Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Perspective:

Human-cyber-physical systems: concepts, challenges, and research opportunities*

Zhiming LIU‡[1], Ji WANG[2]

[1]*RISE-Centre for Research and Innovation in Software Engineering, School of Computer and Information Science, Southwest University, Chongqing 400715, China*

[2]*State Key Laboratory for High Performance Computing, School of Computer, National University of Defense Technology, Changsha 410073, China*

E-mail: zhimingliu88@swu.edu.cn; jiwang@ios.ac.cn

Received Oct. 10, 2020; Revision accepted Oct. 16, 2020; Crosschecked Oct. 26, 2020

Abstract: In this perspective article, we first recall the historic background of human-cyber-physical systems (HCPSs), and then introduce and clarify important concepts. We discuss the key challenges in establishing the scientific foundation from a system engineering point of view, including (1) complex heterogeneity, (2) lack of appropriate abstractions, (3) dynamic black-box integration of heterogeneous systems, (4) complex requirements for functionalities, performance, and quality of services, and (5) design, implementation, and maintenance of HCPS to meet requirements. Then we propose four research directions to tackle the challenges, including (1) abstractions and computational theory of HCPS, (2) theories and methods of HCPS architecture modelling, (3) specification and verification of model properties, and (4) software-defined HCPS. The article also serves as the editorial of this special section on cyber-physical systems and summarises the four articles included in this special section.

Key words: Abstractions; Architecture modelling; Evolution; Software-defined technology
https://doi.org/10.1631/FITEE.2000537 **CLC number:** TP311

1 Introduction

Human-cyber-physical system (HCPS) is one of the most hyped buzzwords in the computing science and technology, control engineering, communication, and information communication technology (ICT) application communities. In light of the intension and extension of a concept, HCPS is not entirely new, but it has emerged from the continuous evolution and integration of the science and technologies of computing, control, and communication. Nevertheless, serious research has been active in investigating the potential applications and challenges in its scientific foundation and engineering techniques, especially those systems that are built by integrating many heterogeneous systems built in multi-disciplinary technologies.

We propose to study HCPS as

(1) the state of the art of ICT, which will continuously evolve in the future;

(2) the emerging architecture style of engineering systems that are formed with cyber-systems (computing systems consisting of hardware and software); physical systems including mechanical, electrical, and chemical processes; human systems in the forms of individuals, organisational and social systems, which are constituent systems (subsystems or components) connected through a network for

‡ Corresponding author

* Project supported in part by the Capacity Development Fund of Southwest University, China (No. SWU116007) and the National Natural Science Foundation of China (Nos. 61732019, 61672435, 61811530327, and 62032019)

ORCID: Zhiming LIU, https://orcid.org/0000-0001-9771-3071

interactions and cooperations; and

(3) the enabling technology for solving major challenges in sustainable development, including the inter-related areas of design and management of urban development, energy needs, social and economic development, and financial and industry development.

HCPS applications are happening, although they are developed mostly using ad-hoc processes, in most areas of major infrastructure development, key areas of development that are crucially important to the national economy and wellbeing of people, including smart grids, smart cities (smart buildings and smart homes), smart transportation, smart education, smart healthcare and medicine, and national defence. Because of the potential enabling capacity in digitising our living, social, and economic activities, all major industrial countries and regions have made strategies and plans for their development beyond 2020, which include the USA Industry Internet, German (European Union) Industry 4.0, and Made in China 2025, which are all based on the development of cyber-physical system (CPS) technology. The notion of CPS was first proposed by Helen GILL at the National Science Foundation of the United States to refer to the integration of computation with physical processes (NSF, 2006; Gill, 2010). HCPS is a natural extension of CPS that adds the consideration of human interactions and cooperations with cyber systems and physical systems, supported by ICT.

To promote research in the area of HCPS, we have organised this special section on CPSs. As its editorial, we have written this perspective article to discuss the development of the HCPS concept, evolution of system architecture, and integration of related technologies. The rest of the article is organised as follows. In Section 2, we recall the historical background and introduce the basic concepts of HCPS. We devote Section 3 to the discussion of the major challenges to the development of the scientific foundation, methods, and tools for building HCPS from a system engineering perspective. Then in Section 4, we propose four research directions towards solving the challenges. Finally in Section 5, we give a summary of the articles that are included in this special section.

2 Background and basic concepts

As mentioned in the previous section, HCPS extends CPS that originated at the first Cyber-Physical Systems Workshop (NSF, 2006), led by Helen GILL. The workshop continued with future events in the following years, focusing on defining the area and discussing the challenges and visions (Lee EA, 2006). In the same year, the European Union Research Council organised a few horizontal working groups to discuss major key areas of research that the council should support. One of the working groups was the Software Intensive Systems Working Group (The first author of this article was a member of the working group). The working group organised three meetings during 2007–2010 and proposed a concept of "ensemble engineering" (Wirsing et al., 2008), which had a similar intention and extension to that of CPS. At the workgroup meetings, societal computing and organic computing were proposed as different concepts, which were later partly incorporated into the notion of CPS (and ensemble) and partly formed the area of societal computing. In this section, we first introduce CPS concepts and problems, then the integration of CPS with big data and cloud computing to form the notion of big data and cloud based CPS (BDC-CPS), and the evolution to the concept of HCPS.

2.1 CPS: intersection of computation, control, and communication

CPS emerged during the development of the technology of embedded systems and control systems, and the rapid advances in communication technology in recent years. Although CPSs have been studied intensively, there is no current standard for CPS architectures. Some descriptive definitions have been given by leading researchers. Here we quote a few widely cited definitions:

(1) Rajkumar et al. (2010) described CPSs as physical and engineered systems whose operations are monitored, coordinated, controlled, and integrated by a computing and communication core.

(2) Lee EA (2010) defined CPSs as integrations of computation with physical processes. He also explained that embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations, and vice versa.

(3) Baheti and Gill (2011) referred to CPSs as

"a new generation of systems with integrated computational and physical capabilities that can interact with humans through many new modalities". The ability to interact with and expand the capabilities of the physical world through computation, communication, and control, is a key enabler for future technology developments.

Based on the above descriptions, we give the following architectural definition for a CPS (Gunes et al., 2014; Khaitan and McCalley, 2015):

A CPS is a system of systems (SoS) comprising cyber systems, communication networks, physical processes, and interfaces, where:

(1) the physical processes include, for example, mechanical, electrical, and chemical processes, which are embedded with sensors and controlled by microprocessors;

(2) the cyber systems are computing systems that are responsible for data processing and controlling physical processes;

(3) the interfaces are middleware systems between the physical systems and the network, including sensors and actuators, A/C and C/A convertors, etc.;

(4) the sensors sense the physical processes and collect data about the behaviours of the physical processes, and the data are transmitted to the cyber systems through the network;

(5) the cyber systems process the collected data and compute decisions for the control and coordination of the behaviours of the physical processes, and the control decisions are transmitted in the form of control commands through the network to the corresponding actuators to carry out the control actions.

Some people consider the network as part of the cyber systems, whereas some others treat the network as part of the interfaces. The physical systems are typically independent embedded systems and are called "unit systems" in a CPS. The unit systems can be designed by different people, executed independently and managed by their own organisations. When they are used to construct a CPS, they are then coordinated by the CPS control units to execute the CPS business processes for the CPS mission. The architecture of such a CPS is shown in Fig. 1 in a simplified view. The CPS control units perform control at a layer above that of the unit systems. The CPS control decisions, i.e., those of the CPS control units, are computed based on the data from the sensors and temporal and spatial events generated during the operation of the CPS (Tan et al., 2009). Therefore, there is usually a database server to which the control units can make queries for making control decisions. The CPS control units also provide services to the users who provide requirements for reconfiguration of control policies and rules.

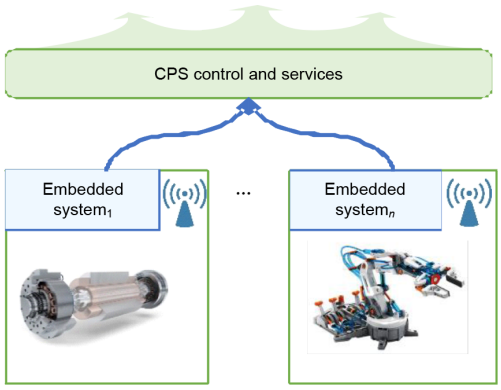

Fig. 1 Architecture of a cyber-physical system (CPS)

We note that the main concern of early CPSs described by the above architecture is to control, coordinate, and manage the physical processes for the realisation of business tasks and processes in the CPS mission. A CPS of this kind still has the characteristic of a closed control loop of observe-decide-act (ODA). However, it is no longer a single isolated input-output system, but a network system of many input-output systems. Such a CPS can thus be understood as a distributed network of embedded systems exhibiting the behaviours of direct interactions among physical processes. The aim of the design of such a CPS is increased autonomy and reduced human involvement in the control of physical systems to automate the execution of the business processes. The main challenge in CPS research concerns interaction between cyber and physical systems and coordination of the subsystems to meet the requirements of the CPS mission. Human knowledge, behaviours, and interactions with cyber systems and physical systems are not the concern of the CPS.

