Designing Cyber-Physical Systems for Runtime Self-Adaptation
Knowing More about What We Miss...

Horváth, Imre; Tavčar, Jože

DOI
10.3233/JID210030

Publication date
2021

Document Version
Accepted author manuscript

Published in
Journal of Integrated Design and Process Science

Transactions of the SDPS:
Journal of Integrated Design and Process Science
XX (XXXX) XX-XX
DOI 10.3233/jid-201x-xxxx
http://www.sdpsnet.org

Designing Cyber-Physical Systems for Runtime Self-Adaptation:

Knowing More about What We Miss …

Imre Horváth[a] and Jože Tavčar[b]

[a] *Professor emeritus, Faculty of Industrial Design Engineering, Delft University of Technology, Delft, the Netherlands.*
[b] *Senior lecturer at the Product Development Division of the Faculty of Engineering, University of Lund, Sweden.*

Abstract

Keywords: Cyber-physical systems, programmed adaptation, runtime adaptation, self-adaptation, self-evolution, self-supervision, autonomous systems, open issues

1. First things first – Our view on cyber-physical systems

We live in the age of an extensive scientific, technological, and paradigmatic convergence [1]. One of the strongest current trends is the integration of social science, cognitive science, biotechnologies, information technologies, and nanotechnology (SCBIN) that enables fusion of bits, atoms, neurons, genes, and memes [2]. Graphically depicted in **Figure 1**, this accelerating merge process is often referred to as the bits-atoms-neurons-genes-memes (b.a.n.g.m.) revolution [3]. Cyber-physical systems (CPSs) represent practical examples of the integration of bits and atoms in human and social contexts, but they also make steps towards integration of neurons and genes into system implementations [4]. The move towards integration of neurons is exemplified by the interest in cyber-bio-physical (CBP) systems (e.g. assistive and corrective implants [5], and artificial limbs/augments [6]), while the results in the latter field are epitomized by gentelligent systems [7]. Consequently, engineered systems are going through a metamorphosis, and the significance of purely hardware (HW), software (SW), and cyberware (CW) systems is shrinking and their places are taken over quickly by heterogeneous and intellectualized systems. From the perspective of system adaptation, the current trends imply the need for a concurrent change of the HW, SW, and CW elements in runtime, in a synergic (compositional) manner. Theoretically, but also practically, the largest challenge in this context is that the operational changes of the HW constituents happen in the spatial-temporal space, the changes of the SW constituents in the logical-temporal space, and those of the CW constituents in the syntactic and semantic spaces.

In our view, software and data/knowledge integrated cyber-physical systems (CPSs): (i) include one or more independent (self-contained) or functionally networked actor nodes, (ii) are characterized by a deep penetration into real-life physical processes, (iii) operate based on multiple sensing-computing-adjusting

 Published by IOS Press

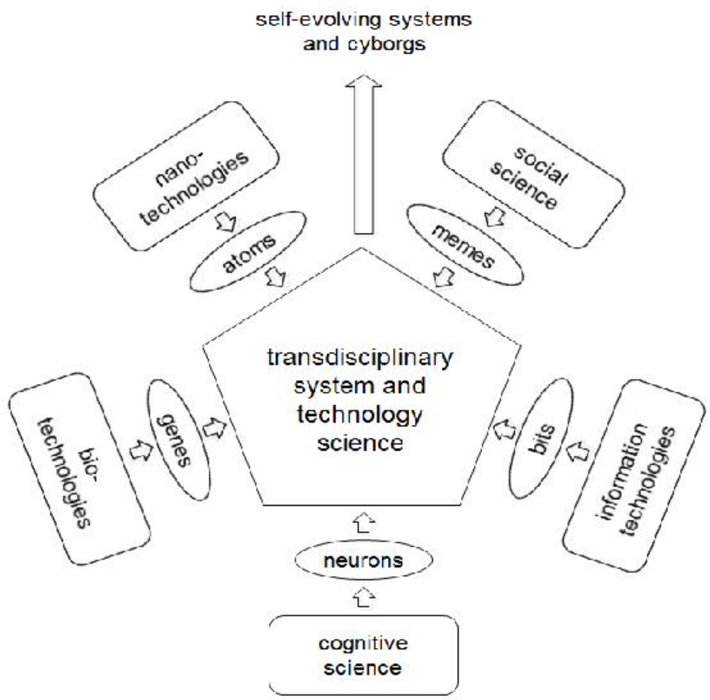

Figure 1: Merger of technologies and disciplines

loops or sensing-reasoning-learning-planning-adapting loops, (iv) provide tailored services and avail resources dynamically in human, social, and industrial applications contexts, (v) have abilities to extend their problem solving knowledge and computational mechanisms (system intelligence), (vi) may manifest as part of a purposefully and synergistically arranged system of systems, and (vii) evolve through generations [8]. Cybernetization of complex engineered systems seems to terminate with highly intellectualized and autonomously operating, but cognitively and socially embedded systems [9]. If, in sociotechnical systems, the technical parts manifest as CPSs, then researchers talk about social-cyber-physical systems, whose adaptation may be according to the principles of centrality of the norms and policy of autonomy, and not only to operational goals and affordances [10].

Though the complex phenomenon of system adaptation is a current hot issue, it is known only partially in the case of complex engineered systems [11]. In the field of biology, adaptation has been defined as the process of subsequent changes by which a living organism or a community of organisms becomes better suited to its environment and increases its chances to survive [12]. Initially proposed for natural systems, this interpretation implies four suppositions: (i) adaptation is towards a goal, purpose, or situation, (ii) adaptation is not a one-time action, but a purposeful sequence of changes, (iii) adaptation is done by the subjects of the changes themselves, and (iv) adaptation is to be put into the context of interaction with the environment or a community of organisms. The same principles have been imposed on engineered systems [13]. However, while biological adaptation is based on evolving bio-physiological and cognitive mechanisms, there are no *ab ovo* granted or naturally evolving mechanisms in the case of engineered systems [14]. Many experts believe that a deeper theoretical understanding of the phenomenon of system adaptation will ultimately lead to the opportunity of developing autonomous systems and adjustable autonomy.

The rest of this extended editorial is organized as follows. Section 2 summarizes the types and forms of system control and adaptation, Section 3 introduces the scientific, engineering, and computational fundamentals and issues of adaptation of first-generation cyber-physical systems (1G-CPSs). Section 4 discusses the phenomenon of self-adaptation of second-generation cyber-physical systems (2G-CPSs) and its fundamental issues. Section 5 offers a (non-exhaustive) landscape of the concerns related to next-generation cyber-physical systems (NG-CPSs). In addition, it discusses the milestone developments, and elaborates on some open questions. Section 6 presents the short synopses of the papers contributed to this special issue. Section 7 reflects on the major findings, what we apparently miss, and may consider as opportunities for future research.

2. A brief overview of the types and forms of adaptation of systems

Natural evolution and selection of living organisms is a long term and strongly conditioned process. The natural adaptation concerns many generations and favours to beings having a higher chance of survival and a wide variation of heritable characteristics. Obviously, engineered systems cannot exhibit such intricate mechanisms of progression. This is why systems science thinks differently about adaptation of such kind of systems. Nevertheless, it assumes the potential and resources of adaptive systems to change as well as the influence of the environment on the manifestation of changes. A birds-eye-view image of the perspectives of system adaptation is shown in **Figure 2**. In general, four sources of the need for adaptation are identified: (i) it is problematic to foreseen all requirements due to broadening and complexification of using such systems in the society, (ii) it is difficult to predefine all system operation and interaction modes due to growing uncertainties concerning applications and stakeholders, (iii) as a consequence of unpredictable incidental effects and changes in the environment, it is difficult to achieve overall resilience in the design phase, and (iv) owing to the emerging technological and servicing affordances, it is often possible to achieve better performance than that the systems have been programmed for. System adaptation can be relative to (i) a generally defined goal, (ii) a specifically defined goal, (iii) a partially defined goal, or (iv) a non-defined goal of operation/servicing. Considering these, adaptation is a means to (i) serve optimally for a purpose, (ii) maximize the fulfilment of operational/servicing goals, (iii) achieve the best relation with the embedding environment, and (iv) provide optimal interaction with other systems. In other words, it is about how something fits into something else and what efforts does it make towards an overall optimum performance in runtime. The action of adaptation may happen within a short operation period or over the entire lifecycle of engineered

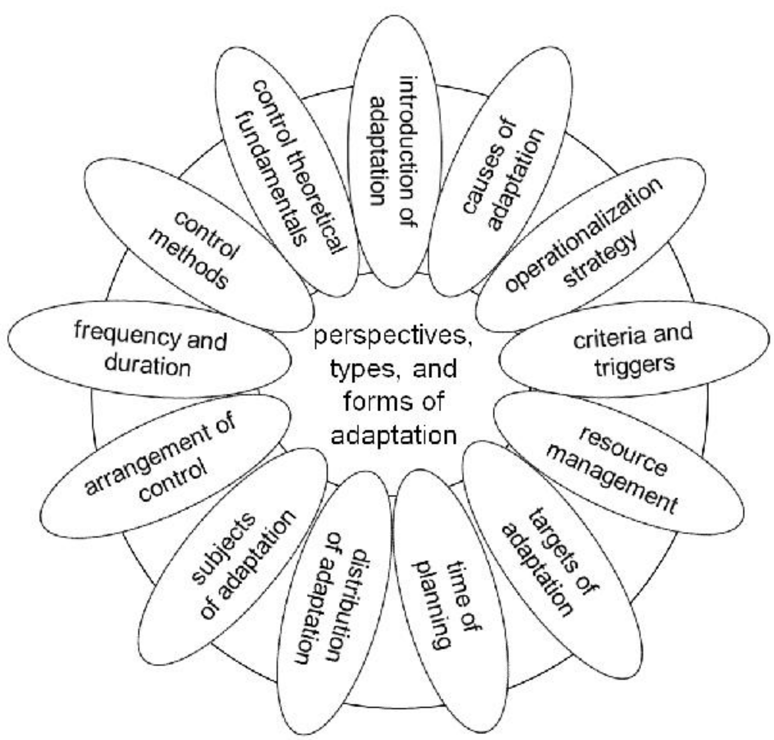

Figure 2: Perspectives of system adaptation

systems [15].

The above similarities and differences triggered the interest towards a universal theory of adaptation of systems, but that is still a work in progress. From a control theoretical perspective the literature discusses (i) traditional control-based adaptation (PID-like or state-representation driven), (ii) advanced control-based adaptation (model-predictive, optimization-based, and stochastic), (iii) knowledge-based control (rule-based, fuzzy, heuristic, and analogical), and reasoning-based control (data-driven, learning-based, abductive, prognostic, twin-based) mechanisms [16]. It must be emphasized that these kinds of adaptation apply to complex software systems, rather than to resource-heterogeneous cyber-physical systems. In line with the current layering of information technological systems, the categories of (i) infrastructural (hardware and software) resource adaptation, (ii) reusable middleware adaptation, and domain specific application software adaptation are imposed [17]. In general, the criteria (trigger) for execution of adaptation may be (i) goal-related, (ii) task-orientated, or (iii) performance-based. Based on the operationalization of the adaptation agency, (i) reactive (after change event), (ii) active (concurrent with change event), and (iii) proactive (before change event) control strategies can be distinguished. Feedback-based control supports reactive strategies, whereas feed-forward control is usually active. Combinations of feedback and feed-forward control can detect disturbances and adjust the inputs before the disturbance affects the system outputs. Consequently, this combination implements a proactive strategy and can be used as a proactive control mechanism.

