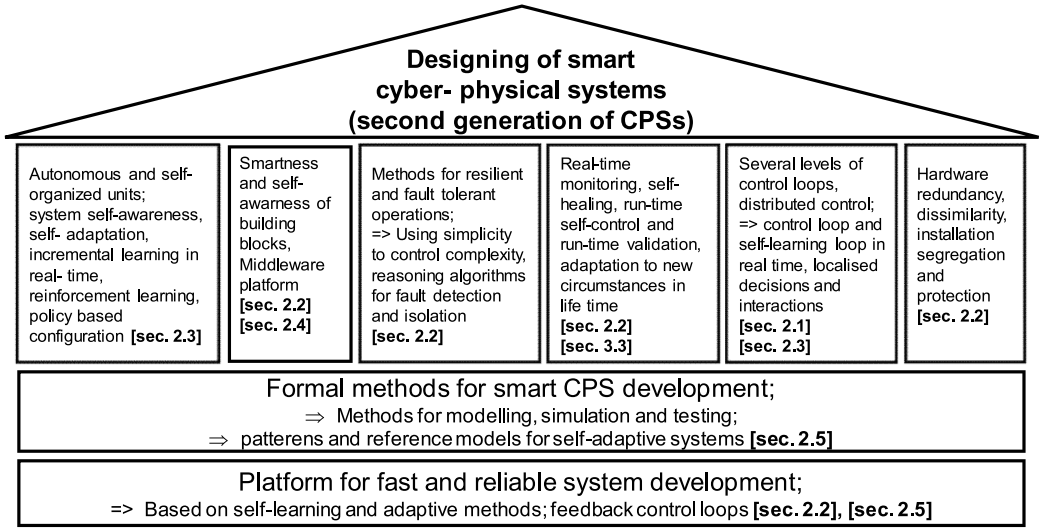


Fig. 4. Reasoning model concerning the methods and principles for designing S-CPSs.

ways? How to encapsulate rules, constraints, and mechanisms for self-adaptation and acquire and process knowledge about themselves, other service components, and their environment [81]. Despite the tremendous progress that has been made in recent decades, the field of machine learning and reasoning for adaptation and awareness is still in its infancy, and its methods are neither as universally applicable, nor as robust as would be desirable [51], [105]. Xu and Hua [20] proposed research strategies for industrial big data analytics including acquisition schemes, ontology modeling, deep neural network-based diagnostic methods, and self-organized reconfiguration mechanisms. High-level decision-making algorithms and theories, based on information collected from different sources, are necessary for system-wide reliability, efficiency, security, robustness, and autonomy of CPSs.

E. System Modeling

Model-based design generates many different models of various fidelity levels and tries to predict the behavior of a CPS, while it is virtually running. In the case of CPSs, complex interactions among the components cannot be modeled exhaustively, especially not across different design views. Another recognized issue is related to the nonlinear and nonincremental nature of multiscale CPSs. It means that adding new components lead to unpredicted and undesirable system behavior [1]. In recent years, modeling has evolved toward "meta-modeling" techniques and "meta-programmable" tools, which allow the introduction of domain-specific modeling languages. There are still several open issues. S-CPSs are by nature distributed on a continuum of heterogeneous platforms. However, the heterogeneity of platforms is also a requirement for fulfilling the needs of the different components of S-CPS [106]. There is still a lack of powerful languages, tools, and frameworks that could help realize adaptation processes and instrument sensors/effectors in a systematic manner [107].

IV. Discussion and Summary of the Explored Principles of Designing for Run-Time Adaptation

S-CPSs with run-time adaptation capability must implement some forms of self-learning and self-control. The overview of the current applications of S-CPS has revealed that they actually do not operate autonomously in safety critical cases. There is still a human in the loop, they operate in a controlled environment, or inside constrained limits [6], [108]. An important obstacle is legislation that requests validation of all operation modes before the product is placed on the market [109]. This is contradictory to the smart behavior of advanced CPSs. This proves our thesis that a smart, self-adaptive CPS needs a higher level of self-control, as presented in Table I. In the coming decade, we can see self-adaptiveness as a key feature of several CPSs applications. Advantages of smart and self-controlled systems are obvious: longer life span, optimized operation/improved efficiency, increased safety, increased reliability (self-maintenance), and lower operation costs. It is expected that the systems adaptation will happen in several smaller steps, and not as a radical change.

Existing legacy manufacturing systems have limited awareness of the CPS requirements, and revolutionary design approaches are necessary to achieve the overall system objectives [110]. "Industry 4.0 Working Group" [9] describes the vision, the basic technologies the idea aims at, and selected scenarios, but do not provide implementation details or design principles [37]. Different initiatives for Industry 4.0 have motivated research centers and companies to contribute through laboratory experiments or industrial applications [11], [22]. Most of the technical ingredients of Industry 4.0 are already available, and they just need to be integrated into the system [111]. The key expectations from future production system are smartness, self-organization, decentralized decisions, real-time control, and autonomous behavior. This paper presents a systematic overview of design principles for S-CPSs (Fig. 4, Table IV). This is a very important attribute for future industrial systems. However, the literature survey has found

TABLE III
OPEN ISSUES OF S-CPSS

Open issues	Current status	What need to be done?
Design process Formal methods	Focus to development-time of CPSs	Shift of activities to run-time from engineers to system itself
Uncertainty and safety	Specify → Design → Validate Methodology	Smartness and autonomy of sub-system
Resource management, and timing requests	Focus to system functionality and behavior	Time-limited reconfiguration, real-time resource management
Knowledge management	Not robust and universally applicable	Goal-oriented decision-making
System modeling and simulation	Linear and separated meta models	Non-linear CPSs modeling and simulation

that several aspects of S-CPS require additional research. The summary of open issues is presented in Table III. The current status of open issues is written in the second column, the target state is presented in the third column.