CPS applications are mainly sensor network based distributed control systems such as environment monitoring and control systems, smart power grids, self-driving systems, smart transportation systems, smart home and smart building systems, and distributed robotic systems. A concrete

example of a simple CPS is the MIT Robot Garden (https://www.csail.mit.edu/research/distributed-robot-garden).

Therefore, they can be potentially applied to the development of systems to control largely distributed systems, such as power grid control and management, industry control systems, and self-driving (or self-flying) systems. In the MIT Robot Garden system, a group of robots cooperate with each other to look after a tomato garden. The design of this system is given with the integration of technologies of a distributed sensor network, control system, automatic navigation, and wireless network.

2.2 Big data and cloud based CPS

A CPS, such as a smart grid system, contains various networks of different kinds, functionalities, and scales, including wired and wireless networks, wireless local area networks, and communication networks. It also uses sensors and actuators that are made by different companies, with different functionalities and of different qualities. During the operation of the system, many events are generated (Lee J et al., 2013; Xu and Duan, 2019). The management and control of these networks, sensors, and actuators, and the coordination of the behaviours of physical systems and other hardware and software systems require processing and analysis to meet the system mission requirements, both functional and performance requirements including real-time requirements, fault-tolerance, stability, adaptivity, safety, and security. More systematically, we summarise the values of data in CPS as follows:

(1) The data collected by various sensors which are of different qualities and the data of events generated during the operation of the system can be used to develop data analytic and artificial intelligence (AI) algorithms, artificial neural networks in particular, for more accurate and effective control of physical processes, especially when real-time restrictions are required (Xu and Duan, 2019).

(2) These data can also be used for detecting, predicting, and handling abnormalities and accidents for fault-tolerance and run-time monitoring, in order to meet the safety, security, and reliability requirements (Liu and Joseph, 1996, 1999; Zhang et al., 2009; Lee J et al., 2013; Xu and Duan, 2019).

(3) The data can be used to analyse, estimate, and predict uncertain behaviours of the physical environment, network, and complex cyber systems, in order to develop middleware systems for self-adaption and reconfiguration to improve the predictability, sustainability, effectiveness, adaptiveness, and autonomy (Banerjee et al., 2012; Lee J et al., 2013; Gunes et al., 2014; Khaitan and McCalley, 2015; Zegzhda, 2016; Xu and Duan, 2019).

(4) Furthermore, value-added services can be developed with the large amount of data that is collected by and generated in the system. For example, data from a city's lightening system can be used by the city government to develop services for the citizens, police authorities to develop social order related functionalities and services in their systems, and the electricity companies to develop analytic software for their business (Liu et al., 2019).

Although the data of a CPS are of the above values, their volume, variety, and velocity require techniques and computer systems beyond those of traditional database systems. Big data and cloud computing technologies provide the techniques and flexible platforms that store the big data of a CPS and for developing, implementing, and running the required analytic software and services. However, big data technology in CPS is not only about offline batch processing and fusion of data, but also, and even more importantly, for real-time data processing. This is a new challenge in CPS. Furthermore, how data analytic models and models of system engineering can be combined in CPS system integration is another significant theoretical challenge.

In addition to the provision of a flexible platform for big data storage, processing, analytics, and services to different users, cloud computing allows data analytic models and software to be provided to the CPS control units. With a cloud computing platform in a CPS, all devices and software products, including sensors, actuators, computing resources, data, application software, and services, can be rented on demand in the execution of business tasks. Because a cloud platform has a clear feature of service-oriented architecture (SOA) style, a CPS can be combined with SOA to support integration of cross-domain CPS. Such an integrated system of CPS forms a new generation of system-driven equipment and it is called a big data and cloud based CPS (BDC-CPS). For example, with the support of big data technology, cloud computing, and SOA, a manufacturing enterprise CPS can be formed based on its product

design and production CPS, supply CPS, product management, and sales CPS.

A CPS introduced in the previous subsection has a three-tiered (or layered) architecture that includes a unit system layer, a CPS control layer, and a service layer. A BDC-CPS has a multilayer version of the three-tiered architecture. Each layer integrates the BDC-CPS systems of the layer below it through its control layer and provides services to users of this layer (Liu et al., 2019). The three tiers of a layer in a CPS are illustrated in Fig. 2. Therefore, big data and cloud computing technologies can largely augment the functionality and performance of a CPS and support vertically upward extension and horizontal extension, to form ultra large systems of societal scale and societal impact. It is important to note that big data and cloud computing provide solutions to achieve the requirements of fault tolerance, self-adaptation, safety, reliability, stability, real-time constraints, etc., but they are a cause of problems of violation of these requirements.

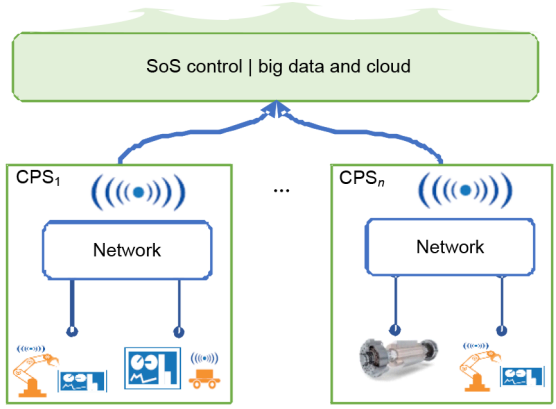

Fig. 2 The three tiers in a layer of a big data and cloud based CPS (BDC-CPS)

2.3 Human-cyber-physical systems

The discussions in the previous two subsections show that early research on CPS focusses on the integration of computation, communication, and control, i.e., 3C-technology integration. With the support of mobile communication and computation, big data and cloud computing technologies, the early CPS now has been evolving to a system of systems for integration and coordination of cross-domain CPS. The architectures of these CPSs are increasingly hierarchical and open. Their functionalities are even more powerful, and support more complex distributed and collaborative workflows involving distributed stakeholders and actors, including human actors, across different domains.

While CPSs are evolving, features and requirements of their applications are growing and require spatial-temporal dynamics, coordination, and intelligence, exposed to problems related to interactions between human actors and cyber and physical systems, especially those in which control is required to be switched between humans and machines. These problems are potential causes of accidents like the two Boeing Max accidents that happened in October 2018 and March 2019. When it became human knowledge and behaviours, human-cyber and human-physical interactions and collaborations to be controlled and managed in a CPS, it comes the notion of human-cyber-physical systems (HCPSs) (Romero et al., 2016), cyber-physical-human systems (CPHSs) (Sowe et al., 2016), or cyber-physical-social systems (CPSSs) (Sheth et al., 2013).

2.3.1 Ubiquitous computing and CPS

To our best knowledge, there are no clear discussions about the architectural and technological differences between HCPS and CPS. We try to understand HCPS from their ubiquitous computing features (Weiser, 1991; Kindberg and Fox, 2002). We recall that Weiser's vision of ubiquitous computing (Weiser, 1991) was for information technology to "weave themselves into the fabric of everyday life until they are indistinguishable from it", or for information technology to "become part of the environment". This would allow us, the humans, in our daily life activities and work, to intentionally or unintentionally use informational technology, or to be supported by, with or without our notice, information technology. Weiser called this way of embedding information technology into the environment "embedded virtuality", which is related to, but not the same as, today's popular virtual reality.

We all realise that our human abilities of observing, collecting, remembering (due to the limit of our memory), processing, and analysing information are very limited and there is little space for further natural extension. The idea of ubiquitous computing is very much revolutionary because ICT can provide unlimited augmentation of the above human abilities

of humans which are required in observation, decision making, and taking action during our daily life and work. For example, we cannot see clearly beyond 50 metres when we drive in thick fog, but with the support computer embedded in the environment, we can "see" anything as far as 1000 metres; we are not able to draw a conclusion with a huge amount of information about a fast-changing environment, but a computer can help us make the right decision quickly with its data processing and analytic power.

The idea of "embedded virtuality" is to embed the results of data processing, computing, and analysis through algorithms into the physical environment, and for computers to generate a smart environment in which humans can live and work. Ubiquitous computing is thus also widely called "ambient intelligence", which is also referred to as "smart environment" (Kindberg and Fox, 2002). The ambition of Weiser is far from being realised. The deep dynamic (in time and space) fusion of computation and the physical world that Weiser envisioned has not been achieved, and large-scale ubiquitous computing systems that dynamically construct non-intrusive intelligent environments do not yet exist. However, we can see, in principle at least, the possibility of improving interactions between humans and cyber systems and between humans and physical systems.