Adaptation is usually not a single action of change, but a logically/functionally related linear sequence or other pattern of change actions. Therefore, it needs logical and procedural planning in the time dimension. In this dimension, various occurrences of adaptations a have been identified. For instance, based on the occurrence frequency of adaptation, (i) consecutive (repetitive), and (ii) incidental (one-time) forms of adaptation are distinguished. Based on the duration of adaptation, periodic (repeated in fixed intervals) or permanent (lasting over a relatively long period of operation or the whole lifecycle of a system) are differentiated. In terms of the introduction of the changes, adaptation can be made in idle-time and/or runtime. In addition, adaptation can be (i) externally initiated (based on intervention, or providing rules by an external controller or supervisor) or (ii) internally initiated (based on observed deviation from intended goal, state, performance, and output, or change of input data). In terms of intentionality (the reason of initiating a specific event), (i) indispensable, (ii) planned, or (iii) self-decided adaptation are distinguished. Adaptations are planned in the (i) design-time, (ii) runtime, or (iii) in both.

With regard to the change of the system's constituents (components), (i) constant resource-based, and (ii) variable resource-based adaptations are implemented. From the perspective of organization of the changes (i) centralized and (ii) decentralized approaches are used. The target of adaptation can be (i) goal, (ii) functions, (iii) architecture, (iv) operation, (v) intellectualization, (vi) interactions, (vii) behaviour, and (viii) combined adaptation. Furthermore, (i) environment centred adaptation (for a proper interaction with a dynamic environment) and (ii) system centred adaptation (guaranteeing the dependability of the states/operations/services) are differentiated.

As discussed by Patikirikorala et al., the control may have single objectives or multiple objectives in the case of software systems, and (i) basic or (ii) composite control schemes are implemented depending on the complexity of the objectives [18]. Basic control schemes are such as (i) model-based fixed-gain control, (ii) model-based runtime dynamic-gain control, (iii) linear quadratic regulator, and (iv) model-based predictive horizon control. Composite control schemes are, for example, (i) cascaded (nested) control: (ii) rules-based gain scheduling, (iii) algorithms reconfiguring control, (iv) top-down distributed (hierarchical) control, (v) decentralized independent control, and (vi) combined event- and time-based (dynamic) controls [19]. The discussed control strategies are usually put under the conceptual umbrella of internal control, which means some form of intertwining application functionality (logic) and control functionality (logic). However, the literature is void concerning (runtime) hardware and cyberware adaptation issues that are especially important in the case of transforming cyber-physical systems [20].

The abovementioned strategies are typically model-based. Models either are predefined in the design-time, or are generated in runtime. Though current model engineering makes the creation of dynamic models possible, the range of adaptation is restricted to self-regulation and self-tuning in the case of control-oriented models. When a fault or an unclear change in the environmental circumstances happens, human intervention is expected. Often, this is referred to as mitigating adaptation. In the case of mitigating adaptation, designers define (i) the specific objectives to achieve, (ii) the boundary conditions of operation, (iii) the conditions of adaptation, (iv) the mechanisms of adaptation, and (v) the appropriateness criteria of adaptation [21]. Another aspect of adaptation is its computational enabling, which can be (i) model-based, (ii) data-driven, awareness-based, and ontology-based enablement. Model-based adaptation strategies involve harmonization of various models such as (i) system models, (ii) control models, (iii) optimization models, (iv) environment models, (v) impact models, and (vi) meta-models.

Internally initiated adaptation is self-adaptation **-** a form of system operation, for which the goals and rules of adaptation are not provided by external controllers. Traditionally, self-adaptation of systems was defined as the abilities to make appropriate corrective actions based on the information about the actions, which will have the best enhancement impact on the system in runtime. Recently, it has been reinterpreted as the capability of (i) setting a new goal at runtime for system-level problem solving, (ii) determining the most efficient strategy, plan and execution of changes, and (iii) working according to this to reach the initially or runtime set goal [22]. This multifaceted capability assumes sufficient awareness, reasoning, learning, planning, and decision-making abilities and mechanisms. For many researchers, the core of designing for adaptation is system-level modelling that (i) defines the relationship with the operational environment, (ii) monitors the objectives and the state of a system, and (iii) configures adaptation mechanisms and strategies in the design-time of a system.

The above overview of the major adaptation aspects intended to shed light on the complexity of the phenomenon of system adaptation. In addition, it attempted to evidence that the landscape of research and development activities towards the realization of system adaptation is a very broad and varied. Two tangible reasons of this are (i) the current wide spectrum of system manifestations, and (ii) the dynamic appearance of new generations of engineered systems.

3. Systems science, engineering, and computational issues of adaptation of first-generation cyber-physical systems

The family of 1G-CPSs include control-intensive plant-type systems for which the primary objective and the logic of operation do not change during the life span. Coordinated control loops are essential to build this kind of adaptive systems, which are actually results of functional enhancement by cyber-physical augmentation (i.e. supplementing physical systems with stand-alone or networked computational platforms) [23]. The interfaces between the physical transformation processes and information computation processes are sensors and effectors (or clusters of these). Another approach to realization of 1G-CPSs is complementing a digital network with physical objects (instruments, devices, robots, vehicles, etc.). This is a typical strategy of the Internet of Things (or Internet Everything) driven development efforts [24]. In view to the capabilities of rapidly progressing higher-level implementations, 1G-CPSs are regarded as low-end cyber-physical systems.

The functionality, architecture, and the logic of operation of the 1G-CPSs are defined in the design phase and they do not change throughout the life span of the system. In other words, this family of CPSs is supposed to adapt to known modes of changes. This assumption makes it possible for the designers to use model-based engineering extensively in their development. Usually, 1G-CPSs systems are equipped with conventional control mechanisms and can regulate the parameters of operation to a known degree through the system model and control model. The end-user can adjust the predefined adaptive control algorithms with some preselected parameters. According to the latest reviews of industrially relevant control strategies, the ones most used in practice are proportional-integral-derivative (PID) control and model-based predictive control (MPC). Such solutions are acceptable for many applications with predictable circumstances and working conditions. However, 1G-CPSs may become unreliable or inefficient in situations that were not predicted in the design phase and they are unable to adapt to.

The self-control implemented by 1G-CPSs may appear in multiple forms such as self-regulation, self-healing, self-resilience or self-tuning. Though these, like self-adaptiveness, are realized typically in a top-down manner, the literature considers these as a limited sub-set of the capabilities that make CPSs self-adaptive [25]. The abovementioned capabilities are differentiated also from self-organization that, with a view to emergent functionalities and to decentralization of their control, works according to a bottom-up manner. We see self-organization as the mutual adaptation and co-evolution of the initially autonomous components of systems, namely, the agents. In the view of the related literature, self-organization is the spontaneous process through which systems emerge and evolve, becoming ever more complex, more adaptive, and more synergetic [26].

Internally initiated control intertwines the logic of application functions and the logic of adaptation functions. This approach is based on programming language features, such as conditional expressions, parametrization, and exceptions, in software systems [27]. The sensors, effectors, and adaptation processes are mixed with the application code. This often leads to poor scalability and maintainability, and the system is costly to test and maintain/evolve. Using external adaptation engine (or adaptation manager), external approaches of self-adaptive software system try to avoid these limitations by offering sophisticated adaptation processes. In addition, it offers reusability (customization and configuration for different systems) of the adaptation engine, or processes tailored for various applications. An adaptation engine can implement both closed adaptation (using defined type/number of adaptive actions) and open adaptation (allowing new software arrangements and behaviors during runtime) [28].

Over the years, a dual-aspect solution emerged in the form of the monitor-analyse-plan-execute (MAPE) approach [29]. This conceptual abstraction and generalization of the external feedback loop-based type of control realizes an adaptation logic that is significant for several reasons. For example, it: (i) allows to separate the concerns of fulfilment of the system functionality and the management of self-control, (ii) facilitates model-based adaptation control, even self-adaptation, by decomposing the control loop into four specific phases, (iii) supports the extension of control information with knowledge stored in a knowledge repository, and (iv) creates a methodological bridge between self-control of 1G-CPSs and self-adaptation of 2G-CPSs. As discussed by Miller, the monitor, analyse, plan, and execute functions must share knowledge. Hence, this modelling approach is often referred to as MAPE-K [30]. Iglesias and Weyns proposed to use formally specified MAPE-K templates that encode design expertise for a family of self-adaptive systems. These includes templates for behavioural specification and modelling the different components of a MAPE-K feedback loop, as well as property specification templates that support verification of the correctness of the adaptation behaviours [31].

However, the MAPE-K approach is limited in terms of runtime variability, including variable structure and functionality systems. Furthermore, the issues of verification of adaptation plans before execution and validation of the results of the completed self-adaptation in context, and the issue of resource generation and management during the lifecycle of the controlled system were not addressed specifically. Tavčar and Horváth argued that these functions should be included in the self-adaptation loop and proposed managing it in four logical steps: (i) planning self-adaptation, (ii) verification before self-adaptation, (iii) operationalization of self-adaptation, and (iv) validation of self-adaptation, which extends MAPE-K into MAPVEV-K [32].

Having analysed the current research on methods and techniques for designing and engineering of adaptive software systems, Hidaka et al. argued that effective development of self-adaptive systems could be achieved through the reuse and adaptation of existing models such as MAPE-K loops [33]. The survey completed by Muccini et al. (2016) explored that the application layer and the middleware layer (rather than the communication, service or cloud layer) are the typical levels of system adaptation and that MAPE-RL (where, RL stands for 'reason and learn'), agents, and self-organization are the dominant adaptation mechanisms [34]. Among others, these functions are seen as crucial elements for self-supervised self-adaptation of cyber-physical systems.

Chandra et al. (2016) analysed and compared architecture frameworks currently proposed for designing self-adaptive systems. The analysis included (i) the observe-decide-act (ODA), (ii) the MAPE-K, (iii) the autonomic computing paradigm (ACP), and (iv) the observer/controller architecture (OCA) frameworks, which are rooted in organic computing research and are intended for different types of distributed systems, such as swarms, systems-of-systems, crowd computing arrangements, computing entity populations, and multi-agent systems [35].) As a typical example of demand-enabled system adaptation, Hummaida et al. (2016) presented a resource management strategy for clouds (allocation of a shared pool of configurable computing resources) [36]. As a concluding remark, we may claim that, in spite of the efforts, only useful pieces of an incomplete theory of system-level self-control of real life systems are available and those do not include the agency of (intuitive and creative) heuristics, or metaheuristics, that helps solve a wide variety of application problems.

4. Systems science, engineering, and computational fundamentals of self-adaptation of second-generation cyber-physical systems

The above discussion is based on five main premises: (i) first-generation CPSs are designed for known modes of changes and to implement self-tuning of their operation, whereas (ii) second-generation CPSs are designed to handle partially or completely unknown modes of changes and are equipped with the capability of self-adaptation of operation, (iii) while human stakeholders play an important role in assurance of system operation and performance of 1G-CPSs, there is a move towards partial automation of adaptation in the case of 2G-CPSs, (iv) the application functions and adaptation functions are purposefully separated in self-adaptive systems, while application logic and adaptation logic are largely mixed in adaptive systems, and (v) research in self-adaptive systems distinguishes between internal and external adaptation mechanisms. These assumptions lend themselves not only to the distinction of various system generations, but also to a natural demarcation of two major realms of system control: internal and external.

In principle, the goal of self-adaptation can be either adapting the environment to maintain the targeted performance of the system, or adapting the system operations according to the environmental changes, or both in combination. Conventionally, adaptive systems are pre-programmed to realize the adaptation logic by means of closed feedback loops, while self-adaptive systems are pre-programmed to find a possible, or the relative best adaptation logic by sophisticated computational mechanisms such as learning, reasoning, and abstracting [37]. In the case of self-adaptation, on the one hand, the designers define (i) the overall objectives to achieve, (ii) the overall operational processes, (iii) the possible resources of adaptation, and (iv) the scenario of realizing possible adaptations. On the other hand, the system decides on: (v) the necessity of adaptation, (vi) the resources to be used for adaptation, (vii) the concrete procedures of adaptation, and (viii) the execution of adaptation. In the case of self-adaptive systems, it is possible to separate the parts of the system that deal with application concerns (i.e. the goals for which the system is built) from the parts that deal with the self-adaptation concerns. Though this separation is useful for system engineering and computational reasons, the application-oriented subsystem and the control-orientated subsystem are supposed to operate in a synergetic functional coupling. Approaching from a computational perspective, 2G-CPSs may exploit (i) search-based techniques, (ii) logical and uncertain reasoning techniques, and (iii) machine learning techniques to deal with unanticipated requests and uncertainties, and preparation for change. By doing so, they implement various forms of autonomic computing [38].