Linear principles dominate in the conventional technical system. Realization of the system adaptability (smartness) increases complexity of the system. A complex system has a myriad of elements and subsystems, therefore the logical consequences are increased uncertainty and reduced reliability. Development of complex technical system on the basis of linear principles is not economical any more. In using decentralized control loops, the overall system behavior emerges from the localized decisions and interactions [68].

Complex, nonhuman controlled and supervised operations of CPSs can be realized only if these systems are capable of building up a broadly based awareness, can reason and learn in varying contexts, can develop context-dependent operation plans, and can adapt themselves in order to implement that operation plan. These elements of smartness, which have to be considered in the early phase of system design, pose a new challenge that was not experienced in system engineering a number of decades ago. This new phenomenon of design has been referred to as the "transformation paradox of design." When the number of interacting components and the overall operational complexity of hardware and software system components increase, the opportunities and cases of having hardware and software faults may also increase. The system architecture and control is supposed to be resilient to these faults, and to be able to adapt to the new operational circumstances in real-time. We investigated the characteristic design principles, approaches and enablers that may facilitate the consideration of different levels of adaptation in the life cycle of CPSs.

The identified principles and enablers are presented in Table IV. Those place in the left column concern second generation. For the sake of comparison, we included also some of those which are applicable to the third generation of CPSs. Table IV provides links to specific references, which offer additional explanations on the specific principles and enablers. Several design principles that are normally used at a lower level designing of CPSs are also supposed to be applicable in an upper level. It has been observed that the set of design principles and methods related to the second generation of CPS are in continuous evolution. This is triggered by the increasing sophistication of software tools, reasoning mechanisms, and computing algorithms developed for self-learning, self-reasoning, and self-adaptation.

TABLE IV
DESIGN ENABLERS FOR THE SECOND GENERATION OF CPSS AND RELEVANT REFERENCES

Second generation of CPSs - Level 2 Known modes of changes	Third generation of CPSs - Level 3 Quasi-known/unknown models of changes
Run-time control (selecting the right mode) [5] [63]	Self-learning and system adaptation algorithm (inside preset limits); Incremental learning in real-time [74] [84]
Reasoning algorithm [73] Language for formal reasoning [112] Genetic algorithm [113] Neuron network [114] Fuzzy logic [115] [116]	Deep machine-learning [105] Reinforcement-learning [6] [8] [76][108] Big data analysis [20] [21]
Environment and system-awareness [77] [80] [78]	Policy-based configuration [112] [117] [118]
Modeling and simulation tools [86] [91] Holistic system modeling [1] DYNAMICO, reference model [47]	Self-aware building blocks [12] [71] [78] [82] [83] [84] [14] [16] [85] [87] [88]
Model-based development [93] [94] [119] Design patterns [48] [49]	Fault-tolerant design, model-based fault detection [24] [57] [58] [60] [61] [62] [63]
Real-time monitoring [6] [52] [54] [55] [118]	Run-time assurance inside preset limits [95] [96] [102] [119] [120] Run-time Enhancement of Trusted Computing [64]
Control with simplicity [59] [121]	Knowledge presentation [79] [81]
Reference model: MAPE-K + evolution [42] [122] Industry 4.0 [11] [17] [41] Platform for development: SEAMS [31], DEECo [45]	Temp. for self-adaptive design [50] Reference model: MAPE-K + evolution [43] Dynamic adaptation [54] [123] Real-time reconfiguration and validation [100] [103] [104] OMA-Cy- architecture for middleware technology [67]

The literature review has also shown that several smart elements have already reached a quite mature state, and that they are extensively used in second generation CPSs, e.g., neural network-based reasoning, context information processing, and situation awareness building. The fusion of the cyber world and the physical world obviously requires design principles for safety and timing. This was also considered in the survey of design principles, which is deemed to be a major contribution of this paper. In addition, we have also analyzed the available tools and methods, and recognized some open issues. The overview of the design principles in Table IV can be used by system developers for selecting design tools and by researchers to identify weakly supported areas in the context of tools for designing S-CPSs.

Self-adaptive behavior implies that certain development and change activities are shifted from development-time to run-time. Everything cannot be specified in the design phase in complex systems that are built for an operational time frame of several decades. The system has to be able of adaptation to new circumstances that appear in the life cycle. These are all facts and define the system autonomous self-adaptive features: system architecture enables adaptation in run-time; building blocks are autonomous and context aware; there is a self-learning mechanism that enables system adaptation inside a predefined constraint; the system is resilient to internal or external changes/faults and it is able of self-healing; and the system is able of self-validation in run-time inside the predefined constraint. On the basis of these conditions, development of second generation CPSs will become interesting for a wider range of applications. Another precondition is also a platform that gives the basic backbone to the system and enables smart integration. Examples of platforms like DEECo and SeSaMe for CPSs development and testing are already available and discussed in [124] and [125]. With the help of these, system developers can focus on the core of the problem, setting of proper goals, and specifying the system policy.