2.3.2 Internet of Things and ubiquitous computing

Applications of ubiquitous computing require the development of many sensing devices. For example, there are many sensors in a car. Some are used for sensing and monitoring the running conditions of the car and the changes in the outside environment to ensure that the car is running smoothly and safely. There are sensors that monitor the temperature and humidity inside the car for passenger comfort. There are also road sensors and sensors of transport equipment. Sensing devices can dynamically change and are mobile, and thus require dynamic and ubiquitous connectivity. These issues used to be significant challenges to the realisation of ubiquitous computing, but now it is easy to see that the Internet of Things (IoT) provides natural technology support. The essence of the IoT is to provide connectivity among "things" and support virtualisation and identification of "things". IoT systems nowadays have evolved into the form of sensor networks in which software is embedded in sensing devices, physical systems, mobile phones, and electrical devices (Lu and Cecil, 2015). Here, physical systems include human systems. Therefore, the modern IoT is actually an Internet of human systems, cyber systems, and physical systems that provides information technology infrastructure for HCPS.

Although controlling the behaviours of physical processes is not its main concern, a ubiquitous computing system is required for the state of the intelligent environment to dynamically change in real time along with the change in the physical environment in which humans live and work. For this, the IoT must provide connectivity and communication among the "things"—human, cyber, and physical systems—but also big data technology, machine learning, cloud computing, and fog and edge computing technologies, to provide computing devices and services anywhere and anytime. Because of rapid and lasting advances in big data and cloud computing technologies, ubiquitous computing based on the Internet of human, cyber, and physical systems is being transformed from traditional environment awareness or environment intelligence to include social awareness or smart society.

We reiterate that the Internet of human, cyber, and physical systems based on IoT, big data, and cloud computing focusses on environment sensing and virtualisation. Compared to the traditional ICT, environment sensing and virtualisation largely augment human cognition and decision-making abilities. Thus, we are entering the era of the Internet of everything, and human beings will live in an intelligent environment. From this perspective, we see HCPSs as a deep integration of CPSs and the above as the Internet of human, cyber, and physical systems.

2.3.3 Human-cyber-physical systems

The notion of HCPS was proposed only recently, and there are three main trends for its interpretation and understanding:

(1) One understanding in HCPS is that human elements are generally treated as physical entities. Then these human elements are interconnected with CPSs of different domains, and controlled and coordinated (rather passively) by cyber systems in the whole system. At the same time, in this kind of HCPS, the importance of big data and cloud computing based control and services has been

emphasised in Broy et al. (2102).

(2) Another trend of understanding is the study of models of human psychology and brain behaviour and their interactions with CPS models. The integration and interaction between models of human psychology and brain behaviour and CPS models will augment human abilities and improve human performance in observation, analysis, cognition, decision making, and operations (Romero et al., 2016; Zhou J et al., 2019).

(3) The third view of HCPS is the integration of CPS and cyber-social systems (CSSs), which forms the so-called cyber-physical-social systems (CPSSs); here, CSSs include social networks and social computing systems (Sheth et al., 2013; Dressler, 2018; Zeng J et al., 2020).

With the understanding of the system characteristics of CPS, ubiquitous computing, and the IoT, and the nature of big data and cloud computing technologies, we consider HCPS as an integration of humans as individuals, social organisations, and social networks in intelligent environments, physical processes, and cyber systems based on big data and cloud computing technologies. Cyber systems include the HCPS control of the coordination of the behaviours and interactions among the human, physical, and cyber systems in the subsystems, and services to the system users. Control software and services rely on big data and cloud computing. In addition, human systems behave in their intelligent environments, and these intelligent environments are generated by ubiquitous computing software.

Apart from involvement of human knowledge and behaviours and dynamic control switching between humans and cyber systems in the operation of the system, HCPSs are like BDC-CPS and they are formed from cross-domain HCPSs in a multiple layered architecture. Each layer has three tiers of sub-HCPS, HCPS control, and services to users, as shown in Fig. 3.

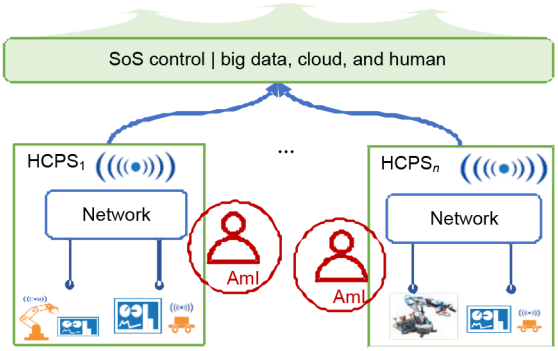

Fig. 3 The three tiers in a layer of a human-cyber-physical system (HCPS)

2.3.4 Conceptual summary and classification

There are several popular terms that are commonly used in relation to CPS and HCPS: IoT, ubiquitous computing, ambient intelligence, smart environment, human-machine and machine-machine interactions, hybrid systems, big data, and cloud computing. The discussions in the above subsections clarify how CPS and HCPS are different from those concepts and the roles that their technologies play in CPS and HCPS engineering systems. Here, we summarise with clarifications as follows:

(1) Hybrid systems are isolated input-output embedded or control systems, and CPSs are networked systems of these hybrid systems as unit systems.

(2) Big data technology supports development of better and intelligent control algorithms, algorithms for monitoring, predicting, and handling abnormalities and uncertainty, and development of value-added services.

(3) Cloud computing provides platform solutions for big data technology and enables cross-domain HCPS interactions and collaborations. Big data and cloud computing technologies are used for the requirements of trustworthiness (e.g., safety, security, reliability, and stability) and performance (e.g., fault tolerance, realtime, and self-adaption).

(4) IoT is mainly for connecting and virtualising "physical things" as data so that the "things" can be identified, tracked, and shared.

(5) Ubiquitous computing is about using and processing data of "physical things" to generate views of the things as smart environments. The realisation of this then requires technologies, computing power, and distributed cloud computing platforms. Smart environments are used in the control and coordination of the physical systems (which then cause changes in the smart environments) and services to the customers.

More importantly, humans in an HCPS behave and interact with cyber and physical systems in their smart environments (indicated as "Aml" in Fig. 3) which greatly augment human abilities.

Therefore, HCPS is about the intersection of

IoT, hybrid systems, ubiquitous computing, big data, cloud computing, human-machine interaction, and machine-machine interaction, rather than their union.

3 Challenges

From the architectural descriptions of CPS, BDC-CPS, and HCPS, we understand, among other problems, the following features and challenges of HCPS.

3.1 Complex heterogeneity

An HCPS comprises heterogeneous constituent systems and devices, including:

(1) various human systems that belong to different organisations and play different roles in the operation of the system;

(2) different physical systems that are designed by different people using different techniques and tools, run dependently and managed by different organisations;

(3) different sensors and actuators of different functionalities, makes, and qualities; and

(4) different software systems that are designed using different technologies and architectural styles, implemented in different programming languages and run on different platforms.

Different software, hardware, physical, and human resources are shared during the execution of distributed and collaborative business processes and workflows, and these resources are of different features. The interactions among the heterogeneous subsystems are very much of different natures.

3.2 Lack of appropriate abstractions

We all understand that advances of computer systems and software engineering have been driven by ideas of abstractions and their automation (In fact this is true for any system engineering discipline). Important software abstractions in the history include the symbolic assembler, subroutine, function and procedure, high-level programming language and compiler, abstract datatype, modularity, object-orientation, and architecture styles (Liu et al., 2019). These, together with abstractions of hardware resources, have contributed to advances in programming languages, operating systems, software architecture, software development tools, and environments. Later, abstractions such as synchronisation, critical sections and regions, mutual exclusion and conditional synchronisation, and atomic actions are all critically important for the development of concurrent and distributed computation and operation systems. We note that all abstractions are represented in models and integrated into system architecture styles, which are common and useful system organisations. On the subject of the impacts of software abstractions, we refer the reader to the recent perspective article (Liu et al., 2019). There, we also argued for the essence and importance of abstractions, including that:

(1) abstractions are an essential and effective means of mastering complexities;

(2) abstractions allow us to focus on common aspects and ignore differences;

(3) abstractions allow to work with high-level information, without the need of low-level domain knowledge;

(4) abstractions are the key to engineering support for the principles of separation of concerns that allow divide and conquer;

(5) abstractions are the key to development of generic principles and systematic methods; and

(6) higher levels of abstraction enable higher degrees of automation.

In system engineering, abstractions emerge from intuitions, and are then developed into models and theories, eventually leading to automated mechanisms. They drive the development of systematic methods and tools.