Self-adaptation of (heterogeneous) CPSs is a more complicated task than that of self-adaptation of software systems. One obvious reason is that the control software should adapt not only itself, also the hardware and cyberware constituents. Another reason is that that planning of the adaptation needs comprehensive context management. Many researchers see self-adaptation as a risk mitigation strategy with regard to the uncertainties caused by runtime changes on the application-oriented subsystem. There is still a knowledge gap with regard to handling real-time changes and constraints accounting for context variability. Rodrigues et al. combined off-line requirements and model checking with on-line data collection and assessment to guarantee the system's goals by fine-tuning the adaptation policies towards optimization of quality attributes [39]. Engelenburg et al. provided a method to identify what elements of the environment are relevant context, which involves three steps: (i) getting insight into context, ii) determining what components are needed to sense and adapt to context, and iii) determining the rules for how the system should adapt in different situations [40]. Since not only static context but also dynamic context is to be managed in specific applications, Don et al. proposed an event-driven awareness mechanism [41]. Another source of complication is that, beyond the change of the operational parameters, self-adaptation extends to changing elements of the system functionality (operations) and the system architecture (configuration and relations of components) in the runtime. Towards the orchestration of these, Braberman et al. proposed a reference architecture that allows for coordinated yet transparent and independent adaptation of system configuration and behaviour [42]. Cansado et al. proposed a formal framework that unifies behavioural adaptation and structural reconfiguration of components and showed the advantages in the context of reconfiguration of a client/server system in which the server has been replaced [43].

It is well known by the software engineering community that the term 'architecture' refers to the conceptual model that defines the behaviour, structure, and characteristics of a software system that fulfils the given requirements. In software engineering, architecture is a bridge between requirements and computational codes [44]. It is conceived also as a formal description of the integrated, distributed, or hybrid arrangement and interconnection of the functional components. Involving qualitative judgment, architectural adaptation is a multi-faceted issue and implies modification on various levels [45]. Understanding its guiding principles and possible forms is a central topic for research in self-adaptive systems. Villegas et al. posited that, besides the regular functional components of the system, the designed architectures must include components that enable self-awareness capabilities, such as monitoring and analyzing its own current state, as well planning and executing self-adaptation actions [46]. There are different possibilities for runtime architectural self-adaptation of composable and compositional systems. Kramer and Magee outlined a three-layer architectural reference model that provides the required level of abstraction and generality for self-management of composable architectures [47].

Compositional adaptation exchanges algorithmic or structural system components with others that improve the fit of the software to the state its current environment. Phan and Lee proposed a compositional multi-modal approach to model, analyse, and design adaptive CPS on a distributed architecture that facilitates adaptiveness, efficient use of resources, and incremental integration [48]. Compositional adaptation is powerful, but its use without appropriate tools to automatically generate and verify code may negatively affect system integrity and security [49]. Compositional self-adaptation control systems should consider both static aspects (such as stability and availability) and dynamic properties (such as functional interconnections and transient change of variables). The dependable emergent ensembles of components (DEECo) framework, presented by Masrur et al., (i) allows modelling large-scale dynamic systems by a set of interacting components, (ii) provides mechanisms to describe transitory interactions between components, and (iii) supports reasoning about timing behaviour of the interacting components [50]. The motivation came from the hypothesis that components may automatically configure their interactions within self-managed software architectures in a way that is compatible with the overall architectural specification and can achieve the goals of the system. Another dimension of self-adaptation is self-adaptation of system of systems that is still in a premature stage of understanding and implementation [51].

Using models as the basis of self-adaptation is both a theoretical issue and a methodological one. The latter is concerned with the dynamic generation and adaptation of system and control models. Runtime models are based on abstractions of the system, while the goals serve as a driver and enabler for semi-automatic reasoning about system adaptations during operation [52]. Many researchers emphasized the role of software models at runtime (M@RT) as an extension of the adaptation control techniques to runtime contexts [53]. For instance, a key challenge for self-adaptive software systems is assurance. Some of the assurance tasks need to be performed at runtime. Towards this end, Cheng et al. argued that research into the use of M@RT is fundamental to the development of runtime assurance techniques and presented what information may be captured by M@RT for the purpose of assurance [54]. Bennaceur et al. developed a four-layer partially causal conceptual M@RT reference model to provide a framework for the core concepts and to situate the computational mechanisms [55].

Klös et al. extended the MAPE-K feedback loop architecture by imposing requirements and a structure on the knowledge base and introducing a meta-adaptation layer. This enables (i) learning new adaptation rules based on executable runtime models, (ii) continuous evaluation of the accuracy of previous adaptations, and (iii) verification of the correctness of the adaptation logic in the current system context [56]. Hadj-Kacem constructed a formal model using a coloured Petri-net for an adaptive system to be trusted after adaptation. This model has sufficient abstraction of details, but still deal with the core of the protocol. This makes the model simpler and the analysis easier due to restricted state space size [57]. Also of theoretical significance is the three-phase approach for modeling and developing dynamically adaptive systems based on the combination of the runtime models technique and the aspect-oriented software development paradigm proposed by Loukil et al. The architecture of the software is specified in the first phase, the executable code is automatically generated in the second phase, and the running system is reconfigured and supervised in the third phase [58].

It is an intensifying trend to use artificial narrow intelligence techniques (in particular deep learning and machine learning) and fully-fledged digital twins in various runtime activities of system self-adaptation and dependable automation [59]. This on-going intellectualization concerns both the tasks related to solving application problems and the tasks related to self-adaptation and self-supervision related [60]. In both respects, both theoretical and practical issues are addressed. Integration of awareness building, machine learning, and ampliative reasoning mechanisms into software makes them capable to behave smartly and to handle not anticipated situations [61]. The latter efforts are justified by the growing need to autonomously detect and manage unanticipated or unknown situations and to plan the adaptation during runtime properly [62]. The inclusion of learning mechanisms in self-adaptive systems improves not only their flexibility, but also their reusability [63]. However, current computational learning allows self-adaptive CPSs to change their operation and/or configuration only up to specific limits or inside a goal-defined operational envelope. Furthermore, not only constrains, but also the usable resources are defined in the design phase [64]. Nevertheless, these technological augmentations of 2G-CPSs (i) transfer discrete functional and architectural adaptation approaches into a continuous (perpetual) self-adaptation, (ii) reduce reliance on human supervisors and increase the level of automation, and (iii) enhance the technological readiness for resource sensitive evolutionary self-adaptation. Three major issues are (i) the purpose-driven selective learning, (ii) the trustworthiness of the learnt data- and rule-driven models, and (iii) the scalability of the proposed solutions. Therefore, many researchers encouraged to gain experiences with industrial systems and applications [65].

5. Towards Next-Generation Cyber-Physical Systems

We made a (non-exhaustive) literature study with the intention to get insights in: (i) the trends of current research, (ii) the probable future developments, and (iii) the recognizable research/knowledge gaps in the field of next-generation cyber-physical systems (NG-CPSs). We focused on those seminal publications that presented front-end and road-paving research and development results, critical and conclusive overviews, or evidenced personal viewpoints. An important observation was that only a small portion of the studied journal articles and conference papers looked ahead to future CPSs, though the number of the related publications progressively increased in the last decade. Based on the selected publications, we attempted to sketch up a landscape of the major concerns of research and development towards NG-CPSs. Towards this end, we imposed an initial classification of the concerns according to what they were related to. The four categories of concerns were: (i) (holistic) system concerns (Σ), (ii) software concerns (S), (iii) hardware concerns (H), and cyber concerns (C). We divided the system concerns into two sub-categories: (i) generic system concerns (Σ_1), and (ii) system supervision concerns (Σ_2). The obtained landscape is shown in **Figure 3**. Due to the abundance of the associated concerns, we allocated the software concerns to three sub-categories: (i) system modelling concerns (S_1), (ii) software

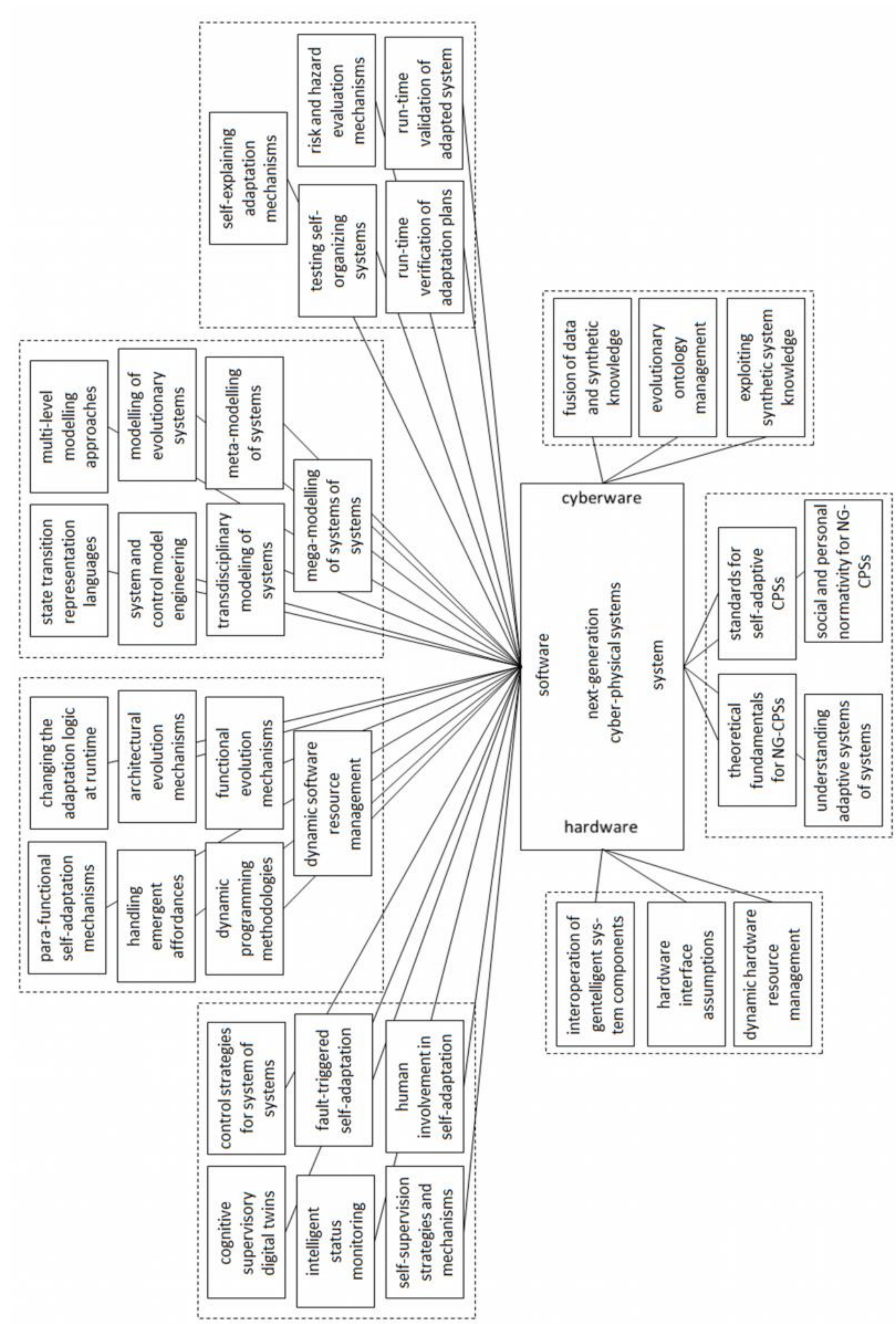

Figure 3: Major concerns of next-generation cyber-physical systems

self-evolution concerns (S_2), and (iii) software dependability concerns (S_3). The sub-categories were decomposed further into concern domains in the following way:

Σ_1 Generic system concerns:
(i) theoretical fundamentals for NG-CPSs [66] [67] [68], (ii) understanding adaptive systems of systems [69] [70], (iii) standards for self-adaptive CPSs [71] [72], and (iv) social and personal normativity for NG-CPSs [73] [74] [75].