V. Conclusion

By their conceptualization, S-CPSs are the systems that have to be design to be self-aware and self-adapting. This entails that they change their behavior and performance to adapt to new operational circumstances and/or opportunities through their life cycle. Delegation a part of the design tasks to these systems, making them capable to change themselves in a trustful way, and providing them with the potential of acquiring new resources that are needed for building awareness and functional/structural adaptation are new challenges not experiences ever before. The development above-mentioned capabilities point into the direction of a human supervised, but essentially largely nondeterministic operation.

On the other hand, safety-critical systems are still requested to operate in a deterministic way. It does not mean that S-CPS will not be applicable in this range of applications, but it does definitely indicate the need to work out the relevant design principles that make run-time adaptation of S-CPSs dependable, without constraining their freedom for finding more favoring operational objectives. A new way of validation (like run-time validation), which will prove that smart CPSs are acceptable in spite of all run-time variation, was already recognized as an important issue, but it is currently an unsolved matter. This paper has assessed the state-of-the-art of design principles for S-CPSs, and cast light on many research areas that need additional systematic research (Table III).

The contribution of this paper manifest in a comprehensive classification of CPSs according to their cognitive capabilities (intellectualization) and the level of self-organization capability. The second generation of CPSs, which are operating under quasi-known changes and with tunable objectives, has a large potential for a wide-spread use in many industrial and social domains in the near future. That was the main reason why they have been put into the focus of our reported research. According to the findings, self-adaptation can be conducted only inside predefined limits, and with constraint resources due to the safety requirements defined in the design phase. It means that in addition to the conventional system engineering tasks, designers of S-CPSs should design the window of self-adaptation, and should determine the range of resources applicable by the system for adaptive operations. These are seen as important issues for a follow up research.

This paper also contributed a holistic map of design principles that are deemed to be necessary for effective S-CPSs development (Fig. 4). From a practical perspective, the complexity of the development process of S-CPSs has to be reduced. It can be achieved by developing and applying smart platforms (reasoning, simulating, context-handling, informing, messaging, etc.), with may incorporate all necessary modules needed for the functionalities mentioned in the round brackets, and even smart objects in the form of context-aware cooperative agents. The smart building blocks of software and hardware may be reused, replaced, or reconfigured in a simple and reliable way by any system integrator. Smartness shall replace complexity: the prevailing hierarchical system structure has to be replaced with anticipating and collaborating system actors, which manifest as autonomous and self-organizing units connected through cyber space.

As argued earlier, improvements can be expected based on the contemplation of the design paradox that a system with higher level adaptation freedom requires a higher level of self-control. As system complexity and unpredictability of behavior are increasing, it is a must to have context-aware and autonomously regulated building blocks. On the basis of the presented literature review, it has been recognized that development of middleware platforms for faster and more reliable architecting and operation of S-CPSs with improved control of resources and real-time properties are needed. Real-time properties are intrinsic for several advanced CPSs. S-CPSs may have numerous potential applications, and smart manufacturing is and will be one of them. Advantages of smart and self-controlled systems are longer life span, improved efficiency, increased safety, increased reliability, and lower operation costs. It is expected that the course of system adaptation will happen in several smaller steps, and not as a one-time radical change.

References

[1] I. Horváth and B. H. Gerritsen, "Cyber-physical system: Concepts, technologies and manifestation," in *Proc. TMCE*, 2012, Karlsruhe, Germany, 2012, pp. 1–16.

[2] J. Fiksel, "Designing resilient, sustainable systems," *Environ. Sci. Technol.*, vol. 37, no. 23, pp. 5330–5339, 2003.

[3] U. Aßmann, S. Götz, J.-M. Jezequel, B. Morin, and M. Trapp, "A reference architecture and roadmap for models@run.time systems," in *Models@Run.Time*. Berlin, Germany: Springer, 2014, pp. 1–18.

[4] I. Horváth, Z. Rusak, and Y. Li, "Order beyond chaos: Introducing the notion of generation to characterize the continuously evolving implementations of cyber-physical systems," in *Proc. ASME IDETC/CIE*, vol. 1. Cleveland, OH, USA, 2017, p. 14.

[5] A. H. Khan, Z. H. Khan, and S. H. Khan, "Optimized reconfigurable autopilot design for an aerospace CPS," in *Intelligence for Decision Support in Cyber-Physical Systems*. Singapore: Springer, 2014.

[6] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *Int. J. Robot. Res.*, vol. 32, no. 3, pp. 263–279, 2013.

[7] J. Chen *et al.*, "CPS modeling of CNC machine tool work processes using an instruction-domain based approach," *Engineering*, vol. 1, no. 2, pp. 247–260, 2015.

[8] J. de Lope, D. Maravall, and Y. Quiñonez, "Self-organizing techniques to improve the decentralized multi-task distribution in multi-robot systems," *Neurocomputing*, vol. 163, pp. 47–55, Sep. 2015.

[9] H. Kagermann, W. Wahlster, and J. Helbig, Eds., "Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0," Working Group, Forschungsunion, Acatech—Nat. Acad. Sci. Eng., Frankfurt, Germany, 2013.