One of the major challenges to the development of HCPS engineering is the lack of appropriate abstractions. First of all, most physical process controls have real-time constraints, but as Lee EA (2006, 2008) noted, there is no abstraction for precisely deciding the time required in computation. For example, there are four layers of abstraction in embedded system design. They are the bottom layer of microprocessor, the second layer of x86 instruction set architecture (ISA), the third layer of bytecode programs, and the fourth layer of source code programs (e.g., Java code). The purpose of this layered structure is for the designer of one layer not to be concerned with the details of the layers below. However, the layers of abstractions fail to meet this purpose in some respects, especially when real-time

requirements are concerned. For example, ISA does not provide a means to guarantee the user's real-time requirements. Also, programming languages do not have suitable abstraction for time and therefore, the execution time of a program cannot be well controlled to meet the user's real-time requirements. In embedded system design, timing analysis uses estimate of worst case execution time (WCET) and this is done based on the operating system that has good predictive scheduling policies. With extensive bench testing, WCET analysis can guarantee real-time constraints for small-scale and closed embedded systems. However, this will not work for ultra-large-scale and open HCPS.

Another serious problem with regard to abstractions in HCPS is the mismatches of the abstractions in cyber systems and physical systems. State changes of physical systems are continuous and in real time, and their abstract models are in general, for example, the ordinary differential equation (ODE), differential-algebraic equation (DAE), and partial differential equation (PDE). On the contrary, the states and state changes of cyber systems are discrete, and their models are in general automata or state machines. Time, concurrency, and synchronisation in physical systems are all truly physical, happening in real time. However, clocks in computer systems are all approximations, and execution of programs is essentially sequential. Concurrency in programs is simply a result of the abstraction of internal execution details and it exhibits complex nondeterminism. We can program concurrency and synchronisation in programs using interruptions to suspend and resume a process, but this does not in general work for physical processes. Therefore, it is hard to define a model of interfaces between cyber and physical systems to characterise their interactions, concurrency, and synchronisation. It is important to note that handling the interaction, concurrency, and synchronisation among software processes and physical processes in classical embedded systems (or in the so-called hybrid systems) is not such a significant problem. This is because the systems are closed and of small scale. Also, when such a closed system fails, it would not affect any other system. However, when a large number of such systems are opened up and integrated in an HCPS, failures of an individual system can be propagated to other constituent systems and cause unintended emergent behaviours (Hu et al., 2008). Also, the current approach to interactions of embedded systems with sensors and actuators based on interrupts does not have abstraction in programming language (Lee EA, 2008).

Furthermore, we have to develop abstractions for interaction, concurrency, and synchronisation between humans, humans and machines (cyber systems), and humans and physical systems, and these abstractions could be dependent on the ability and roles of humans in the system. These abstractions are needed for the analysis and design of systems for monitoring and control of human behaviours, and for coordinating human behaviours with the behaviours of cyber and physical systems. Being intelligent and autonomous, humans as individuals, organisations, and social systems are another dimension of source of uncertainty, and thus pose difficulties in the design, analysis, and verification of an HCPS.

3.3 Dynamic black-box integration of heterogeneous systems

From the introduction to BDC-CPS in Section 2.2 and HCPS in Section 2.3, we see an HCPS as an integration of many heterogeneous subsystems. The behaviours of the HCPS emerge through the HCPS control services of the behaviours and interactions of many heterogeneous subsystems for their coordination. The HCPS control services are organised in hierarchical layers and each layer controls and coordinates the interactions and behaviours of a set of subsystems. The control services of a layer include orchestration and choreography of the services of the subsystems of the layer and their composition into business process services of the corresponding layer of abstraction. These subsystems are designed by different people using varying technology frameworks, running independently, and controlled and managed by different organisations.

The subsystems are used as black-box systems in the integration of the HCPS for its mission requirements. Even for a subsystem that is developed for a specific HCPS, abstract black-box models (i.e., interface models) should be developed and documented. Black-box models for subsystems are used for general purposes, such as:

(1) by high-level designers, who do not have or are unable to understand the low-level details of the design and implementation of the subsystem,

(2) for the maintenance of the subsystem in the

HCPS, such as its upgrades and replacement, and

(3) in other contexts within the HCPS and even in other HCPS applications.

A major challenge to black-box integration of subsystems of complex heterogeneity in HCPS is to achieve the interoperability among the subsystems. Interoperability in cyber systems is traditionally defined to be "the ability of two or more systems or components to exchange information and to correctly use such information" (IEEE, 1990). It requires that the information exchange should work well even in scenarios where the subsystems have differences in programming languages, interfaces, and execution platforms. The difficulty is mainly due to the fact that a common understanding of the data formats, procedures, contracts, standards, quality, and interfaces is difficult to achieve among the different stakeholders. There is a large body of literature concerning interoperability of cyber systems and we refer the reader to Chen D et al. (2008) and Kubicek et al. (2011). However, interoperability of multiple dimensions must be considered in the development of an HCPS, which needs interaction and cooperation among different organisations, suppliers, and types of systems. With the HCPS layered architecture style, horizontal interoperability requirements must be defined to allow interactions and exchange of data and information among heterogeneous and distributed systems that are provided and developed by different suppliers. Vertical interoperability requirements should be established to allow cooperation among cross-domain stakeholders, organisations, and businesses acting in distinct domains and located in various environments.

It is important to note that interoperable integration is the key to ensuring the system requirements of HCPS, including performance safety, security, reliability, and stability. Furthermore, a large HCPS cannot be built from scratch. It always starts from a simple and working initial system, and then continuously evolves (Liu and Chen, 2014; Liu et al., 2019). Bottom-up evolution of an HCPS allows individual subsystems to enter or exit from the HCPS or refine their behaviours at any moment, and requires adaptations of SoS internal structures. On the other hand, top-down evolution is required when system requirements and/or business processes change. We propose a general evolutionary framework to allow the following system refinement:

(1) Plug in new subsystems or components and refine existing systems and components.

(2) Dynamically find and connect subsystems and components.

(3) Add more interface devices and/or improve their performance, such as allowing cyber systems to:

(a) monitor more and better about its environment,

(b) become more autonomous (self*),

(c) make more intelligent control decisions and provide smarter services, and

(d) refine the HCPS control and coordinate more and better physical components.

An HCPS is continuously evolving to improve optimisation, smartness, connectivity, autonomy, and trustworthiness (safety, security, reliability, and dependability). We note that evolution takes technology development into account, such as improved technology for sensor production, advanced technologies related to the application domains of HCPS, and development of new technologies in computing, information, and communication.

3.4 Complex requirements for functionalities, performance, and quality of services

The mission of an HCPS is to provide solutions to complex problems in our everyday life, and social, economic, and industrial development, including social order, biomedicine and health, transportation, and manufacture (Rajkumar et al., 2010; Gunes et al., 2014; Khaitan and McCalley, 2015). Different applications can have varying requirements with respect to functionality, performance, and quality of service (QoS), which are outlined below:

(1) High reliability. Reliability requirements include safety, fault tolerance, and real-time requirements. Applications, such as those in medicine and health, industry control systems, manufacturing systems, equipment monitoring, and control systems, have high reliability requirements.

(2) Security. Security is required to prevent information leakage and malicious attacks, and protect privacy. Applications in medicine and health, transportation and industry control systems, smart home, and smart cities have high security requirements.

(3) Adaptability. Requirements for adaptability are to ensure the stable and sustainable execution of the system through effective handling of dynamic

uncertainty in the internal execution environment, external operational environment, and in the requirements. It is important to understand that not only the environment is open with complex uncertainty but also the HCPS system itself is open with subsystems dynamically being added in or removed out, and mobile systems being moving into or out of a local execution environment. Therefore, effective self-adaptation and reconfiguration must be designed to cope with the complex uncertainty due to the double openness.

It is a great challenge to understand, formulate, and specify the requirements for an HCPS, and to our best knowledge, an established method for requirement capture and analysis of HCPS requirements does not exist.

3.5 Design, implementation, and maintenance of HCPS to meet requirements

Because of the lack of understanding of the principles, abstractions, and theoretical foundations, we do not have systematic methods and tools for correct design and implementation of an HCPS, or technology frameworks for maintaining an HCPS to ensure that it continuously evolves and grows healthily. More concretely, there are few clear ideas about and approaches to:

(1) defining an engineering process for the mission of an HCPS (it seems that neither a top-down decomposition and refinement approach nor a bottom-up synthesis approach alone would work);

(2) defining architectural strategies for the selection of existing subsystems or their capabilities which are needed and feasible for the mission requirements of HCPS;

(3) defining strategies and selecting strategies for interoperable integration of individual, distributed, and smart subsystems.

There are also known difficulties in the realisation of orchestration and choreography of behaviours of heterogenous subsystems. IoT, big data, and cloud computing can in principle provide technology and platform support in connectivity, data processing, analytics and usage, control decision making, and control policy reconfiguration in the system operation and maintenance. They also bring in challenges such as the following:

(1) Resource control and management. IoT and cloud platforms in an HCPS control and manage a large amount of resources. They are heterogenous components of the HCPS and include data, hardware, software services, physical processes, and even human elements. Many of these resources are created for specific tasks and problems. They are normally at a large granularity and rigid; thus, they are not flexible for dynamic reallocation and adjustment at runtime for changing requirements and environments. This implies the need and difficulties in development of abstractions and a virtual model of heterogenous resources for building system software including operating systems and middleware components (Tröger et al., 2015; Schätz, 2016; Mei and Guo, 2018) to satisfy the system adaptability, stability, resilience, and real-time constraints.