Σ_2 System supervision concerns:
(i) self-supervision strategies, frameworks, and mechanisms [76] [77], (ii) human involvement in self-adaptation [78] [79] [80], (iii) intelligent status monitoring [81] [82], (iv) fault-triggered self-adaptation [83] [84], (v) cognitive supervisory digital twins [85] [86], and (vi) control strategies for system of systems [87] [88].

S_1 System modelling concerns:
(i) state transition representation languages [89] [90] [91], (ii) multi-level modelling approaches [92] [93] [94], (iii) modelling of evolutionary systems [95] [96] [97], (iv) system and control model engineering and optimization [98] [99] [100], (v) transdisciplinary modelling of systems [101] [102], (vi) meta-modelling of systems [103] [104] [105] [106], and (vii) mega-modelling of systems of systems [107] [108] [109] [110].

S_2 Software self-evolution concerns:
(i) dynamic programming methodologies [111] [112], (ii) functional evolution mechanisms [113] [114], (iii) architectural evolution mechanisms [115] [116], (iv) handling emergent affordances [117] [118], (v) changing the adaptation logic at runtime [119] [120] [121], (vi) para-functional self-adaptation mechanisms [122] [123] [124], and (vii) software resource management [125] [126] [127].

S_3 Software dependability concerns:
(i) run-time verification of self-adaptation plans [128] [129] [130], (ii) run-time validation of self-adapted systems [131] [132] [133], (iii) testing self-organizing systems [134] [135], (iv) risk and hazard evaluation mechanisms [136] [137] [138], (v) self-explaining adaptation mechanisms [139] [140] [141] [142].

H_1 Hardware management concerns:
(i) dynamic hardware resource management [143] [144], (ii) hardware interface assumptions [145] [146] [147], and (iii) interoperation of gentelligent system components [148] [149] [150].

C_1 Cyberware management concerns:
(i) fusion of data and synthetic knowledge [151] [152] [153] [154], (ii) evolutionary ontology management [155][156][157]), and (iii) exploiting synthetic system knowledge [158] [159] [160].

The references included above are only examples of typical publications orientated to the particular concern domains. It must be fairly mentioned that the landscape shown in Figure 3 is probably incomplete and subjective. The reasons of incompleteness are multiple. For instance, our literature analysis could cover only a limited set of the abundant amount of relevant publications. Due to the obvious space limitations, even less could be included in the above overview. It was also a technical issue that several studied papers addressed multiple concerns or intended to contribute to multiple concern domains. We made an attempt to sort them in the most relevant category. In addition to our personal interpretations, views, and judgments, this also contributed to the subjective nature of the landscape. We have not done any further research yet to validate its comprehensiveness and appropriateness, and to consolidate it in a broader application context.

Notwithstanding these issues, the presented landscape is deemed a starting point for further discussions and analyses. It can be observed that the number of concerns related to hardware, software, and cyberware categories are largely different. The overwhelming majority of them are related to software that plays multiple roles (such as integrator, driver, processor, mechanism, manager, and utility) in current and future CPSs. The landscape also reflects certain trends, which are summarized in the Conclusions section of this extended editorial. In the next section, we use it to position the contributed papers in the most relevant concern domain.

6. Short Synopses of the Contributed Papers

This special issue is based on an open Call for papers initially presented on the journal's website. The Call attracted the attention of many potential authors. The selection of the submitted manuscripts for the peer review process and, after that, the best ones for publication was not a simple task. There were excellently written papers addressing somewhat conventional topics, and there were less well-elaborated papers addressing novel and essential topics. Concerning the whole of the submitted manuscripts, it is fair to mention that there was only a weak thematic coherence among them. For the above reasons, less than half of the reviewer papers could be considered for publication. It means that, in the end, six original contributions have been included in this special issue. Based on their actual objectives and contributions, these papers can be arranged into three general groups: (i) road-mapping for systems science and engineering (P1 and P2), (ii) methodological approaches to designing self-adaptive systems (P3 and P4), and (iii) enablers for realization of self-adaptive systems (P5 and P6). Below we briefly introduce these high quality papers.

The first paper, submitted by Danny Weyns, Jesper Andersson, Mauro Caporuscio, Francesco Flammini, Andreas Kerren, and Welf Lowe, proposes "*A Research Agenda for Smarter Cyber-Physical Systems*". This paper contributes to the conceptual framing and understanding of several concerns domains in the sub-categories of software self-evolution and software dependability concerns, as well as in the sub-categories of generic system concerns and system supervision concerns, and provides a broad and deep theoretical underpinning for next-generation cyber-physical systems. The work complements the existing perspectives on system smartness by taking a more holistic perspective that integrates systems operation with the processes to engineer them. The authors argue that both systems and the way they are engineered must become smarter. Systems and engineering processes must adapt themselves, and evolve based on stakeholders' input and from experience through a perpetual process that continuously improves their capabilities and utility to deal with environmental and operational uncertainties and amounts of data they face throughout their lifetime. The authors highlight key engineering areas (cyber-physical systems, runtime self-adaptation, data-driven technologies, and visual analytic reasoning), and outline some major challenges in each of them. They explain the synergies between these key areas. The second part of the paper presents the authors' proposal for a comprehensive research agenda. This addresses three themes: (i) assurances for unknowns (in the case of decentralised and smarter cyber-physical-systems that operate under uncertainty), (ii) self-explainability of autonomous decisions (concerning a lifelong self-learning and self-explainable cyber-physical systems), and (iii) smarter ecosystems for perpetual adaptation and evolution (including a unified modelling approach and self-governance for smarter cyber-physical systems). Exhibiting a high-level of autonomy, smarter cyber-physical ecosystems require reflective capabilities based on which they data about their utility and adjust according to their shifting operational goals. Recognizing the necessity of convergence, the research agenda calls for a multi-year concerted effort of research teams active in the different key areas of studying and developing novel solutions for trustworthy and sustainable cyber-physical systems.

The second paper, entitled "*Designing Runtime Evolution for Dependable and Resilient Cyber-Physical Systems Using Digital Twins*", presents the work and the results of Luis F. Rivera, Miguel Jimenez, Gabriel Tamura, Norha M. Villegas, and Hausi A. Muller. The main contribution of this paper belongs to the concern domain of cognitive supervisory digital twins in the system supervision concerns sub-category, but it also adds to the sub-category of software self-evolution concerns, more specifically, to concern domain of functional/architectural evolution mechanisms, and to the self-supervision strategies, frameworks, and mechanisms concern domain. The authors emphasize that designing of smart cyber-physical systems must address not only dependable autonomy, but also operational resiliency. Their goal was to implement reliable self-adaptation and self-evolution mechanisms and to include them in the design of SCPS. Their results are threefold: (i) a reference architecture for designing dependable and resilient SCPS that integrates concepts from the fields of digital twins, adaptive controls, and autonomic computing, (ii) a model identification mechanism to guide self-evolution, evolutionary optimization, and dynamic simulation, and (iii) a gradient descent-based adjustment mechanism for self-adaptation to achieve operational resiliency. In addition to the model identification and the adjustment mechanisms, a featured contribution of this work is a so-called 'reference architecture' for designing digital twin-based autonomic control for dependable and resilient cyber-physical systems. The authors implemented prototypes and showed their viability using real data from a case study in the domain of intelligent transportation systems. The proposed execution adjustment mechanism finds appropriate control parameters so that the controller can enforce the control objectives in the CPS.

The next paper was submitted by Camille Salinesi, Asmaa Achtaich, Nissrine Souissi, Raul Mazo, Ounsa Roudies, and Angela Villota, under the title: "*State-Constraint Transition: A Language for the Formal Specification of Self-Adaptive Requirements*". It offers a methodological approach to designing self-adaptive systems. The main contribution covers the concern domain of dynamic programming methodologies in the sub-category of software self-evolution concerns, and the concern domain of state-transition representation languages in the sub-category of system modelling concerns. The observation of the authors was that existing formal languages focus on the fulfilment of the users' requirements by the designed system in the current context. However, they hardly consider runtime dynamically emerging requirements and context-sensitive requirements. Therefore, the authors introduced a state-constraint transition (SCT) modelling language to provide a solution to the problem of specifying dynamic requirements. An essential feature of this solution is the concept of configuration states, in which requirements are translated into constraints. The paper explains both the syntax and semantics of SCT and provides examples for reconfiguration scenarios. The authors realized the SCT requirement specification process relying on the finite-state machines (FSM) approach that provided the necessary computational power and expressiveness for constraint programming. Their preliminary evaluation explored both the benefits (expressiveness, scalability, domain independence) and the limitations (temporal constraints, scheduled reconfigurations, and validation of constraints) of SCT.

The fourth paper, entitled "*One-of-a-Kind Production in Cyber-Physical Production Systems Considering Machine Failures*", presents the results of Guido Vinci Carlavan and Daniel Alejandro Rossit. Though the topic of the paper is broader than a software concern, its scientific contribution can be related to the concern domain of 'advanced control strategies for system of systems' in the sub-category of 'system supervision concerns'. Within customized production, the one-of-a-kind production (OKP) paradigm is the extreme case for production control and scheduling. Cyber-physical systems used in Industry 4.0 are supposed to facilitate the management of information related to each singular product, as well as the resolution of conflicts that may arise in processes with a very high variability. That is the reason why the authors studied the implementation of the constant work-in-progress (CONWIP) control logic in OKP systems from the perspective of productive job shop configurations in Industry 4.0 environments. The CONWIP control logic was able to handle the challenging Industry 4.0 problem in an efficient manner, with a relatively low need of investment in CPS related equipment. However, they also found that the performance is sensitive to the stress of the scenario, i.e. the arrival rate of jobs - an issue closely related to the used dispatching rules. The general conclusion of the authors was that dispatching rules associated with due dates tend to improve the overall performance of the system, and the first-in, first-out (FIFO) rule has the worst performance in all experiments. Essential feature of their work is that simulation-based experimental studies were developed and their results have been compared systematically. As design concerns of the next-generation cyber-physical systems, Carlavan and Rossit elaborated upon on intelligent status monitoring, fault-triggered self-adaptation, and system and control model engineering.

The title of the fifth paper is: "*Remote Runtime Failure Detection and Recovery Control for Quadcopters*". The authors, Sajad Shahsavari, Mohammed Rabah, Eero Immonen, Mohammad-Hashem Haghbayan, and Juha Plosilab identified managing failures as a basic enabler for realization of dependable self-adaptive systems, such as quadcopter drones. This work contributes to the concern domains of fault-triggered self-adaptation and cognitive supervisory digital twins in the system supervision concerns sub-category. The authors implemented a distributed control system that includes: (i) a local on-board PID-based control sub-system responsible for manoeuvring the drone in all conditions, (ii) a remote control sub-system responsible for detecting normal or failure states of the drone and communicating with the drone in real time, and (iii) a digital twin co-execution sub-system responsible for a real-time two-way data exchange between the above sub-systems. The measured RPM values of the quadcopter's motors are transmitted to a remote computer, which hosts the failure detection and recovery software platform. The control concept was implemented using the Simulink tool. The authors propose a modification of the Quad-Sim simulation model to represent motor failure situations. In addition, they offer a fast fault detection and recovery technique capable to work at run-time, and a two-way data-stream management facility. The experimental results obtained by using the MCX co-execution platform show the applicability and efficiency of the proposed approach in detecting failures and safely landing drones after failure detection.