[10] L. Wang, M. Törngren, and M. Onori, "Current status and advancement of cyber-physical systems in manufacturing," *J. Manuf. Syst.*, vol. 37, pp. 517–527, Oct. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612515000400

[11] Y. Liao, F. Deschamps, E. F. R. Loures, and L. F. P. Ramos, "Past, present and future of industry 4.0—A systematic literature review and research agenda proposal," *Int. J. Prod. Res.*, vol. 55, no. 12, pp. 3609–3629, 2017.

[12] M. F. Filho, Y. Liao, E. R. Loures, and O. Canciglieri, "Self-aware smart products: Systematic literature review, conceptual design and prototype implementation," *Procedia Manuf.*, vol. 11, pp. 1471–1480, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978917304869

[13] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *J. Ind. Inf. Integr.*, vol. 6, pp. 1–10, Jun. 2017.

[14] G. Kortuem, F. Kawsar, V. Sundramoorthy, and D. Fitton, "Smart objects as building blocks for the Internet of Things," *IEEE Internet Comput.*, vol. 14, no. 1, pp. 44–51, Jan./Feb. 2010.

[15] F. Tao, Y. Wang, Y. Zuo, H. Yang, and M. Zhang, "Internet of Things in product life-cycle energy management," *J. Ind. Inf. Integr.*, vol. 1, pp. 26–39, Mar. 2016.

[16] L. D. Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.

[17] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: A self-organized multi-agent system with big data based feedback and coordination," *Comput. Netw.*, vol. 101, pp. 158–168, Jun. 2016.

[18] H. S. Yan and C. G. Xue, "Decision-making in self-reconfiguration of a knowledgeable manufacturing system," *Int. J. Prod. Res.*, vol. 45, no. 12, pp. 2735–2758, 2007.

[19] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg, "How virtualization, decentralization and network, building change the manufacturing landscape: An industry 4.0 perspective," *Int. J. Mech. Aerosp. Ind. Mechtronics Manuf. Eng.*, vol. 8, no. 1, pp. 37–44, 2014.

[20] X. Xu and Q. Hua, "Industrial big data analysis in smart factory: Current status and research strategies," *IEEE Access*, vol. 5, pp. 17543–17551, 2016.

[21] J. Wan *et al.*, "A manufacturing big data solution for active preventive maintenance," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 2039–2047, Aug. 2017.

[22] K.-D. Thoben, S. Wiesner, and T. Wuest, "'Industrie 4.0' and smart manufacturing—A review of research issues and application examples," *Int. J. Autom. Technol.*, vol. 11, no. 1, pp. 4–16, 2017.

[23] V. Tuominen, "The measurement-aided welding cell—Giving sight to the blind," *Int. J. Adv. Manuf. Technol.*, vol. 86, nos. 1–4, pp. 371–386, 2016.

[24] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu, "Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data," *Mech. Syst. Signal Process.*, vols. 72–73, pp. 303–315, May 2016.

[25] L. Monostori, "Cyber-physical production systems: Roots, expectations and R&D challenges," *Procedia CIRP*, vol. 17, pp. 9–13, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827114003497

[26] M. M. Herterich, F. Uebernickel, and W. Brenner, "The impact of cyber-physical systems on industrial services in manufacturing," *Procedia CIRP*, vol. 30, pp. 323–328, 2015.

[27] W. S. Black, P. Haghi, and K. B. Ariyur, "Adaptive systems: History, techniques, problems, and perspectives," *Systems*, vol. 2, no. 4, pp. 606–660, 2014.

[28] Z. Ding, Y. Zhou, and M. Zhou, "Modeling self-adaptive software systems with learning petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 4, pp. 483–498, Apr. 2016.

[29] Y. Brun *et al.*, "A design space for self-adaptive systems," in *Software Engineering for Self-Adaptive Systems II*. Berlin, Germany: Springer, 2013, pp. 33–50.

[30] B. H. C. Cheng *et al.*, *Software Engineering for Self-Adaptive Systems: A Research Roadmap* (LNCS 5525). Heidelberg, Germany: Springer, 2009, pp. 1–26.

[31] D. Weyns, S. Malek, and J. Andersson, "FORMS: A formal reference model for self-adaptation," in *Proc. 7th IEEE Int. Conf. Auton. Comput. (ICAC)*, Washington, DC, USA, 2010, pp. 205–214.

[32] Y. Yin, Y. Zeng, X. Chen, and Y. Fan, "The Internet of Things in healthcare: An overview," *J. Ind. Inf. Integr.*, vol. 1, pp. 3–13, Mar. 2016.

[33] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Comput.*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[34] J. Kramer and J. Magee, "Self-managed systems: An architectural challenge," in *Proc. Future Softw. Eng.*, 2007, pp. 259–268.

[35] F. Civerchia *et al.*, "Industrial Internet of Things monitoring solution for advanced predictive maintenance applications," *J. Ind. Inf. Integr.*, vol. 7, pp. 4–12, Sep. 2017.

[36] C. Faller and D. Feldmüller, "Industry 4.0 learning factory for regional SMEs," *Procedia CIRP*, vol. 32, pp. 88–91, 2015.