(2) Trustworthiness. Big data and cloud computing provide the technologies and facilities for virtualisation of hardware and physical systems as resources. Virtualised resources can be duplicated to provide redundancies for ubiquitous connectivity and fault tolerance. The big data and facilities for the development and execution of AI algorithms, those of powerful deep neural networks (DNNs) in particular, can help handle problems caused by the complex uncertainty of HCPS, either in the internal behaviours or in the external environment. However, these intelligent systems, together with human intelligence and autonomy, are serious causes of uncertainty too. Their trustworthiness, in the state of the art of HCPS and AI, cannot yet be defined. Thus, we are facing significant challenges in verifying, controlling, composing, and reusing DNNs when they are integrated in engineering systems like the HCPS.

(3) Control and management of interactions of heterogenous subsystems and intelligent agents. In an HCPS, heterogenous human, physical, and cyber systems, which can be mobile, join and leave the system through the network. Human and other intelligent agents in different layers of abstraction interact and communicate with each other. Therefore, defining and implementing their communication protocols is challenging. Also, given requirements, there are no design and programming methods for designing interoperable scheduling, coordination, and orchestration of human and intelligent agents (Calvaresi et al., 2017).

4 Proposed research directions

Since 2006 when the notion of CPS was proposed, there have been intensive studies. An important part of the research has been about defining the area of research and investigating major challenges. Examples of pioneering work include Lee EA (2006, 2008, 2010), Sha et al. (2008), Gill (2010), and Rajkumar et al. (2010). Most of the remaining work attempts mainly to use existing models and techniques in computing, communication, and control engineering and their conservative extensions in building CPS, but there is little advance in the study of the intersection of the three engineering disciplines. Based on our literature study, initial research experience and results, and the challenges discussed in the previous section, we propose the following research directions.

4.1 Abstractions and computational theory of HCPS

From a system engineering perspective, development of systematic methods, techniques, and tools in HCPS engineering relies on a well-founded scientific foundation and abstractions. Abstractions are built up for general characteristics of the systems, and general principles and idiomatic solutions to common problems in constructing, operating, and maintaining HCPS. The relationships among abstractions, scientific foundation, methods, and tools are that abstractions come as intuitions gained in practice; then they are rigorously studied and modelled to form theories, developed as methods and tools, and implemented as system constructs and mechanisms. These constructs and mechanisms are the fundamental elements used to build system architectures.

The development of computer systems and software engineering has been driven by abstractions, their theories, and their implementations, such as symbolic assemblers (Wilkes et al., 1951), high-level programming languages and compilers (Giloi, 1997), subroutines (Wheeler, 1952), abstract datatypes (ADT) (Liskov and Zilles, 1974), modularity (Parnas, 1972; Lindsey and Boom, 1978) and object orientation (Nygaard and Dahl, 1978), atomic actions, interactions of concurrency and synchronisation, and architecture styles like component-based and service-oriented architecture styles (Liu et al., 2019). There are many well-established models and theories, e.g., theories of automata, state machines, algebraic theory of ADT, process algebras including CSP (Hoare, 1985) and CCS (Milner, 1989), and temporal logics.

Therefore, the first direction of research we propose is to develop necessary abstractions and a computational model of HCPS. The research starts with a focus on interactions among human systems, cyber systems, and physical systems, and the flows of control, data, and communication. To this end, we propose a theory of human-cyber-physical automata (HCP-A). HCP-A is an input/output automaton and is required to be able to model the interactions among human systems, cyber systems, and physical systems. It is an extension to the model of hybrid automata (Alur et al., 1995; Lynch N et al., 2003). However, it is not a trivial extension because HCP-A needs to capture the controlled switches of control between human systems and cyber systems. This implies that HCP-A has a learning component and it is context-aware and knows the model of the autonomous behaviours of human systems.

It is important to note here that we are not seeking to model generic human intelligence or the so-called strong AI (yet), but we believe that it is possible to have a model that learns the behaviours of a particular human system in a given execution scenario. Still, this is a new and difficult research problem, as we do not know any initial work in this direction. However, we understand that this is also very fundamental and that its importance can be compared with that of the traditional theory for classical programming, theory of I/O automata for concurrent and distributed computing, theory of real-time automata for real-time computing, etc.

The research can start with considering human-cyber interactions to develop a model of I/O automata, called intelligent I/O automata, with a learning oracle, to represent the intelligent behaviours of a human system or a DNN and their interaction with traditional I/O automata (Lynch NA, 1996). Then, we incrementally extend the model to a model of real-time intelligent I/O automata to deal with real-time constraints, and eventually a model of hybrid intelligent I/O automata for human-cyber-physical interactions and concurrency. To deal with the multi-dimensional uncertainty of HCPS, probabilistic extensions to the above models are required.

In the research on the above computational models, we need to study their expressiveness and

decidability problems. Composition and equivalence manipulations of models are also important research topics.

4.2 Theories and methods of HCPS architecture modelling

In general, the architecture of an engineering system defines its components or the subsystems, their functionalities, interaction relations, performance, and QoS. The architecture is hierarchical and can be represented at different levels of abstraction. The specifications at various levels are for system requirements, design, implementation, and maintenance.

System architecture is also important for defining and managing the system development process, which decides what subsystems will be integrated or designed and what methods and tools will be used according to the expertise of the development team (Liu et al., 2019). The study of architecture modelling is the basis for the development of methods and tools for system development, including modelling languages, programming models and languages, and system analysis, verification, and simulation.

An HCPS controls and coordinates many heterogeneous subsystems, but we do not yet have a clear model to define its architecture. We need such a modelling theory for these ultra-large-scale SoS in order to develop HCPS architecture modelling and design methods. Without such a theory, we cannot ensure the conceptual integrity and stability in the whole lifecycle of requirement definition, design, implementation, maintenance, and evolution.

Therefore, the second research direction we propose is theories and methods of HCPS architecture modelling. The aim is to establish unified mathematical models for describing the structures, functionality, performance, and QoS at different levels of abstraction. This is to establish the foundation for developing methods, techniques, and tools for model construction, analysis, decomposition, composition, and other manipulations. The main difficulty is to define a unified meta behaviour model for the heterogeneous subsystems for their interoperable integration. What we propose is a research programme on interface contract-based model-driven HCPS development which has, among others, the following components:

1. Model of HCPS interface contracts

Because an HCPS architecture integrates and controls, through different layers, many heterogeneous subsystems, their end-to-end interaction behaviours and composition are essential aspects to model. The model must support separation of the interface specification and its implementation for black-box integration. Therefore, an interface contract specifies an assumption about its environment and the behaviours it guarantees to deliver if the environment satisfies the assumption.

A subsystem in an HCPS has a large variety of aspects and we can understand them from several viewpoints. Thus, we can define models of different viewpoints for an HCPS interface, including its interface static type which defines the static structure of the interface, its external interaction type which specifies the interaction protocol and communication flow with the environment of the subsystem, and its dynamic behaviour type which defines the control flow and data flow of the subsystem.

The interface behaviour type also describes how the interface static type and interface interaction type are realised through the interface static types and interaction types of subsystems in the layer below, which is the coordination of the behaviours of the subsystems through their interfaces. Therefore, the interface contract model, simply called the contract, is an aggregation of these viewpoint models. The interface interaction type and behaviour type should be defined based on the definition of HCP-A in Section 4.1.

2. Theory of HCPS interface contract composition and refinement

To support the layered architecture style of the HCPS and black-box integration, we need to define the necessary composition operations of interface contracts and study their equivalence properties. The interface static type, interaction type, and behaviour type of a composite interface contract are determined through calculation of the interface static types, interaction types, and behaviour types of the component interface contracts. Algebraic properties of the composition operations must be studied.

The layered architecture also requires that an interface contract at a higher level is correctly realised by a composition of interface contracts at the layer below. We need a notion for the characterisation of this realisation relation of a contract by a

composition of several contracts. Furthermore, we need to support maintenance operations and system evolution by allowing a subsystem or a component of it at any layer to be replaced by a better or upgraded subsystem or a component. To this end, we need to define the relation of contract refinement. This relation is required to be a partial order; that is, it is reflexive, transitive, and antisymmetric.

Contract composition operations are also called architecture operations. They are required to preserve the refinement partial order. These operations provide information hiding, connectors, adaptors, data converters, etc. Refinement preserving architecture operations support architecture evolution to improve reliability, safety, security, adaptivity, and autonomy through adding and upgrading software components (including middleware components).

There is some initial progress in this direction. We refer the reader to the actor-based model (Lee EA et al., 2003), hybrid CSP (He, 1994), hybrid Hoare logic (Wang et al., 2015), contract-based design for CPS (Sangiovanni-Vincentelli et al., 2012), and our idea to extend rCOS (Palomar et al., 2016; Chen X and Liu, 2017; Liu et al., 2019).