Included as last in this special issue, the work of Amal Ahmed Anda and Daniel Amyot mainly addresses the concern domain of 'system and control model engineering and optimization' in the sub-category of software 'system modelling concerns'. Nevertheless, their paper, entitled, "*Goal and Feature Model Optimization for the Design and Self-Adaptation of Socio-Cyber-Physical Systems*", also contributes to the concern domains of run-time validation of adapted system, functional evolution mechanisms, intelligent status monitoring, and human involvement in self-adaptation. The presented optimization method provides design-time and runtime solutions for goal-based self-adaptation of socio-cyber-physical systems (SCPSs), while supporting the validation of their design models. The goal satisfaction is supported by a simultaneous monitoring the system's environment and operational qualities, while constraints enforcing correctness are specified in the feature model. The arithmetic functions are generated automatically from goal and feature models. The generated goal-feature model is solved by an optimization tool, which calculates optimal adaptation solutions for foreseen common situations at design-time. In addition, runtime optimization is used also by the system in order to adapt to situations unanticipated in the design-time. To assess how well the proposed approach could be used to manage selection among alternatives while solving emergent conflicts, it was applied to a smart home management system. The optimized performance of the system was assessed through the fulfilment of time, total programing time, memory usage, and program memory usage goals/constraints. The approach proposed by Anda and Amyot facilitates iterative processes, reduces design errors, and increases system reliability.

7. Some Conclusions about What We Miss …

Though significant progress has been achieved both in the research and development and in the theories and practices, there are still many open issues and unanswered questions. As our above analysis showed, this can be attributed to the extreme rapid shifts in the research phenomena and the academic interests. Below we attempt to pinpoint the open issues that are expedient to get resolved on a short notice:

1. Second-generation cyber-physical systems are based on a balanced utilization of hardware, software, and cyberware resources. Nevertheless, most of the research efforts focus on software challenges and issues. This can be explained by the dominance of research in information processing and smart reasoning systems, but self-adaptation of transformative (such as production, robotic, medical, and transportation) 2G-CPSs require sophisticated hardware and cyberware resource management potentials. Publications on their integral theoretical fundamentals and methodological approaches are scarce in the current literature.

2. As explained above, a functional motivation for self-adaptation is enabling systems to handle operational uncertainties that were difficult to foresee before deployment. At the same time, a non-functional motivation for self-adaptation is freeing system operators and administrators from the need of continuously monitoring and adjusting systems operating round-the-clock. Self-adaptation may introduce various levels of transformative operations such as (i) self-tuning, (ii) self-adaptation, (iii) self-conversion, and (iv) self-reproduction. In all cases, self-adaptive systems are inherently

nonlinear, as they possess parameters that are functions of their states and conditions. Thus, self-adaptive systems are simply a special class of nonlinear systems that either measure their own performance, operating environment, and operating conditions of the components and adapt their dynamics or those of their operating environments to ensure that measured performance is close to targeted performance or specifications.

3. Facilitating systems' self-evolution and reaching autonomy seem to be two dominant tracks of developing next-generation cyber-physical systems. Adaptation turns to evolution when new resources are provided for a system runtime. Functional evolution and evolutionary adaptation assume extending the system resources (hardware, software, and cyberware) in runtime and adapting the system objectives, operation, performance, and relationships accordingly the obtained affordances. Autonomous adaptation has been interpreted as self-adaptation without any form of human interaction. In this case, the system itself is responsible for self-supervising the both the planning and the execution of adaptation, considering all risk factors and implications. The current literature offer neither robust underpinning theories, nor structured methodologies for evolutionary and autonomous self-adaptation.

4. Artificial narrow intelligent techniques (in particular, various mechanisms of computational learning) are increasingly used in self-control and self-adaptation of second-generation cyber-physical systems. Artificial neural network-based and other AI-based controller mechanisms extend the self-adaptation potential with additional functionality, but are not able to adapt to frequent requirements changes at runtime or to scale up to complex real life situations. Sections 2- 5 hinted at some open design issues that cannot be resolved since the knowledge they need is partly or entirely not available. To explore the knowledge gaps and eliminate the knowledge deficiencies, first the problems are to be correctly identified. Cognitive engineering will play an important role with regard to next generation systems.

5. Dynamic management of the operational and servicing goals of systems based on runtime emerging requirements is recognized as important topic for further studies, but dynamic development of goal models it is still in its infancy. The changes during the software lifecycle lead to software architecture erosion and make the management of software architecture evolution a complex task. Most existing computational approaches to architecture evolution enable evolution of early stage models only and fail to support the whole lifecycle of component-based software.

6. The fundamental mechanisms of automatic runtime (fine-)tuning of the adaptation logic to unanticipated conditions, runtime verification of adaptation plans, learning the impact of adaptation decisions on the goals of the system, and validation and testing the performance of self-adaptive systems after (multiple) adaptations are still concerns for research and development. These are especially relevant issues for networked times 2G-CPSs and mission critical systems.

7. A rapid shift can be observed in the literature from self-adaptive systems to self-supervised self-evolving systems, without providing complete solutions for the self-adaptation problem. The idea of layering was introduced in the design of self-adaptive software systems in order to separate the different types of concerns and to address various kinds of uncertainties. An interesting and important, but narrowly addressed research topic is functional emergence and utilization functional affordances in the case of NG-CPSs. Emergence may be a result of self-organisation, in particular in the case of multi-agent-based systems.

8. Designing CPSs requires an extensive collection of heterogeneous computational models, such as systems models, morphological models, physical models, structural models, hardware models, software model, information model, control models, reasoning models, and so forth, to enable deep semantic integration, simulation, and analysis. Models should interoperate and provide a sufficiently complete representation of the operation, structure, and behaviour of 2G-CPSs. In spite the efforts to introduce meta-models and mega-models, the currently used models (i) work in conceptually different engineering dimensions, (ii) are based on different abstractions, and (iii) involve different

representation formalisms. The methodology of coherent and consistent transdisciplinary and multi-dimensional system modelling and a cross-domain (hardware, software, and cyberware) representation formalisms need further attention in research. Formal criteria for structural and semantic consistency of modelling tools are not addressed with sufficient emphasis.

9. Several authors emphasize both the (restricted) necessity and feasibility of building self-explainable systems that monitor and analyse their behaviour and generate an explanation for human stakeholders involved in supervision based on explanation models. This approach however loses its significance in the case of systems with high level of autonomy.

10. Self-adaptive systems mostly consider parametric, functional, and architectural properties that capture concerns such as performance, reliability, and cost. A recent development in research is addressing non-functional or para-functional characteristics of NG-CPSs, such as trust, awareness, intellect, and emotions. These topics seem to be ready for immediate or near-future research.

Acknowledgments

The guest editors are very grateful to everybody who supported and/or contributed to the realization of this special issue. We sincerely appreciate the understanding cooperation of the authors in the rather long-lasting, multi-cycle review and publication process. The subsequent revisions of the submitted manuscripts helped a lot to consolidate the contents of the papers and to achieve the best possible results. We thank the valuable contribution of the peer reviewers who expressed their opinions critically but very constructively. Many thanks go to the fellow editor colleagues for their inspiration and professional support.

References

[1] Franz, J.H. (2011). A Critique of Technology and Science: An Issue of Philosophy. *Southeast Asia: A Multidisciplinary Journal*, Vol 11, pp. 23-36.

[2] Yankovskaya, V.V., & Kukushkin, S.N. (2019). The Role of the High School in the “Triple Loop” Model: SCBIN Technologies. In: *IOP Conference Series: Earth and Environmental Science*. IOP Publishing, 274(1), p. 012115.

[3] Wetter, K.J. (2006). Implications of Technologies Converging at the Nano-Scale: Are We Ready for the Little BANG? In: *Nanotechnologien Nachhaltig Gestalten*. Institut für Kirche und Gesellschaft: Iserlohn, Germany, pp. 31-41.

[4] Horváth, I., & Gerritsen, B.H. (2012). Cyber-Physical Systems: Concepts, Technologies and Implementation Principles. In: *Proceedings of the International Symposium on Toools and Methods of Competitive Engineering - TMCE 2012*, Vol. 1, Delft university Press, Delft, pp. 19-36.

[5] Ospina, P.D., Díaz, M.C., & Lara, S. (2016). New Technologies in Eye Surgery - A Challenge for Clinical, Therapeutic, and Eye Surgeons. In: *Advances in Eye Surgery*. IntechOpen, pp. 1-20.

[6] Zhang, F., DiSanto, W., Ren, J., Dou, Z., Yang, Q., & Huang, H. (2011). A Novel CPS System for Evaluating a Neural-Machine Interface for Artificial Legs. In: *Proceedings of the IEEE/ACM Second International Conference on Cyber-Physical Systems*, IEEE, pp. 67-76.

[7] Denkena, B., Dittrich, M.A., Stamm, S., Wichmann, M., & Wilmsmeier, S. (2021). Gentelligent Processes in Biologically Inspired Manufacturing. *CIRP Journal of Manufacturing Science and Technology*, 32, pp. 1-15.

[8] Horváth, I., Rusák, Z., & Li, Y. (2017). Order Beyond Chaos: Introducing the Notion of Generation to Characterize the Continuously Evolving Implementations of Cyber-Physical Systems. In: *Proceedings of the International Design Engineering Technical Conferences*, Vol. 58110, American Society of Mechanical Engineers. p. V001T02A015.

[9] Fitzgerald, J., Larsen, P.G., & Verhoef, M. (2014). From Embedded to Cyber-Physical Systems: Challenges and Future Directions, In: *Collaborative Design for Embedded Systems*, Springer, Heidelberg New York, pp. 293-302.

[10] Zhang, J.J., Wang, F.Y., Wang, X., Xiong, G., Zhu, F., Lv, Y., ... & Lee, Y. (2018). Cyber-physical-social systems: The state of the art and perspectives. *IEEE Transactions on Computational Social Systems*, 5(3), pp. 829-840.

[11] Black, W.S., Haghi, P., & Ariyur, K.B. (2014). Adaptive Systems: History, Techniques, Problems, and Perspectives. *Systems*, 2(4), pp. 606-660.

[12] Bock, W.J. (1980). The Definition and Recognition of Biological Adaptation. *American Zoologist*, 20(1), pp. 217-227.

[13] Holland, J.H. (1992). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Boston.

[14] Horváth, I., Suárez Rivero, J.P., & Hernández Castellano, P. M. (2019). Past, Present and Future of Behaviourally Adaptive Engineered Systems. *Journal of Integrated Design and Process Science*, 23(1), pp. 1-15.

[15] Tavčar, J., & Horváth, I. (2018). A Review of the Principles of Designing Smart Cyber-Physical Systems for Runtime Adaptation: Learned Lessons and Open Issues. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(1), pp. 145-158.

[16] Shevtsov, S., Berekmeri, M., Weyns, D., & Maggio, M. (2017). Control-Theoretical Software Adaptation: A Systematic Literature Review. *IEEE Transactions on Software Engineering*, 44(8), pp. 784-810.

[17] Salehie, M., & Tahvildari, L. (2009). Self-Adaptive Software: Landscape and Research Challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4(2), pp. 1-42.

[18] Patikirikorala, T., Colman, A., Han, J., & Wang, L. (2012). A Systematic Survey on the Design of Self-Adaptive Software Systems using Control Engineering Approaches. In: *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, IEEE, pp. 33-42.

[19] Swarnkar, P., Jain, S.K., & Nema, R.K. (2014). Adaptive Control Schemes for Improving the Control System Dynamics: A Review. *IETE Technical Review*, 31(1), pp. 17-33.

[20] Estrada-Jimenez, L. A., Pulikottil, T., Peres, R. S., Nikghadam-Hojjati, S., & Barata, J. (2021). Complexity Theory and Self-Organization in Cyber-Physical Production Systems. *Procedia CIRP*, 104, pp. 1831-1836.

[21] Khoramshahi, M., & Billard, A. (2019). A Dynamical System Approach to Task-Adaptation in Physical Human–Robot Interaction. *Autonomous Robots*, 43(4), pp. 927-946.

[22] Cámara, J., Lopes, A., Garlan, D., & Schmerl, B. (2016). Adaptation Impact and Environment Models for Architecture-Based Self-Adaptive Systems. S*cience of Computer Programming*, 127, pp. 50-75.