[37] M. Hermann, T. Pentek, and B. Otto, "Design principles for Industrie 4.0 scenarios," in *Proc. Hawaii Int. Conf. Syst. Sci. (HICSS)*, 2016, pp. 3928–3937.

[38] S. Weyer, M. Schmitt, M. Ohmer, and D. Gorecky, "Towards industry 4.0—Standardization as the crucial challenge for highly modular, multi-vendor production systems," *IFAC PapersOnLine*, vol. 48, no. 3, pp. 579–584, 2015.

[39] A. J. C. Trappey, C. V. Trappey, U. H. Govindarajan, J. J. Sun, and A. C. Chuang, "A review of technology standards and patent portfolios for enabling cyber-physical systems in advanced manufacturing," *IEEE Access*, vol. 4, pp. 7356–7382, 2016.

[40] A. J. C. Trappey, C. V. Trappey, U. H. Govindarajan, A. C. Chuang, and J. J. Sun, "A review of essential standards and patent landscapes for the Internet of Things: A key enabler for industry 4.0," *Adv. Eng. Informat.*, vol. 33, pp. 208–229, Aug. 2017.

[41] J. Lee, B. Bagheri, and H.-A. Kao, "A Cyber-physical systems architecture for industry 4.0-based manufacturing systems," *Manuf. Lett.*, vol. 3, pp. 18–23, Jan. 2015.

[42] P. Arcaini, E. Riccobene, and P. Scandurrav, "Modeling and analyzing MAPE-K feedback loops for self-adaptation," in *Proc. IEEE SEAMS*, Florence, Italy, 2015, pp. 13–23.

[43] A. M. Sharifloo, A. Metzger, C. Quinton, L. Baresi, and K. Pohl, "Learning and evolution in dynamic software product lines," in *Proc. IEEE SEAMS*, Austin, TX, USA, 2016, pp. 158–164.

[44] P. Vromant, D. Weyns, S. Malek, and J. Andersson, "On interacting control loops in self-adaptive systems," in *Proc. SEAMS*, 2011, pp. 202–207.

[45] M. Kit, I. Gerostathopoulos, T. Bures, P. Hnetynka, and F. Plasil, "An architecture framework for experimentations with self-adaptive cyber-physical systems," in *Proc. IEEE SEAMS*, Florence, Italy, 2015, pp. 93–96.

[46] N. Capodieci, E. Hart, and G. Cabri, "Artificial immunology for collective adaptive systems design and implementation," *ACM Trans. Auton. Adapt. Syst.*, vol. 11, no. 2, 2016, Art. no. 6.

[47] N. M. Villegas, G. Tamura, H. A. Muller, L. Duchien, and R. Casallas, *DYNAMICO: A Reference Model for Governing Control Objectives and Context Relevance in Self-Adaptive Software Systems* (LNCS 7475), R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, Eds. Heidelberg, Germany: Springer, 2013, pp. 265–293.

[48] A. J. Ramirez and B. H. C. Cheng, "Design patterns for developing dynamically adaptive systems," in *Proc. SEAMS*, 2010, pp. 49–58.

[49] D. Weyns *et al.*, "On patterns for decentralized control in self-adaptive systems," in *Software Engineering for Self-Adaptive Systems II*. Heidelberg, Germany: Springer, 2013, pp. 76–107.

[50] D. G. D. L. Iglesia and D. Weyns, "MAPE-K formal templates to rigorously design behaviors for self-adaptive systems," *Trans. Auton. Adapt. Syst.*, vol. 10, no. 3, 2015, Art. no. 15.

[51] M. Hölzl and T. Gabor, *Reasoning and Learning for Awareness and Adaptation* (LNCS 8998). Cham, Switzerland: Springer, 2015, pp. 249–290

[52] P. O. Meredith, D. Jin, D. Griffith, F. Chen, and G. Rosu, "An overview of the MOP runtime verification framework," *Int. J. Softw. Tools Technol. Transf.*, vol. 14, no. 3, pp. 249–289, 2012.

[53] S. Ruiz-Arenas, I. Horváth, R. Mejía-Gutiérrez, and E. Z. Opiyo, "Towards the maintenance principles of cyber-physical systems," *Strojniški vestnik J. Mech. Eng.*, vol. 60, no. 12, pp. 815–831, 2014.

[54] B. Morin, O. Barais, J.-M. Jezequel, F. Fleurey, and A. Solberg, "Models@Run.time to support dynamic adaptation," *Computer*, vol. 42, no. 10, pp. 44–51, Oct. 2009.

[55] A. Bennaceur *et al.*, *Mechanisms for Leveraging Models at Runtime in Self-Adaptive Software*, in *Models@Run.Time* (LNCS 8378). Cham, Switzerland: Springer, 2014, pp. 19–46.

[56] P. Alho and J. Mattila, "Service-oriented approach to fault tolerance in CPSs," *J. Syst. Softw.*, vol. 105, pp. 1–17, Jul. 2015.

[57] M. Chen, P. Shi, and C.-C. Lim, "Adaptive neural fault-tolerant control of a 3-DOF model helicopter system," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 2, pp. 260–270, Feb. 2016.

[58] H. Mo and M. Xie, "A dynamic approach to performance analysis and reliability improvement of control systems with degraded components," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 10, pp. 1404–1414, Oct. 2016.