4.3 Specification and verification of model properties

HCPS systems (and subsystems) are composition of heterogeneous subsystems and they have inter-related global and local properties of many kinds. Their specification and verification are difficult. Among other properties of a large SoS, the following properties are particularly important and difficult to specify and verify:

(1) Time-spatial properties. The time-spatial requirements of HCPS are to guarantee that right events occur at the right time and in the right location. This does not mean the faster the better, but more about the predictability of time and location in the design, implementation, and maintenance of the system. For example, we are not clear about how to specify when an event is to occur in the interface contract model, and how it can be realised by a composition of many components. The currently used model in programming is clock-based timed automata. However, these clocks do not have a semantic interpretation for physical processes. We envision a hybrid specification of time constraints that combines models of interval time, accumulative time, frequency, and superdense time (Zhou CC et al., 1991; Liu et al., 1998).

(2) Safety, security, and fault tolerance. We can see that, based on the theory and method of architecture modelling discussed in Section 4.2, middleware components implementing encryption algorithms, access control, runtime monitoring, fault detection, and fault recovery can be designed and deployed in an HCPS to refine existing subsystems and HCPS control software (Liu et al., 2008, 2019; Liu and Chen, 2014). These methods for security are in general known secure by design and fault-tolerant by transformation (Liu and Joseph, 1996, 1999). However, how these methods can be used for human systems and DNN systems is entirely unknown.

(3) Emergent behaviours. The global behaviours of an HCPS emerge from the interaction and collaboration of its many subsystems. Many of these behaviours cannot (or at least we do not know how they can) be defined by behaviours of subsystems (Hu et al., 2008). Examples of these behaviours reflect as system features of resilience and flexibility. Such behaviours are not achieved through the design of one or more subsystems but emerge as results of coordination and/or collaboration of the subsystems, and they are called emergent behaviours. Some emergent behaviours are required or desired, but some others are harmful and even disastrous. We do not know any general method for specifying and verifying emergent behaviours, or design methods to achieve or avoid an emergent behaviour.

Most of the current work on verification of safety, liveness, security, realtime, and robustness is to extend methods of theorem proving, model checking, and testing, based on extended models. We would like to point out that, due to the scale and complex heterogeneity, the verification techniques are relatively limited and their combination with simulation techniques is now taken seriously. Moreover, the construction of an HCPS often uses various kinds of legacy systems which do not have models. Therefore, model learning and control synthesis are also important techniques for model construction. In addition to the above properties, specification, verification, and simulation of properties of probabilistic models of human systems, machine learning systems, and other models dealing with uncertainty are all important research topics.

4.4 Software-defined HCPS

A very important nature of a layered architecture of an HCPS (discussed in Section 2) is that a control service of the control tier in a layer coordinates the behaviours of the subsystems in the layer below, through orchestration and choreography. Directly programming the coordination by coupling the behaviours of the subsystems is not feasible, because of the many subsystems. It is not feasible either, because there can be many different control services that coordinate the subsystems in different ways. This is why black-box coordination through interface specifications is required.

HCPS computational models and architecture models are used for the development of programming models, specification languages, programming languages, and software development tools and platforms for developing, running, and maintaining software systems of HCPS. This requires the establishment of abstractions for some data, software services, hardware devices, physical and human systems in a layer that are commonly used and shared among the business service of that layer to be encapsulated as virtual models of resources.

Resource models are the bases for the development of system software for easier and more effective and efficient control, management, and allocation of the resources to computational tasks. Resources in HCPS can be much more generalised than those of the input and output devices, memory, and processing units in traditional operating systems. We need to study what more abstractions are needed for HCPS resources and/or what extensions to the traditional abstractions such as memory and processing units should be developed. Furthermore, various kinds of system software in HCPS are organised in different layers where a higher-layer operating system coordinates several operating systems in a layer below. We can call them ubiquitous resources and ubiquitous operating systems (Mei and Guo, 2018), respectively.

The abstractions to encapsulate resources as virtual models and then to manage the resources by system software are the key to software for an HCPS, and it is very much like the principle of technologies of software-defined systems (Jararweh et al., 2015, 2016; Mei and Guo, 2018). The principle of software-defined techniques is not novel; its main idea is easy to understand in the following ways:

(1) separation of the laws of control from the physical hardware through abstractions of hardware and equipment,

(2) separation of the use of the resource functionalities from their implementations, through application programming interfaces (APIs), and

(3) higher-level business services and processes which are composed based on control flows and data flows.

The general form of programs in a software-defined system is as follows:

API-based encapsulated components + control program.

These ideas originate first from control engineering as in a programmable logic controller (PLC), and also the design of operating systems. They are then used in modern programming paradigms, such as object-oriented programs and service-oriented programs, in the following forms, respectively:

Classes + main program and
WSDL + BPEL program.

However, the main challenges in the development of methods and tools for software-defined HCPS (SD-HCPS) are to:

(1) define data (or virtual) models of heterogeneous resources and their capabilities;

(2) define APIs at different granularities for heterogeneous subsystems (including resources) for flexible coordination;

(3) define task models with information about resource requirements and execution models with resource utilities;

(4) design and analyse algorithms for heterogeneous resource management and allocation, involving hierarchical scheduling policies.

Although software-defined techniques have been available around for some time, it is not yet certain how they are effective for HCPS development. It is important to study the integration of technologies of software-defined network (Molina and Jacob, 2018), software-defined storage (Darabseh et al., 2015), software-defined cloud (Jararweh et al., 2016), software-defined IoT (Jararweh et al., 2015; Zeng DZ et al., 2020), software-defined security, and so on, to develop a coherent technology for software-defined HCPS.

5 Summary of the special section

We have presented our perspective on the main concepts of HCPS and challenges to further development of HCPS engineering. Our view is that although there has been active research in the area and a large body of work has been published, the research is still in its infancy and little progress has been made towards solutions to the challenges outlined in Section 3. We have thus proposed four interrelated research directions in Section 4 to promote further research efforts in building the scientific foundation for engineering HCPS.

As part of the effort, we have organised this special section on Model-Driven Software Development for Cyber-Physical Systems in *Frontiers of Information Technology & Electronic Engineering*, with the hope of presenting some research results related to the challenges. We received seven submissions and organised reviews following the requirements of the journal. We have accepted four manuscripts to include in this special section. Each of these four papers was accepted after revisions following the comments from two rounds of reviews by two or three experts. We present the abstracts of these papers as a summary below:

1. Emergence in cyber-physical systems: potential and risk, by Shmuel TYSZBEROWICZ and David FAITELSON

Cyber-physical systems (CPSs) are distributed assemblages of computing, communicating, and physical components that sense their environment, algorithmically assess the incoming information, and affect their physical environment. Thus, they share a common structure with other complex adaptive systems, and therefore share both the possible benefits and the probable harmful effects of emergent phenomena. Emergence is an often unexpected pattern that arises from the interactions among the individual system components and the environment. In this paper we focus on three major problems concerning emergence in the context of CPSs: how to successfully exploit emergence, how to avoid its detrimental effects in a single CPS, and how to avoid harmful emergence that arises due to unexpected interaction among several independently developed CPSs that are operating in the same environment. We review the state of the research with regard to these problems and outline several approaches that could be used to address them.

2. A survey of model-driven techniques and tools for cyber-physical systems, by Bo LIU et al.

Cyber-physical systems (CPSs) have emerged as a potential enabling technology to handle the challenges in social and economic sustainable development. Since it was proposed in 2006, intensive research has been conducted, showing that the construction of a CPS is a hard and complex engineering process due to the nature of integrating a large number of heterogeneous subsystems. Among other approaches to dealing with the complex design issues, model-driven design of CPSs has shown its advantages. In this review paper, we present a survey of research on model-driven development of CPSs. We are concerned mainly with the widely used methods, techniques, and tools, and discuss how these are applied to CPSs. We also present comparative analyses on the surveyed techniques and tools from various perspectives, including their modeling languages, functionalities, and the challenges which they address in CPS design. With our understanding of the surveyed methods, we believe that model-driven approaches are an inevitable choice in building CPSs and further research effort is needed in the development of model-driven theories, techniques, and tools. We also argue that a unified modeling platform is needed. Such a platform would benefit research in the academic community and practical development in industry, and improve the collaboration between these two communities.