[23] Cómbita, L.F., Giraldo, J., Cárdenas, A.A., & Quijano, N. (2015). Response and Reconfiguration of Cyber-Physical Control Systems: A Survey. In: *Proceedings of the 2nd Colombian Conference on Automatic Control*, IEEE, pp. 1-6.

[24] Miraz, M.H., Ali, M., Excell, P.S., & Picking, R. (2015). A Review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT). In: *Proceedings of the 2015 Internet Technologies and Applications Conference*, IEEE, pp. 219-224.

[25] Brun, Y., Di Marzo Serugendo, G., Gacek, C., Giese, H., Kienle, H., Litoiu, M., Muller, H., Pezze, M., & Shaw, M. (2009), Engineering Self-Adaptive Systems through Feedback Loops. In: *Software Engineering for Self-Adaptive Systems*, Springer-Verlag, Berlin, pp. 48-70.

[26] Heylighen, F. (2011). Self-Organization of Complex, Intelligent Systems: An Action Ontology for Transdisciplinary Integration. *Integral Review*, 134, pp. 1-39.

[27] Floch, J., Hallsteinsen, S., Stav, E., Eliassen, F., Lund, K., & Gjorven, E. (2006). Using Architecture Models For Runtime Adaptability. *IEEE Software,* 23(2), pp. 62-70.

[28] Vogel, T., & Giese, H. (2013). Model-Driven Engineering of Adaptation Engines for Self-Adaptive Software: Executable Runtime Megamodels. *Technische Berichte No. 66*. Universitätsverlag Potsdam. 1-59.

[29] Kephart, J.O., & Chess, D.M. (2003). The Vision of Autonomic Computing. *Computer*. 36(1), pp. 41–50.

[30] Miller, B. (2005). The autonomic computing edge: The Role of Knowledge in Autonomic Systems. DB2 Developer Domain. http://www128.ibm.com/developerworks/autonomic/library/ac-edge6/.

[31] Iglesia, D.G., & Weyns, D. (2015). MAPE-K Formal Templates to Rigorously Design Behaviors for Self-Adaptive Systems. *ACM Transactions on Autonomous and Adaptive Systems*, 10(3), pp. 1-31.

[32] Tavčar, J., & Horváth, I. (2020), How Can a Smart Cyber-Physical System Validate its Runtime Adaptation Actions Before and After Executing Them?, In: *Proceedings of the Thirteenth International Symposium on Tools and Methods of Competitive Engineering Symposium,* Delft University of Technology, pp. 103-114.

[33] Hidaka, S., Hu, Z., Litoiu, M., Liu, L., Martin, P., Peng, X., ... & Yu, Y. (2019). Design and Engineering of Adaptive Software Systems. In: *Engineering Adaptive Software Systems*, Springer, Singapore, pp. 1-33.

[34] Muccini, H., Sharaf, M., & Weyns, D. (2016). Self-Adaptation for Cyber-Physical Systems: A Systematic Literature Review. In: *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 75-81.

[35] Chandra, A., Lewis, P.R., Glette, K., & Stilkerich, S.C. (2016). Reference Architecture for Self-Aware and Self-Expressive Computing Systems. In: *Self-Aware Computing Systems*, Springer, Cham. pp. 37-49.

[36] Hummaida, A.R., Paton, N.W., & Sakellariou, R. (2016). Adaptation in Cloud Resource Configuration: A Survey. *Journal of Cloud Computing*, 5(1), pp. 1-16.

[37] Cámara, J., Papadopoulos, A.V., Vogel, T., Weyns, D., Garlan, D., Huang, S., & Tei, K. (2020). Towards Bridging the Gap between Control and Self-Adaptive System Properties. In: *Proceedings of the 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, *IEEE/ACM*, pp. 78-84.

[38] Sanchez, M., Exposito, E., Aguilar, J. (2020). Autonomic Computing in Manufacturing Process Coordination in Industry 4.0 Context. *Journal of Industrial Information Integration*, 19, pp.1-16.

[39] Rodrigues, A., Caldas, R.D., Rodrigues, G.N., Vogel, T., & Pelliccione, P. (2018). A Learning Approach to Enhance Assurances for Real-Time Self-Adaptive Systems. In: *Proceedings of 13th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, IEEE, pp. 206-216.

[40] Engelenburg, S., Janssen, M., & Klievink, B. (2019), Designing Context-Aware Systems: A Method for Understanding and Analysing Context in Practice. *Journal of Logical and Algebraic Methods in Programming*, 103(2019), pp. 79-104.

[41] Don, S., Choi, E., & Min, D. (2012). Event Driven Adaptive Awareness System for Medical Cyber Physical Systems. In: *Proceedings of the 4th International Conference on Awareness Science and Technology*. IEEE, pp. 238-242.

[42] Braberman, V., D'Ippolito, N., Kramer, J., Sykes, D., & Uchitel, S. (2015), MORPH: A Reference Architecture for Configuration and Behaviour Self-Adaptation. In: *Proceedings of the 1st International Workshop on Control Theory for Software Engineering*, August 2015 pp. 9-16.

[43] Cansado, A., Canal, C., Salaün, G., & Cubo, J. (2010). A Formal Framework for Structural Reconfiguration of Components under Behavioural Adaptation. *Electronic Notes in Theoretical Computer Science*, 263, pp. 95-110.

[44] Garlan, D. (2001). Software Architecture. In: *Wiley Encyclopedia of Software Engineering*. John Wiley & Sons, Inc., New York. pp. 1-10.

[45] Cheng, S.W., Garlan, D., & Schmerl, B. (2006). Architecture-Based Self-Adaptation in the Presence of Multiple Objectives. In: *Proceedings of the 2006 International Workshop on Self-Adaptation and Self-Managing Systems*, pp. 2-8.
[46] Villegas, N.M., Tamura, G., & Müller, H.A. (2017). Architecting Software Systems for Runtime Self-Adaptation: Concepts, Models, and Challenges. In: *Managing Trade-offs in Adaptable Software Architectures*, Morgan Kaufmann, x, pp. 17-43.
[47] Kramer J., & Magee, J. (2007). Self-Managed Systems: An Architectural Challenge. In: *Proceedings of the Conference on the Future of Software Engineering*. Washington, DC, IEEE Computer Society, pp. 259-268.
[48] Phan, L.T., & Lee, I. (2011). Towards a Compositional Multi-Modal Framework for Adaptive Cyber-Physical Systems. In: *Proceedings of the 17th International Conference on Embedded and Real-Time Computing Systems and Applications*, Vol. 2, IEEE, pp. 67-73.
[49] McKinley, P.K., Sadjadi, S.M., Kasten, E.P., & Cheng, B.H. (2004). Composing Adaptive Software. *Computer*, 37(7), pp. 56-64.
[50] Masrur, A., Kit, M., Matěna, V., Bureš, T., & Hardt, W. (2016). Component-Based Design of Cyber-Physical Applications with Safety-Critical Requirements. *Microprocessors and Microsystems*, 42, pp. 70-86.
[51] Weyns, D., & Andersson, J. (2013). On the Challenges of Self-Adaptation in Systems of Systems. In: *Proceedings of the First International Workshop on Software Engineering for Systems-of-Systems*, pp. 47-51.
[52] Blair, G., Bencomo, N., & France, R.B. (2009). “Models@run.time,” *Computer*, 42(10), pp. 22-27.
[53] Szvetits, M., & Zdun, U. (2016). Systematic Literature Review of the Objectives, Techniques, Kinds, and Architectures of Models at Runtime. *Software & Systems Modeling*, 15(1), pp. 31-69.
[54] Cheng, B.H., Eder, K.I., Gogolla, M., Grunske, L., Litoiu, M., Müller, H.A., ... & Villegas, N.M. (2014). Using Models at Runtime to Address Assurance for Self-Adaptive Systems. In: *Models@Run.time*. Lecture Notes in Computer Science, Vol. 8378, Springer, Cham. pp. 101-136.
[55] Bennaceur, A., France, R., Tamburrelli, G., Vogel, T., Mosterman, P. J., Cazzola, W., ... & Redlich, D. (2014). Mechanisms for Leveraging Models at Runtime in Self-Adaptive Software. In: *Models@Run.time*. Lecture Notes in Computer Science, Vol. 8378, Springer, Cham. pp. 19-46.
[56] Klös, V., Göthel, T., & Glesner, S. (2018). Comprehensible and Dependable Self-Learning Self-Adaptive Systems. *Journal of Systems Architecture*, 85, pp. 28-42.
[57] Hadj-Kacem, N., Kacem, A.H., & Drira, K. (2009). A Formal Model of a Multi-Step Coordination Protocol for Self-Adaptive Software Using Coloured Petri Nets. *International Journal of Computing and Information Sciences*, 7(1), pp. 1-15.
[58] Loukil, S., Kallel, S., Jmaiel, M. (2017). An Approach Based on Runtime Models for Developing Dynamically Adaptive Systems. *Future Generation Computer Systems*, 68(3), pp. 365-375.
[59] Müller, M., Müller, T., Talkhestani, B.A., Marks, P., Jazdi, N., & Weyrich, M. (2021). Industrial Autonomous Systems: A Survey on Definitions, Characteristics and Abilities. *AT-Automatisierungstechnik*, 69(1), pp. 3-13.
[60] Casadei, R., Pianini, D., Placuzzi, A., Viroli, M., & Weyns, D. (2020). Pulverization in Cyber-Physical Systems: Engineering the Self-Organizing Logic Separated from Deployment. *Future Internet*, 12(11), 203, pp. 1-28.
[61] Lee, J., Azamfar, M., Singh, J., & Siahpour, S. (2020). Integration of digital twin and deep learning in cyber-physical systems: Towards smart manufacturing. In: *IET Collaborative Intelligent Manufacturing*, 2(1), pp. 34-36.
[62] Zhou, P., Zuo, D., Hou, K.M., Zhang, Z., Dong, J., Li, J., & Zhou, H. (2019). A Comprehensive Technological Survey on the Dependable Self-Management CPS: From Self-Adaptive Architecture to Self-Management Strategies. *Sensors*, 19(5), p. 1033.
[63] Pearce, H., Yang, X., Pinisetty, S., & Roop, P.S. (2021). Runtime Interchange for Adaptive Re-use of Intelligent Cyber-Physical System Controllers. arXiv preprint arXiv:2110.01974.

[64] Macher, G., Diwold, K., Veledar, O., Armengaud, E., & Römer, K. (2019). The Quest for Infrastructures and Engineering Methods Enabling Highly Dynamic Autonomous Systems. In: *Proceedings of the European Conference on Software Process Improvement*. Springer, Cham, pp. 15-27.

[65] Kühnle, H., & Bayanifar, H. (2017). An Approach to Develop an Intelligent Distributed Dependability and Security Supervision and Control For Industry 4.0 Systems. *Asian Journal of Information and Communications*, 9, pp. 8-16.

[66] Tan, T., & Chen, J. (2021). Model Predictive and Inversive Control for State Transition of Dynamics Systems. *Advances in Astronautics Science and Technology*, pp. 1-6.

[67] Colombo, A.W., Karnouskos, S., & Bangemann, T. (2014). Towards the Next Generation of Industrial Cyber-Physical Systems. In: *Industrial Cloud-Based Cyber-Physical Systems*. Springer, Cham, pp. 1-22.

[68] Yilma, B.A., Panetto, H., & Naudet, Y. (2021). Systemic Formalisation of Cyber-Physical-Social System (CPSS): A Systematic Literature Review. *Computers in Industry*, 129, pp. 1-25.

[69] Johnson, B., & Hernandez, A. (2016). Exploring Engineered Complex Adaptive Systems of Systems. *Procedia Computer Science*, 95, pp. 58-65.

[70] Johnson, B. (2020). Intelligent and Adaptive Systems of Systems for Engineering Desired Self-organization and Emergent Behavior. In: *Proceedings of the Future Technologies Conference*, Springer, Cham, pp. 126-146.

[71] Ali, S., Al Balushi, T., Nadir, Z., & Hussain, O. K. (2018). Standards for CPS. In: *Cyber Security for Cyber Physical Systems*. Springer, Cham, pp. 161-174.