[59] L. Sha, "Using simplicity to control complexity," *IEEE Softw.*, vol. 18, no. 4, pp. 20–28, Jul./Aug. 2001.

[60] H. Noura, D. Theilliol, J.-C. Ponsart, and A. Chamseddine, *Fault-Tolerant Control Systems, Design and Practical Applications*. London, U.K.: Springer, 2009.

[61] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 3, pp. 636–653, May 2010.

[62] P. Goupil, "Industrial practices in fault tolerant control," in *Fault Tolerant Flight Control: A Benchmark Challenge*, C. Edwards, T. Lombaerts, and H. Smaili, Eds. Heidelberg, Germany: Springer, 2010, pp. 157–166.

[63] P. Chu, J. A. Mulder, and J. Breeman, "Real-time identification of aircraft physical models for fault tolerant flight control," in *Fault Tolerant Flight Control: A Benchmark Challenge*, C. Edwards, T. Lombaerts, and H. Smaili, Eds. Heidelberg, Germany: Springer, 2010, pp. 129–153.

[64] M. M. Farag, "Architectural enhancements to increase trust in cyber-physical systems containing untrusted software and hardware," Ph.D. dissertation, Faculty Virginia Polytechnic Inst. State Univ., Blacksburg, VA, USA, 2012. [Online]. Available: https://search.proquest.com/docview/1512627509?pq-origsite=gscholar

[65] E. A. Lee, "Time for high-confidence cyber-physical systems," in *Proc. Perform. Metrics Intell. Syst. Workshop*, College Park, MD, USA, 2012, pp. 1–30.

[66] N. Mohamed, J. Al-Jaroodi, S. Lazarova-Molnar, and I. Jawhar, "Middleware to support cyber-physical systems," in *Proc. IEEE 35th Int. Perform. Comput. Commun. Conf.*, Las Vegas, NV, USA, 2016, pp. 1–3.

[67] M. García-Valls and R. Baldoni, "Adaptive middleware design for CPS: Considerations on the OS, resource managers, and the network runtime," in *Proc. ARM*, Vancouver, BC, Canada, 2015, Art. no. 3.

[68] T. Cucinotta *et al.*, "A real-time service-oriented architecture for industrial automation," *IEEE Trans. Ind. Informat.*, vol. 5, no. 3, pp. 267–277, Aug. 2009.

[69] R. de Lemos *et al.*, "Software engineering for self-adaptive systems: A second research roadmap," in *Software Engineering for Self-Adaptive Systems II* (LNCS 7475), R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, Eds. Heidelberg, Germany: Springer, 2013, pp. 1–32.

[70] A. A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst. (ICDCSW)*, Beijing, China, 2008, pp. 495–500.

[71] A. A. Minai, D. Braha, and Y. Bar-Yam, "Complex engineered systems: A new paradigm," in *Complex Engineered Systems*. Heidelberg, Germany: Springer, 2006, pp. 1–21.

[72] S.-W. Cheng, V. V. Poladian, D. Garlan, and B. Schmerl, "Improving architecture-based self-adaptation through resource prediction," in *Software Engineering for Self-Adaptive Systems*. Heidelberg, Germany: Springer-Verlag, 2009, pp. 48–70.

[73] A. Håkansson, R. Hartung, and E. Moradian, "Reasoning strategies in smart cyber-physical systems," *Procedia Comput. Sci.*, vol. 60, pp. 1575–1584, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050915023947

[74] S. Gallacher, E. Papadopoulou, N. K. Taylor, and M. H. Williams, "Learning user preferences for adaptive pervasive environments: An incremental and temporal approach," *ACM Trans. Auton. Adapt. Syst.*, vol. 8, no. 1, 2013, Art. no. 5.

[75] F. Wotawa, "Adaptive autonomous systems—From the system's architecture to testing," in *Leveraging Applications of Formal Methods, Verification, and Validation*, vol. 336. Heidelberg, Germany: Springer, 2011, pp. 76–90.

[76] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[77] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 2, no. 4, pp. 263–277, 2007.

[78] J.-Y. Hong, E.-H. Suh, and S.-J. Kim, "Context-aware systems: A literature review and classification," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 8509–8522, 2009.

[79] D. Fisch, M. Jänicke, E. Kalkowski, and B. Sick, "Techniques for knowledge acquisition in dynamically changing environments," *ACM Trans. Auton. Adapt. Syst.*, vol. 7, no. 1, 2012, Art. no. 16.

[80] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the Internet of Things: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 414–454, 1st Quart., 2014.

[81] E. Vassev and M. Hinchey, "Knowledge representation for adaptive and self-aware systems," in *Software Engineering for Collective Autonomic Systems* (LNCS 8998). Cham, Switzerland: Springer, 2015, pp. 221–247.

[82] H. Schmeck, C. Müller-Schloer, E. Çakar, M. Mnif, and U. Richter, "Adaptivity and self-organization in organic computing systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 5, no. 3, 2010, Art. no. 10.

[83] N. Dutt, A. Jantsch, and S. Sarma, "Toward smart embedded systems: A self-aware system-on-chip (SoC) perspective," *ACM Trans. Embedded Comput. Syst.*, vol. 15, no. 2, 2016, Art. no. 22.