3. Decentralized runtime enforcement for robotic swarms, by Chi HU et al.

Robotic swarms are usually designed in a bottom-up way, which can make robotic swarms vulnerable to environmental impact. It is particularly true for the widely used control mode of robotic swarms, where it is often the case that neither the correctness of the swarming tasks at the macro level nor the safety of the interaction among agents at the micro level can be guaranteed. To ensure that the behaviors are safe at runtime, it is necessary to take into account the property guard approaches for robotic swarms in uncertain environments. Runtime enforcement is an approach which can guarantee the given properties in system execution and has no scalability issue. Although some runtime enforcement methods have been studied and applied in different domains, they cannot effectively solve the problem of property enforcement

on robotic swarm tasks at present. In this paper, an enforcement method is proposed on swarms which should satisfy multi-level properties in uncertain environments. We introduce a macromicro property enforcing framework with the notion of agent shields and a discrete-time enforcing mechanism called $\mathcal{D}$-time enforcing. To realize this method, a domain specification language and the corresponding enforcer synthesis algorithms are developed. We then apply the approach to enforce the properties of the simulated robotic swarm in the robotflocksim platform. We evaluate and show the effectiveness of the method with experiments on specific unmanned aerial vehicle swarm tasks.

4. Architecture-level particular risk modeling and analysis for a cyber-physical system with AADL, by Ming-rui XIAO et al.

Cyber-physical systems (CPSs) are becoming increasingly important in safety-critical systems. Particular risk analysis (PRA) is an essential step in the safety assessment process to guarantee the quality of a system in the early phase of system development. Human factors like the physical environment are the most important part of particular risk assessment. Therefore, it is necessary to analyze the safety of the system considering human factor and physical factor. In this paper, we propose a new particular risk model (PRM) to improve the modeling ability of the Architecture Analysis and Design Language (AADL). An architecture-based PRA method is presented to support safety assessment for the AADL model of a cyber-physical system. To simulate the PRM with the proposed PRA method, model transformation from PRM to a deterministic and stochastic Petri net model is implemented. Finally, a case study on the power grid system of CPS is modeled and analyzed using the proposed method.

Contributors

The challenges and research problems in this paper are formulated through long-time discussions and exchanges of ideas between Zhiming LIU and Ji WANG. The research directions are proposed following their personal communications. The paper is typed by Zhiming LIU and checked by Ji WANG.

Acknowledgements

We would like to thank all the authors who made significant effort to submit their work for this special section and the authors of the four papers included in this special section for their effort in revising their manuscripts. Our many thanks go to the reviewers for their dedicated work. We also thank Prof. Jonathan BOWEN, Prof. Shmuel TYSZBEROWICZ, and Mr. Guisen WU for their reading and comments on early versions of this article and the reviewers of this perspective for their comments. Parts of the discussion in this perspective are based on the application of the Chinese National Natural Science Foundation Key Project "Theory of Modelling and Software Defined Method for Systems of Human-Cyber-Physical Computing". Zhiming LIU, as the coordinator of the project, would like to thank his project colleagues Prof. Wei DONG and Dr. Wan-wei LIU of the National University of Defence Technology, Profs. Miao-miao ZHANG and Guan-jun LIU of Tongji University, and Drs. Heng-jun ZHAO, Bo LIU, and Yuan-rui ZHANG of RISE at Southwest University for the fruitful collaboration and hard work, and Drs. Bin GU and Yu JIANG for their great supports to the preparation for the project application. Zhiming LIU would also like to express his appreciation to his students Quan SUN, Huan TU, Gui-sen WU, Wei ZHANG, Ting-ting ZHANG, Wei ZHAO, et al. for their help.

Compliance with ethics guidelines

Zhiming LIU and Ji WANG declare that they have no conflict of interest.

References

Alur R, Courcoubetis C, Halbwachs N, et al., 1995. The algorithmic analysis of hybrid systems. *Theor Comput Sci*, 138(1):3-34. https://doi.org/10.1016/0304-3975(94)00202-T

Baheti R, Gill H, 2011. Cyber-physical systems. *Impact Contr Technol*, 12(1):161-166.

Banerjee A, Venkatasubramanian KK, Mukherjee T, et al., 2012. Ensuring safety, security, and sustainability of mission-critical cyber-physical systems. *Proc IEEE*, 100(1):283-299. https://doi.org/10.1109/JPROC.2011.2165689

Broy M, Cengarle MV, Geisberger E, 2012. Cyber-physical systems: imminent challenges. Proc 17th Monterey Workshop, p.1-28. https://doi.org/10.1007/978-3-642-34059-8_1

Calvaresi D, Marinoni M, Sturm A, et al., 2017. The challenge of real-time multi-agent systems for enabling IoT and CPS. Proc Int Conf on Web Intelligence, p.356-364. https://doi.org/10.1145/3106426.3106518

Chen D, Doumeingts G, Vernadat F, 2008. Architectures for enterprise integration and interoperability: past, present and future. *Comput Ind*, 59(7):647-659. https://doi.org/10.1016/j.compind.2007.12.016

Chen X, Liu Z, 2017. Towards interface-driven design of evolving component-based architectures. In: Hinchey M, Bowen JP, Olderog ER (Eds.), Provably Correct Systems. Springer, Cham, p.121-148. https://doi.org/10.1007/978-3-319-48628-4_6

Darabseh A, Al-Ayyoub M, Jararweh Y, et al., 2015. SDStorage: a software defined storage experimental framework. IEEE Int Conf on Cloud Engineering, p.341-346. https://doi.org/10.1109/IC2E.2015.60

Dressler F, 2018. Cyber physical social systems: towards deeply integrated hybridized systems. Proc Int Conf on Computing, Networking and Communications, p.420-424. https://doi.org/10.1109/ICCNC.2018.8390404

Gill H, 2010. Cyber-physical systems: beyond ES, SNs, SCADA. Proc Trusted Computing in Embedded Systems Workshop.

Giloi WK, 1997. Konrad Zuse's Plankalkül: the first high-level, "non von Neumann" programming language. *IEEE Ann History Comput*, 19(2):17-24. https://doi.org/10.1109/85.586068

Gunes V, Peter S, Givargis T, et al., 2014. A survey on concepts, applications, and challenges in cyber-physical systems. *KSII Trans Intern Inform Syst*, 8(12):4242-4268. https://doi.org/10.3837/tiis.2014.12.001

He JF, 1994. From CSP to hybrid systems. In: Roscoe AW (Ed.), A Classical Mind: Essays in Honour of C. A. R. Hoare. Prentice Hall International, New York, USA, p.171-189.

Hoare CAR, 1985. Communicating Sequential Processes. Prentice Hall International, Englewood Cliffs, USA.

Hu J, Liu Z, Reed GM, et al., 2008. Ensemble engineering and emergence. In: Wirsing M, Banâtre JP, Hölzl M, et al. (Eds.), Software-Intensive Systems and New Computing Paradigms: Challenges and Visions. Springer Berlin Heidelberg, p.162-178. https://doi.org/10.1007/978-3-540-89437-7]_11

IEEE, 1990. IEEE Standard Computer Dictionary: a Compilation of IEEE Standard Computer Glossaries. IEEE, New York, USA.

Jararweh Y, Al-Ayyoub M, Darabseh A, et al., 2015. SDIoT: a software defined based Internet of Things framework. *J Amb Intell Human Comput*, 6(4):453-461. https://doi.org/10.1007/s12652-015-0290-y

Jararweh Y, Al-Ayyoub M, Ala'Darabseh, et al., 2016. Software defined cloud: survey, system and evaluation. *Fut Gener Comput Syst*, 58:56-74. https://doi.org/10.1016/j.future.2015.10.015

Khaitan SK, McCalley JD, 2015. Design techniques and applications of cyberphysical systems: a survey. *IEEE Syst J*, 9(2):350-365. https://doi.org/10.1109/JSYST.2014.2322503

Kindberg T, Fox A, 2002. System software for ubiquitous computing. *IEEE Perv Comput*, 1(1):70-81. https://doi.org/10.1109/MPRV.2002.993146

Kubicek H, Cimander R, Scholl HJ, 2011. Layers of interoperability. In: Kubicek H, Cimander R, Scholl HJ (Eds.), Organizational Interoperability in E-Government. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-22502-4_7

Lee EA, 2006. Cyber-physical systems—are computing foundations adequate? NSF Workshop on Cyber-Physical Systems.

Lee EA, 2008. Cyber physical systems: design challenges. Proc 11th IEEE Int Symp on Object and Component-Oriented Real-Time Distributed Computing, p.363-369. https://doi.org/10.1109/ISORC.2008.25

Lee EA, 2010. CPS foundations. Proc 47th Design Automation Conf, p.737-742. https://doi.org/10.1145/1837274.1837462

Lee EA, Neuendorffer S, Wirthlin MJ, 2003. Actor-oriented design of embedded hardware and software systems. *J Circ Syst Comput*, 12(3):231-260. https://doi.org/10.1142/S0218126603000751

Lee J, Lapira E, Bagheri B, et al., 2013. Recent advances and trends in predictive manufacturing systems in big data environment. *Manuf Lett*, 1(1):38-41. https://doi.org/10.1016/j.mfglet.2013.09.005

Lindsey CH, Boom HJ, 1978. A modules and separate compilation facility for ALGOL 68. *ALGOL Bull*, (43):19-53.