[72] Ahmadi, A., Cherifi, C., Cheutet, V., & Ouzrout, Y. (2017). A Review of CPS 5 Components Architecture for Manufacturing Based on Standards. In: *Proceedings of the 11th International Conference on Software, Knowledge, Information Management and Applications*, IEEE, pp. 1-6.

[73] Hohwy, J. (2020). Self-Supervision, Normativity and the Free Energy Principle. *Synthese*, 10. pp. x-x

[74] Yin, D., Ming, X., & Zhang, X. (2020). Understanding Data-Driven Cyber-Physical-Social System (D-CPSS) Using a 7C Framework in Social Manufacturing Context. *Sensors*, 20(18), p. 5319.

[75] Yilma, B. A., Naudet, Y., & Panetto, H. (2018). Introduction to Personalisation in Cyber-Physical-Social Systems. In: *Proceedings of the OTM Confederated International Conferences on the Move to Meaningful Internet Systems*, Springer, Cham, pp. 25-35.

[76] Klös, V., Göthel, T., Glesner, S. (2018). Runtime Management and Quantitative Evaluation of Changing System Goals in Complex Autonomous Systems. *Journal of Systems and Software*, Vol. 144, October 2018, pp. 314-327.

[77] Elkhodary, A., Esfahani, N., & Malek, S. (2010). FUSION: A Framework for Engineering Self-Tuning Self-Adaptive Software Systems. In: *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 7-16.

[78] Cámara, J., Garlan, D., Moreno, G.A., & Schmerl, B. (2017). Evaluating Trade-Offs of Human Involvement in Self-Adaptive Systems. In: *Managing Trade-Offs in Adaptable Software Architectures*, Morgan Kaufmann, pp. 155-180.

[79] Cimini, C., Pirola, F., Pinto, R., & Cavalieri, S. (2020). A Human-in-the-Loop Manufacturing Control Architecture for the Next Generation of Production Systems. *Journal of Manufacturing Systems*, 54, pp. 258-271.

[80] Firouznia, M., Peng, C., & Hui, Q. (2018). Toward Human-in-the-Loop Supervisory Control for Cyber-Physical Networks. arXiv preprint arXiv:1805.02611.

[81] Cogliati, D., Falchetto, M., Pau, D., Roveri, M., & Viscardi, G. (2018). Intelligent Cyber-Physical Systems for Industry 4.0. In: *Proceedings of the First International Conference on Artificial Intelligence for Industries*, IEEE, pp. 19-22.

[82] Finogeev, A., Finogeev, A., Fionova, L., Lyapin, A., Lychagin K. A. (2019), Intelligent Monitoring System for Smart Road Environment, *Journal of Industrial Information Integration*, 15(9), pp. 15-20.

[83] Krishna, C.M., & Koren, I. (2013). Adaptive Fault-Tolerance for Cyber-Physical Systems. In: *Proceedings of the International Conference on Computing, Networking and Communications*. IEEE, pp. 310-314.

[84] Ruíz Arenas, S. (2018). Exploring the role of system operation modes in failure analysis in the context of first generation cyber-physical systems (*Doctoral dissertation*, Delft University of Technology, Delft, the Netherlands).

[85] Eirinakis, P., Kalaboukas, K., Lounis, S., Mourtos, I., Rožanec, J. M., Stojanovic, N., & Zois, G. (2020). Enhancing Cognition for Digital Twins. In: *Proceedings of the International Conference on Engineering, Technology and Innovation*, IEEE, pp. 1-7.

[86] Müller, H.A., Rivera, L.F., Jiménez, M., Villegas, N M., Tamura, G., Akkiraju, R., ... & Erpenbach, E. (2021). Proactive AIOps Through Digital Twins. In: *Proceedings of the 31st Annual International Conference on Computer Science and Software Engineering*, pp. 275-276.

[87] Ferrer, B.R., Mohammed, W.M., Lastra, J.L., Villalonga, A., Beruvides, G., Castaño, F., & Haber, R.E. (2018). Towards the Adoption of Cyber-Physical Systems of Systems Paradigm in Smart Manufacturing Environments. In*: Proceedings of the 16th International Conference on Industrial Informatics*, IEEE, pp. 792-799.

[88] Axelsson, J. (2019). Game Theory Applications in Systems-of-Systems Engineering: A Literature Review and Synthesis. *Procedia Computer Science*, 153, pp. 154-165.

[89] Du, D., Guo, T., & Wang, Y. (2020). DSML4CS: An Executable Domain-Specific Modeling Language for Co-Simulation Service in CPS. *International Journal of Web Services Research*, 17(2), pp. 59-75.

[90] Buffoni, L., Ochel, L., Pop, A., Fritzson, P., Fors, N., Hedin, G., ... & Sjölund, M. (2021). Open Source Languages and Methods for Cyber-Physical System Development: Overview and Case Studies. *Electronics*, 10(8), pp. 902-x.

[91] Ruchkin, I. (2019). Integration of Modeling Methods for Cyber-Physical Systems (Doctoral dissertation (*PhD thesis*, Institute for Software Research School of Computer Science, Carnegie Mellon University, Pittsburgh, CMU-ISR-18-107).

[92] Merelli, E., Paoletti, N., & Tesei, L. (2012). A Multi-Level Model for Self-Adaptive Systems. arXiv preprint arXiv:1209.1628.

[93] Frank. U. (2014). Multilevel Modeling - Toward a New Paradigm of Conceptual Modeling and Information Systems Design. *Business & Information Systems Engineering*, 6(6), pp. 319–337.

[94] Kühne. A. (2018). Story of Levels. In: *Proceedings of the 5th International Workshop on Multi-Level Modelling*, Volume 2245 of CEUR Workshop Proceedings, pp. 673–682.

[95] Khakpour, N., Jalili, S., Talcott, C., Sirjani, M., & Mousavi, M. (2012). Formal Modeling of Evolving Self-Adaptive Systems. *Science of Computer Programming*, 78(1), pp. 3-26.

[96] Hartsell, C., Mahadevan, N., Ramakrishna, S., Dubey, A., Bapty, T., Johnson, T., ... & Karsai, G. (2019). Model-Based Design for CPS with Learning-Enabled Components. In: *Proceedings of the Workshop on Design Automation for CPS and IoT*, pp. 1-9.

[97] Bonci, A., Pirani, M., Bianconi, C., & Longhi, S. (2018). RMAS: Relational Multiagent System for CPS Prototyping and Programming. In: *Proceedings of the 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, IEEE, pp. 1-6.

[98] Zhang, L., Zeigler, B.P., & Laili, Y. (2019). Introduction to Model Engineering for Simulation. In: *Model Engineering for Simulation*, Academic Press, pp. 1-23.

[99] de Lamotte, F.F., Berruet, P., Rossi, A., & Philippe, J.L. (2008). Control Code Generation using Model Engineering for an Electric Train. *IFAC Proceedings Volumes*, 41(2), pp. 8327-8332.

[100] Fishwick, P.A. (1990). Toward an integrated approach to simulation model engineering. *International Journal of General System*, 17(1), pp. 1-19.

[101] Hashmi, M.A., Mo, J.P., & Beckett, R. (2021). Transdisciplinary Systems Approach to Realization of Digital Transformation. *Advanced Engineering Informatics*, 49, p. 101316.

[102] Mo, J.P., & Beckett, R.C. (2021). Transdisciplinary System of Systems Development in the Trend to X4.0 for Intelligent Manufacturing. *International Journal of Computer Integrated Manufacturing*, pp. 1-15.
[103] van Gigch, J.P. (1993). Metamodeling: The Epistemology of System Science. *Systems Practice*, 6(3), pp. 251-258.
[104] Sangiovanni-Vincentelli, A., Shukla, S.K., Sztipanovits, J., Yang, G., & Mathaikutty, D. A. (2009). Metamodeling: An Emerging Representation Paradigm for System-Level Design. *IEEE Design & Test of Computers*, 26(3), pp. 54-69.
[105] Odell, J., Nodine, M., & Levy, R. (2004). A Metamodel for Agents, Roles, and Groups. In: *Proceedings of the International Workshop on Agent-Oriented Software Engineering*, Springer, Berlin, Heidelberg, pp. 78-92.
[106] Yang, Z., Eddy, D., Krishnamurty, S., Grosse, I., & Lu, Y. (2018). A Super-Metamodeling Framework to Optimize System Predictability. In: *Proceedings of the International Design Engineering Technical Conferences,* Vol. 51722, American Society of Mechanical Engineers, p. V01AT02A009.
[107] Villar, E., Merino, J., Posadas, H., Henia, R., & Rioux, L. (2020). Mega-Modeling of Complex, Distributed, Heterogeneous CPS Systems. *Microprocessors and Microsystems*, 78, p. 103244.
[108] Gašević, D., Kaviani, N., & Hatala, M. (2007). On Metamodeling in Megamodels. In: *Proceedings of the International Conference on Model Driven Engineering Languages and Systems*, Springer, Berlin, Heidelberg, pp. 91-105.
[109] Favre, J. M., & Nguyen, T. (2005). Towards a Megamodel to Model Software Evolution Through Transformations. *Electronic Notes in Theoretical Computer Science*, 127(3), pp. 59-74.
[110] Vogel, T., & Giese, H. (2012). A Language for Feedback Loops in Self-Adaptive Systems: Executable Runtime Megamodels. In: Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE, pp. 129-138.
[111] Bayne, J.S. (1995). A Distributed Programming Model for Real-Time Industrial Control. *Control Engineering Practice*, 3(8), pp. 1133-1138.
[112] Bickhard, M.H. (2019). Dynamics Is Not Enough: An Interactivist Perspective. *Human Development*, 63(3-4), pp. 227-244.
[113] Gleizes, M.-P., Camps, V., George, J.-P., & Capera, D. (2007). Engineering Systems Which Generate Emergent Functionalities. In: *Proceedings of the Satellite Conference on Engineering Environment-Mediated Multiagent Systems*. Dresden, Germany, pp. x-x.
[114] Alcocer, J.P., Beck, F., & Bergel, A. (2019). Performance Evolution Matrix: Visualizing Performance Variations Along Software Versions. In: *Proceedings of the Working Conference on Software Visualization*, IEEE, pp. 1-11.
[115] Garces, L., Martinez-Fernandez, S., Neto, V.V. & Nakagawa, E.Y. (2020). Architectural Solutions for Self-Adaptive Systems. *Computer*, 53(12), pp. 47-59.
[116] Mokni, A., Urtado, C., Vauttier, S., Huchard, M., & Zhang, H.Y. (2016). A Formal Approach for Managing Component-Based Architecture Evolution. *Science of Computer Programming*, 127, pp. 24-49.
[117] Ballesteros, J., Ayala, I., Caro-Romero, J.R., Amor, M., Fuentes, L. (2020), Evolving Dynamic Self-Adaptation Policies of mHealth Systems for Long-Term Monitoring, *Journal of Biomedical Informatics*, pp. x-x.
[118] Zhou, P., Zuo, D., Hou, K.M., Zhang, Z., Dong, J., Li, J., & Zhou, H. (2019). A Comprehensive Technological Survey on the Dependable Self-Management CPS: From Self-Adaptive Architecture to Self-Management Strategies. *Sensors*, 19(5), 1033, x-x.
[119] Epifani, I., Ghezzi, C., Mirandola, R., & Tamburrelli, G. (2009). Model Evolution by Run-Time Parameter Adaptation. In: *Proceedings of the 31st International Conference on Software Engineering*, IEEE, pp. 111-121.