[84] B. Subagdja and A.-H. Tan, "Interactive teachable cognitive agents: Smart building blocks for multiagent systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 12, pp. 1724–1735, Dec. 2016.

[85] D. Ye, M. Zhang, and A. V. Vasilakos, "A survey of self-organization mechanisms in multiagent systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 441–461, Mar. 2017.

[86] H. Kopetz, "The complexity challenge in embedded system design," in *Proc. ISORC*, Orlando, FL, USA, 2008, pp. 3–12.

[87] K. Alexopoulos, S. Makris, V. Xanthakis, K. Sipsas, and G. Chryssolouris, "A concept for context-aware computing in manufacturing: The white goods case," *Int. J. Comput. Integr. Manuf.*, vol. 29, no. 8, pp. 839–849, 2016.

[88] R. J. Anthony, D. Chen, M. Pelc, M. Persson, and M. Törngren, "Context-aware adaptation in DySCAS," *Electron. Commun. EASST*, vol. 19, pp. 1–15, 2009.[Online]. Available: https://journal.ub.tu-berlin.de/eceasst/article/view/245/232

[89] E. A. Lee, "The past, present and future of cyber-physical systems: A focus on models," *Sensors*, vol. 15, no. 3, pp. 4837–4869, 2015.

[90] D. Gürdür, J. El-Khoury, T. Seceleanu, and L. Lednicki, "Making interoperability visible: Data visualization of cyber-physical systems development tool chains," *J. Ind. Inf. Integr.*, vol. 4, pp. 26–34, Dec. 2016.

[91] J. Vanhuyse *et al.*, "Nonlinear model predictive control design using AMESIM models," in *Proc. TMCE*, Aix-en-Provence, France, 2016, pp. 143–152.

[92] S. Pourtalebi and I. Horváth, "Towards a methodology of system manifestation features-based pre-embodiment design," *J. Eng. Design*, vol. 27, nos. 4–6, 2016, pp. 232–268.

[93] A. Canedo, E. Schwarzenbach, and M. A. A. Faruque, "Context-sensitive synthesis of executable functional models of cyber-physical systems," in *Proc. ICCPS*, Philadelphia, PA, USA, 2013, pp. 99–108.

[94] B.-H. Schlingloff, "Cyber-physical systems engineering," in *Engineering Trustworthy Software Systems* (LNCS 9506), Z. Liu and Z. Zhang, Eds. Cham, Switzerland: Springer, 2016, pp. 256–289.

[95] J. Andersson *et al.*, *Software Engineering Processes for Self-Adaptive Systems* (LNCS 7475), R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, Eds. Heidelberg, Germany: Springer, pp. 51–75, 2013.

[96] L. Baresi and C. Ghezzi, "The disappearing boundary between development-time and run-time," in *Proc. FSE/SDP FoSER*, Santa Fe, NM, USA, 2010, pp. 17–22.

[97] C. Bartelet, A. Rausch, and K. Rehfeld, "Quo vadis cyber-physical systems: Research areas of cyber-physical ecosystems," in *Proc. CTSE*, Bergamo, Italy, 2015, pp. 22–25.

[98] K. J. Park, R. Zheng, and X. Liu, "Cyber-physical systems: Milestones and research challenges," *Comput. Commun.*, vol. 36, no. 1, pp. 1–7, 2012.

[99] E. A. Lee, "Cyber physical systems: Design challenges," in *Proc. IEEE ISORC*, 2008, pp. 363–369.

[100] M. García-Valls, I. R. López, and L. F. Villar, "iLAND: An enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 228–236, Feb. 2013.

[101] M. Clark *et al.*, "A study on run time assurance for complex cyber physical systems," Air Force Res. Lab., Vanderbilt Uni., Nashville, TN, USA, Iowa State Univ., Ames, IA, USA, Univ. Pennsylvania, Philadelphia, PA, USA, and Galois Inc., Portland, OR, USA, Rep., 2013. [Online]. Available: http://www.dtic.mil/docs/citations/ADA585474

[102] S. Mitsch and A. Platzer, "ModelPlex: Verified runtime validation of verified cyber-physical system models," in *Run-Time Verification* (LNCS 8734), B. Bonakdarpour and S. A. Smolka, Eds. Cham, Switzerland: Springer, 2014, pp. 199–214.

[103] M. García-Valls, D. Perez-Palacin, and R. Mirandola, "Time-sensitive adaptation in CPS through run-time configuration generation and verification," in *Proc. IEEE 38th ICSAC*, 2014, pp. 332–337.

[104] M. M. Bersani and M. García-Valls, "Online verification in cyber-physical systems: Practical bounds for meaningful temporal costs," *J. Softw. Evol. Process*, vol. 30, 2018, Art. no. e1880. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/smr.v30.3/issuetoc, doi: 10.1002/smr.1880.

[105] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning in manufacturing: Advantages, challenges, and applications," *Prod. Manuf. Res.*, vol. 4, no. 1, pp. 23–45, 2016.

[106] B. Morin, F. Fleurey, and O. Barais, "Taming heterogeneity and distribution in sCPS," in *Proc. IEEE/ACM SESCPS (SEsCPS)*, 2015, pp. 40–43.

[107] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Trans. Auton. Adapt. Syst.*, vol. 4, no. 2, pp. 1–42, 2009.