Liskov B, Zilles S, 1974. Programming with abstract data types. Proc ACM SIGPLAN Symp on Very High Level Languages, p.50-59. https://doi.org/10.1145/800233.807045

Liu Z, Chen XH, 2014. Model-driven design of object and component systems. Proc 1st Int School Engineering Trustworthy Software Systems, p.152-255. https://doi.org/10.1007/978-3-319-29628-9_4

Liu Z, Joseph M, 1996. Verification of fault tolerance and real time. Proc 26th Annual Int Symp on Fault-Tolerant Computing, p.220-229.

Liu Z, Joseph M, 1999. Specification and verification of fault-tolerance, timing, and scheduling. *ACM Trans Program Lang Syst*, 21(1):46-89. https://doi.org/10.1145/314602.314605

Liu Z, Ravn AP, Li X, 1998. Verifying duration properties of timed transition systems. Proc IFIP TC2/WG2.2, 2.3 Int Conf on Programming Concepts and Methods, p.327-345. https://doi.org/10.1007/978-0-387-35358-6_22

Liu Z, Morisset C, Stolz V, 2008. A component-based access control monitor. Proc 3rd Int Symp on Leveraging Applications of Formal Methods, Verification and Validation, p.339-353.

Liu Z, Bowen JP, Liu B, et al., 2019. Software abstractions and human-cyber-physical systems architecture modelling. Proc 5th Int School Engineering Trustworthy Software Systems, p.159-219. https://doi.org/10.1007/978-3-030-55089-9_5

Lu YJ, Cecil J, 2015. An Internet of Things (IoT) based cyber physical framework for advanced manufacturing. Proc Move to Meaningful Internet Systems: OTM 2015 Workshops, p.66-74. https://doi.org/10.1007/978-3-319-26138-6_10

Lynch N, Segala R, Vaandrager F, 2003. Hybrid I/O automata. *Inform Comput*, 185(1):103-157. https://doi.org/10.1016/S0890-5401(03)00067-1

Lynch NA, 1996. Distributed Algorithms. Morgan Kaufmann Publishers, San Francisco, USA.

Mei H, Guo Y, 2018. Toward ubiquitous operating systems: a software-defined perspective. *Computer*, 51(1):50-56. https://doi.org/10.1109/MC.2018.1151018

Milner R, 1989. Communication and Concurrency. Prentice Hall International, New York, USA.

Molina E, Jacob E, 2018. Software-defined networking in cyber-physical systems: a survey. *Comput Electr Eng*, 66:407-419. https://doi.org/10.1016/j.compeleceng.2017.05.013

NSF, 2006. NSF Workshop on Cyber-Physical Systems. Austin, Texas, USA.

Nygaard K, Dahl OJ, 1978. The development of the SIMULA languages. *ACM SIGPLAN Not*, 13(8):245-272. https://doi.org/10.1145/960118.808391

Palomar E, Chen XH, Liu Z, et al., 2016. Component-based modelling for scalable smart city systems interoperability: a case study on integrating energy demand response

systems. *Sensors*, 16(11):1810. https://doi.org/10.3390/s16111810

Parnas DL, 1972. On the criteria to be used in decomposing systems into modules. *Commun ACM*, 15(12):1053-1058. https://doi.org/10.1145/361598.361623

Rajkumar R, Lee I, Sha L, et al., 2010. Cyber-physical systems: the next computing revolution. Proc 47th Design Automation Conf, p.731-736. https://doi.org/10.1145/1837274.1837461

Romero D, Bernus P, Noran O, et al., 2016. The operator 4.0: human cyber-physical systems & adaptive automation towards human-automation symbiosis work systems. Proc IFIP WG 5.7 Int Conf on Advances in Production Management Systems, p.677-686. https://doi.org/10.1007/978-3-319-51133-7_80

Sangiovanni-Vincentelli A, Damm W, Passerone R, 2012. Taming Dr. Frankenstein: contract-based design for cyber-physical systems. *Eur J Contr*, 18(3):217-238. https://doi.org/10.3166/ejc.18.217-238

Schätz B, 2016. Platforms for cyber-physical systems—fractal operating system and integrated development environment for the physical world. Proc 3rd Int Workshop on Emerging Ideas and Trends in Engineering of Cyber-Physical Systems, p.1-4. https://doi.org/10.1109/EITEC.2016.7503688

Sha L, Gopalakrishnan S, Liu X, et al., 2008. Cyber-physical systems: a new frontier. Proc IEEE Int Conf on Sensor Networks, Ubiquitous, and Trustworthy Computing, p.1-9. https://doi.org/10.1109/SUTC.2008.85

Sheth A, Anantharam P, Henson C, 2013. Physical-cyber-social computing: an early 21st century approach. *IEEE Intell Syst*, 28(1):78-82. https://doi.org/10.1109/MIS.2013.20

Sowe SK, Simmon E, Zettsu K, et al., 2016. Cyber-physical-human systems: putting people in the loop. *IT Prof*, 18(1):10-13. https://doi.org/10.1109/MITP.2016.14

Tan Y, Vuran MC, Goddard S, 2009. Spatio-temporal event model for cyber-physical systems. Proc 29th IEEE Int Conf on Distributed Computing Systems Workshops, p.44-50. https://doi.org/10.1109/ICDCSW.2009.82

Tröger P, Werner M, Richling J, 2015. Cyber-physical operating systems—what are the right abstractions? Proc 4th Mediterranean Conf on Embedded Computing, p.13-16. https://doi.org/10.1109/MECO.2015.7181874

Wang SL, Zhan NJ, Zou L, 2015. An improved HHL prover: an interactive theorem prover for hybrid systems. Proc 17th Int Conf on Formal Engineering Methods, p.382-399. https://doi.org/10.1007/978-3-319-25423-4_25

Weiser M, 1991. The computer for the 21st century. *Sci Am*, 265(3):94-104.

Wheeler DJ, 1952. The use of sub-routines in programmes. Proc ACM National Meeting, p.235. https://doi.org/10.1145/609784.609816

Wilkes MV, Wheeler DJ, Gill S, 1951. The Preparation of Programs for an Electronic Digital Computer. Addison-Wesley, Wokingham, UK.

Wirsing M, Banatre JP, Hölzl M, et al., 2008. Software-Intensive Systems and New Computing Paradigms—Challenges and Visions. Springer Berlin Heidelberg.

Xu LD, Duan L, 2019. Big data for cyber physical systems in Industry 4.0: a survey. *Enterp Inform Syst*, 13(2):148-169. https://doi.org/10.1080/17517575.2018.1442934

Zegzhda DP, 2016. Sustainability as a criterion for information security in cyber-physical systems. *Autom Contr Comput Sci*, 50(8):813-819. https://doi.org/10.3103/S0146411616080253

Zeng DZ, Gu L, Pan SL, et al., 2020. Software Defined Systems: Sensing, Communication and Computation. Springer, Cham, Germany.

Zeng J, Yang LT, Lin M, et al., 2020. A survey: cyber-physical-social systems and their system-level design methodology. *Fut Gener Comput Syst*, 105:1028-1042. https://doi.org/10.1016/j.future.2016.06.034

Zhang MM, Liu Z, Morisset C, et al., 2009. Design and verification of fault-tolerant components. In: Butler M, Jones C, Romanovsky A, et al. (Eds.), Methods, Models and Tools for Fault Tolerance. Springer, Berlin, p.57-84. https://doi.org/10.1007/978-3-642-00867-2_4

Zhou CC, Hoare CAR, Ravn AP, 1991. A calculus of durations. *Inform Process Lett*, 40(5):269-276. https://doi.org/10.1016/0020-0190(91)90122-X

Zhou J, Zhou YH, Wang BC, et al., 2019. Human–cyber–physical systems (HCPSs) in the context of new-generation intelligent manufacturing. *Engineering*, 5(4):624-636. https://doi.org/10.1016/j.eng.2019.07.015

Zhiming LIU received his MS degree in Computing Science from Software Institute of CAS in 1988 and PhD degree in Computer Science from University of Warwick in 1991. He worked in three universities in the UK during 1988–2005 and 2013–2015, and the United Nations University - International Institute for Software Technology (Macau) during 2002–2013. In 2016 he joined Southwest University in Chongqing as a full-time professor, leading the development of the University Centre for Research and Innovation in Software Engineering (RISE). He is an editorial board member of *Frontiers of Information Technology & Electronic Engineering*. He has been working in the area of software theory and methods, and is known for work on transformational approach to fault-tolerant and real-time systems, probabilistic duration calculus for system dependability analysis, and the rCOS method for object-oriented and component-based software.

Ji WANG received his BS degree and PhD degree in Computer Science in 1987 and 1995, respectively, from the College of Computer, National University of Defense Technology, Changsha, China. He is now a full professor at the State Key Laboratory of High Performance Computing, National University of Defense Technology, China. He is an executive associate editor-in-chief of *Frontiers of Information Technology & Electronic Engineering*. His research interests include formal methods and software engineering.