[120] Roth, F.M., Krupitzer, C., & Becker, C. (2015). Runtime Evolution of the Adaptation Logic in Self-Adaptive Systems. In: *Proceedings of the International Conference on Autonomic Computing*, IEEE, pp. 141-142.
[121] Pasquale, L., Baresi, L., & Nuseibeh, B. (2011). Towards Adaptive Systems Through Requirements@Runtime. In: *Proceedings of the 6th Workshop on Models@run.time*, p. 39.
[122] Basich, C., Svegliato, J., Wray, K.H., Witwicki, S., Biswas, J., & Zilberstein, S. (2020). Learning to Optimize Autonomy in Competence-Aware Systems. arXiv preprint arXiv:2003.07745.
[123] Löffler, D., Schmidt, N., & Tscharn, R. (2018). Multimodal expression Of Artificial Emotion In Social Robots Using Color, Motion and Sound. In: *Proceedings of the International Conference on Human-Robot Interaction*, ACM/IEEE, pp. 334-343.
[124] Zhu, Y.B., & Li, J.S. (2019). Collective Behavior Simulation Based nn Agent with Artificial Emotion. *Cluster Computing*, 22(3), pp. 5457-5465.
[125] Catuogno, L., Galdi, C., & Pasquino, N. (2018). An Effective Methodology For Measuring Software Resource Usage. *IEEE Transactions on Instrumentation and Measurement*, 67(10), pp. 2487-2494.
[126] Liu, Q., Li, Q., Li, Z., & Wu, H. (2020). Research on Software Resource Management Model Based on Cloud Service. In: *Proceedings of the 8th International Conference on Information Technology: IoT and Smart City*, pp. 65-68.
[127] Wan, J., Chen, B., Imran, M., Tao, F., Li, D., Liu, C., & Ahmad, S. (2018). Toward Dynamic Resources Management tor IoT-Based Manufacturing. *IEEE Communications Magazine*, 56(2), pp. 52-59.
[128] Hachicha, M., Halima, R.B., & Kacem, A.H. (2019). Formal Verification Approaches of Self-Adaptive Systems: A Survey. *Procedia Computer Science*, 159, pp. 1853-1862.
[129] Nia, M.A., Kargahi, M., & Faghih, F. (2020). Probabilistic Approximation of Runtime Quantitative Verification in Self-Adaptive Systems. *Microprocessors and Microsystems*, 72, p. 102943.
[130] Redfield, S.A., & Seto, M.L. (2017). Verification Challenges for Autonomous Systems. In: *Autonomy and Artificial Intelligence: A Threat or Savior?*, Springer, Cham, pp. 103-127.
[131] Eze, T., Anthony, R.J., Walshaw, C., & Soper, A. (2011). The Challenge of Validation for Autonomic and Self-Managing Systems. In: *Proceedings of the Seventh International Conference on Autonomic and Autonomous Systems*, pp. 128-133.
[132] Dewasurendra, S.D., Vidanapathirana, A.C., & Abeyratne, S.G. (2019). Integrating Runtime Validation and Hardware-in-the-Loop (HiL) testing with V&V in Complex Hybrid Systems. *Journal of the National Science Foundation of Sri Lanka*, 47, 4.
[133] Cardozo, N., Christophe, L., De Roover, C., & De Meuter, W. (2014). Run-Time Validation of Behavioral Adaptations. In: *Proceedings of 6th International Workshop on Context-Oriented Programming*, pp. 1-6.
[134] Abuseta, Y., & Swesi, K. (2015). Towards a Framework for Testing and Simulating Self-Adaptive Systems. In: *Proceeding of the 6th International Conference on Software Engineering and Service Science*, IEEE, pp. 70-76.
[135] Eberhardinger, B., Seebach, H., Knapp, A., & Reif, W. (2014). Towards Testing Self-Organizing, Adaptive Systems. In: *Proceedings of the IFIP International Conference on Testing Software and Systems*. Springer, Berlin, Heidelberg, pp. 180-185.
[136] Schierman, J., Ward, D., Dutoi, B., Aiello, A., Berryman, J., DeVore, M., ... & Wadley, J. (2008). Run-Time Verification and Validation for Safety-Critical Flight Control Systems. In: *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 6338.
[137] Marshall, A., Jahan, S., & Gamble, R. (2018). Toward Evaluating the Impact of Self-Adaptation on Security Control Certification. In: *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, pp. 149-160.
[138] Cámara, J., Peng, W., Garlan, D., & Schmerl, B. (2018). Reasoning about Sensing Uncertainty and its Reduction in Decision-Making for Self-Adaptation. *Science of Computer Programming*, 167, pp. 51-69.

[139] Drechsler, R., Lüth, C., Fey, G., & Güneysu, T. (2018). Towards Self-Explaining Digital Systems: A Design Methodology for the Next Generation. In: *Proceedings of the 3rd International Verification and Security Workshop*, IEEE, pp. 1-6.

[140] Blumreiter, M., Greenyer, J., Garcia, F.J., Klös, V., Schwammberger, M., Sommer, C., ... & Wortmann, A. (2019). Towards Self-Explainable Cyber-Physical Systems. In: *Proceedings of the 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion*, IEEE, pp. 543-548.

[141] Wickramasinghe, C.S., Amarasinghe, K., Marino, D.L., Rieger, C., & Manic, M. (2021). Explainable Unsupervised Machine Learning for Cyber-Physical Systems. *IEEE Access*, 9, pp. 131824-131843.

[142] Daglarli, E. (2021). Explainable Artificial Intelligence (xAI) Approaches and Deep Meta-Learning Models for Cyber-Physical Systems. In: *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems*, IGI Global, pp. 42-67.

[143] Huang, J., Bastani, F., Yen, I.L., & Jeng, J.J. (2009). Toward a Smart Cyber-Physical Space: A Context-Sensitive Resource-Explicit Service Model. In: *Proceedings of the 33rd Annual International Computer Software and Applications Conference*, Vol. 2, IEEE, pp. 122-127.

[144] Schmidt, D.C., White, J., & Gill, C.D. (2014). Elastic Infrastructure to Support Computing Clouds for Large-Scale Cyber-Physical Systems. In: *Proceedings of the 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, IEEE, pp. 56-63.

[145] González-Nalda, P., Etxeberria-Agiriano, I., Calvo, I., & Otero, M. C. (2017). A Modular CPS Architecture Design Based on ROS and Docker. *International Journal on Interactive Design and Manufacturing*, 11(4), pp. 949-955.

[146] Landolfi, G., Barni, A., Menato, S., Cavadini, F.A., Rovere, D., & Dal Maso, G. (2018). Design of a multi-sided platform supporting CPS deployment in the automation market. In: *Proceedings of the 2018 IEEE Industrial Cyber-Physical Systems*, IEEE, pp. 684-689.

[147] Shin, D.Y., & Park, J.W. (2020). Modularization for Personal Social Service Robots. *The Journal of the Convergence on Culture Technology*, 6(2), pp. 349-355.

[148] Denkena, B., Henning, H., & Lorenzen, L.E. (2010). Genetics and Intelligence: New Approaches in Production Engineering. *Production Engineering*, 4(1), pp. 65-73.

[149] Lachmayer, R., Mozgova, I., & Scheidel, W. (2016). An Approach to Describe Gentelligent Components in Their Life Cycle. *Procedia Technology*, 26, pp. 199-206.

[150] Möhring, H.C., Wiederkehr, P., Erkorkmaz, K., & Kakinuma, Y. (2020). Self-optimizing Machining Systems. *CIRP Annals*, 69(2), pp. 740-763.

[151] Cuevas, A.D., & Guzman-Arenas, A. (2010). Automatic Fusion of Knowledge Stored in Ontologies. *Intelligent Decision Technologies*, 4(1), pp. 5-19.

[152] Zhang, S., Yang, L.T., Feng, J., Wei, W., Cui, Z., Xie, X., & Yan, P. (2021). A Tensor-Network-Based Big Data Fusion Framework for Cyber-Physical-Social Systems (CPSS). *Information Fusion*, 76(2021), pp. 337-354.

[153] Sahu, A., Mao, Z., Wlazlo, P., Huang, H., Davis, K., Goulart, A., & Zonouz, S. (2021). Multi-Source Multi-Domain Data Fusion for Cyberattack Detection in Power Systems. *IEEE Access*, 9, pp. 119118-119138.

[154] Petrovska, A., Quijano, S., Gerostathopoulos, I., & Pretschner, A. (2020). Knowledge Aggregation with Subjective Logic in Multi-Agent Self-Adaptive Cyber-Physical Systems. In: *Proceedings of the 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, IEEE/ACM, pp. 149-155.

[155] Bürger, J., Kehrer, T., & Jürjens, J. (2020). Ontology Evolution in the Context of Model-Based Secure Software Engineering. In: *Proceedings of the International Conference on Research Challenges in Information Science, Springer*, Cham, pp. 437-454.

[156] Kotis, K.I., Vouros, G.A., & Spiliotopoulos, D. (2020). Ontology Engineering Methodologies for the Evolution of Living and Reused Ontologies: Status, Trends, Findings and Recommendations. *The Knowledge Engineering Review*, 35(4), pp. 1-34.

[157] Ferranti, N., Soares, S.S., & de Souza, J.F. (2021). Metaheuristics-Based Ontology Meta-Matching Approaches. *Expert Systems with Applications*, 173, 114578.
[158] Horváth, I. (2020). On Reasonable Inquiry and Analysis Domains of Sympérasmology. *Journal of Integrated Design and Process Science*, 24(1), pp. 5-43.
[159] Gembarski, P.C. (2019). The Meaning of Solution Space Modelling and Knowledge-Based Product Configurators for Smart Service Systems. In: *Proceedings of the International Conference on Information Systems Architecture and Technology*, Springer, Cham, pp. 28-37.
[160] Abel, G. (2014). Systematic Knowledge Research. Rethinking Epistemology. *Rivista Internazionale di Filosofia e Psicologia*, 5(1), pp. 13-28.

Author Biographies

Dr. Imre Horváth is emeritus professor of the Faculty of Industrial Design Engineering, Delft University of Technology, the Netherlands. In the last years, his research group focused on research, development, and education of smart cyber-physical system design, with special attention to cognitive engineering. Prof. Horváth is also interested in systematic design research methodologies. He was the promotor of more than 20 Ph.D. students. He was first author or co-author of more than 430 publications. His scientific work was recognized by five best paper awards. He has a wide range of society memberships and contribution. He is past chair of the Executive Committee of the CIE Division of ASME. Since 2011, he is a fellow of ASME. He is member of the Royal Dutch Institute of Engineers. He received honorary doctor titles from two universities, and the Pahl-Beitz ICONNN award for internationally outstanding contribution to design science and education. He was distinguished with the Lifetime Achievement Award by the ASME's CIE Division in 2019. He has served several international journals as editor. He was the initiator of the series of International Tools and Methods of Competitive Engineering (TMCE) Symposia. His current research interests are in various philosophical, methodological, and computational aspects of smart product, system, and service design, as well as in synthetic knowledge science and development of self-adaptive systems.

Dr. Jože Tavčar has been working as senior lecturer at Product Development Division, LTH, of the University of Lund, Sweden, since 2020. He earned B.Sc., M.Sc., and Ph.D. degrees in mechanical engineering from the University of Ljubljana, in 1991, 1994, and 1999, respectively. He started his research career in 1991, focusing on technical information systems, information flow in product development, processes re-engineering, and methodology of design. Starting in 2006, he spent a decade in industry as the Head of the Noise and Vibration Lab at the Domel Company, and as the Head of the Quality Department at Iskra Mehanizmi, respectively. During this period he was involved in several product-development teams of international corporations such as Philips, Electrolux, and Rowenta. He co-ordinated the development of a motor diagnostic system and studied various topics of noise-reduction and vibrations. He was also developing quality systems for the automotive industry (ISO/IATF 16949) and for medical devices (ISO 13485, FDA). Between 2011 and 2020, he was lecturing at the University of Ljubljana in various courses such as Design Methodology, Engineering Design Techniques, and Engineering Design from Non-Metallic Materials. He was guest researcher at the ProSTEP AG, Germany, in 1995, at the University Karlsruhe, Germany, in 1996, and at the Delft University of Technology, the Netherlands, in 2016. Now, he has a unique combination of concrete product development experiences and a holistic view on system approach. His current research topics are designing smart cyber-physical systems, data mining and big-data analysis, and application of agility methods and knowledge management in product-development processes. He published over 40 SCI papers, over 50 conference papers, 5 book chapters, and over 80 technical reports.