[108] B. D. Argalla, S. Chernovab, M. Velosob, and B. Browninga, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[109] K. Alexander and P. J. Clarkson, "A validation model for the medical devices industry," *J. Eng. Design*, vol. 13, no. 3, pp. 197–204, 2002.

[110] V. Gunes, S. Peter, T. Givargis, and F. Vahid, "A survey on concepts, applications, and challenges in cyber-physical systems," *KSII Trans. Internet Inf. Syst.*, vol. 8, no. 12, pp. 4242–4268, 2014.

[111] R. Drath, and A. Horch, "Industrie 4.0: Hit or hype? [Industry forum]," *IEEE Ind. Electron. Mag.*, vol. 8, no. 2, pp. 56–58, Jun. 2014.

[112] R. D. Nicola, M. Loreti, R. Pugliese, and F. Tiezzi, "A formal approach to autonomic systems programming: The SCEL language," *ACM Trans. Auton. Adapt. Syst.*, vol. 9, no. 2, 2014, Art. no. 7.

[113] F. O. Karray and C. W. Silva, *Soft Computing and Intelligent Systems Design—Theory, Tools, and Applications*. New York, NY, USA: Addison-Wesley, 2004.

[114] L. Wang, K. C. Tan, and C. M. Chew, *Evolutionary Robotics: From Algorithms to Implementations*, vol. 28. Singapore: World Sci., 2006.

[115] C. Chrysostomou, C. Djouvas, and L. Lambrinos, "Fuzzy logic-based adaptive decision support in autonomous vehicular networks," in *Computational Intelligence for Decision Support in Cyber-Physical Systems*, vol. 540, Z. H. Khan, A. Ali, and Z. Riaz, Eds. Singapore: Springer, 2014, pp. 215–236.

[116] M. Abid, A. Q. Khan, M. Rehan, and Haroon-ur-Rasheed, "TS fuzzy approach for fault detection in nonlinear cyber physical systems," in *Computational Intelligence for Decision Support in Cyber-Physical Systems*, vol. 540. Singapore: Springer, 2014, pp. 421–447. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-981-4585-36-1_14

[117] R. J. Anthony, "A policy-definition language and prototype implementation library for policy-based autonomic systems," in *Proc. ICAC*, Dublin, Ireland, 2006, pp. 265–276.

[118] J. Campos *et al.*, "Robust regulation adaptation in multi-agent systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 8, no. 3, 2013, Art. no. 13.

[119] R. Weissnegger *et al.*, "Simulation-based verification of automotive safety-critical systems based on EAST-ADL," *Procedia Comput. Sci.*, vol. 83, pp. 245–252, 2016.

[120] D. Schneider and M. Trapp, "Conditional safety certification of open adaptive systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 8, no. 2, 2013, Art. no. 8.

[121] L. Sha and J. Meseguer, "Design of complex cyber physical systems with formalized architectural patterns," in *Software-Intensive Systems* (LNCS 5380), M. Wirsing, J. P. Banâtre, M. Hölzl, and A. Rauschmayer, Eds. Heidelberg, Germany: Springer, 2008, pp. 92–100.

[122] D. Gorlan *et al.*, "Rainbow: Architecture-based self-adaptation with reusable infrastructure," *Computer*, vol. 37, no. 10, pp. 46–54, Oct. 2004.

[123] D. Weyns, M. U. Iftikhar, D. G. de la Iglesia, and T. Ahmad, "A survey of formal methods in self-adaptive systems," in *Proc. C3S2E*, 2012, pp. 67–79.

[124] T. Bures *et al.*, "DEECo: An ensemble-based component system," in *Proc. ACM CBSE*, 2013, pp. 81–90.

[125] L. Baresi, S. Guinea, and A. Shahzada, "SeSaMe: Towards a semantic self adaptive middleware for smart spaces," in *Engineering Multi-Agent Systems* (LNCS 8245). Heidelberg, Germany: Springer, 2013, pp. 1–18.

Jože Tavčar was born in 1966. He received the Ph.D. degree from the University of Ljubljana, Ljubljani, Slovenia, in 1999.

He has been an Associate Professor with the Faculty of Mechanical Engineering, University of Ljubljana since 2011. He has over a decade of industrial experiences. He was a Quality Manager with Iskra Mehanizmi Company, Slovenia, and as a Product Developer and a Researcher with Domel Electric Motor Company, Slovenia. His current research interests include information flow analyses, concurrent engineering, quality systems, and engineering design techniques.

Imre Horváth was born in 1954. He received the M.Sc. degrees in mechanical engineering and engineering education in 1978 and 1980, respectively, and the Dr.Univ., Ph.D., and the C.D.Sc. degrees.

He has been a Full Professor with the Faculty of Industrial Design Engineering, Delft University of Technology, Delft, The Netherlands, since 1997. He is the Head of the Cyber-Physical System Design Research Group. He has coauthored over 380 papers and articles. His current research interests include cognitive engineering of CPSs, systematic design research, and personalized/socialized system development.

Dr. Horváth initiated the TMCE Symposia. He is an Emeritus Editor-in-Chief of the *Journal of Computer-Aided Design*, and an Associate Editor of the *Journal of Engineering Design*. He served as the Chair for the Executive Committee of the CIE Division. He is a Fellow of the ASME.