Guest Editors' Introduction: Cross-Layer Design of Cyber–Physical Systems

Samarjit Chakraborty
University of North Carolina at Chapel Hill, Chapel Hill, NC, USA

Jian-Jia Chen
Technical University of Dortmund, Dortmund, Germany

Anuradha Annaswamy
Massachusetts Institute of Technology, Cambridge, MA, USA

Devendra Rai
Robert Bosch R&D, Germany

Cyber–physical system (CPS) design is characterized by modeling a computer (or cyber) system *together* with the physical system that the computer controls. This approach differs from both classical control systems design and also embedded systems design. In the former case, given properties of the physical system or the "plant" to be controlled, a control law or strategy is derived, that when applied to the plant enforces the desired behavior on it. Such control laws are described mathematically, and their implementations in software already result in a CPS, viz., the plant together with the software implementation of the controller. But traditionally, the design of the control law did not take into account all the peculiarities of software implementation—such as delays involved in computing the control input based on the state of the plant, or numerical limitations of the processor running the software, or how faithfully the software implements the mathematical description of the controller. When these issues are not accounted for when designing the controller, there might be a deviation in the behavior of the system when analyzing the mathematical description of the controller, versus what would be observed with the software implementation.

Digital Object Identifier 10.1109/MDAT.2020.3047734
Date of current version: 28 September 2021.

The need to close this *model-implementation gap* has been recognized for quite some time and has led to the theory of *computer-controlled systems* (see the book with this title by Åström and Wittenmark, Dover Publications, 2011). Here, the *control law* is designed by accounting for software-implementation issues like delays and numerical limitations. However, this is not yet a *CPS design*, because it is rather only concerned with how the physical plant is manipulated (or actuated), but not with how the *computer system* implements the law in software. In a similar vein, the area of *networked control systems* also studies how distributed controllers implemented over a wireless network are to be designed. The goal here is to mitigate the impact of variable delays and packet drops—arising from the wireless network when communicating plant states and control signals—on the performance of the closed-loop system. Again, here the focus has been on *designing the controller*, with the wireless network assumed to be *given*. On the other hand, a CPS-oriented design, which has also been subsequently studied, is concerned with *jointly* designing the controller *and* the wireless network (e.g., by considering suitable packet scheduling strategies) to optimize control performance, together with possibly other metrics like channel utilization or the power consumed by wireless sensor nodes.

Similarly, while embedded systems have long been used for implementing control tasks, such tasks have

Copublished by the IEEE CEDA, IEEE CASS, IEEE SSCS, and TTTC

almost always been considered as black boxes characterized by activation periods, execution times, and deadlines, which were used for the purpose of scheduling them. The parameters came from the control designer and the goal of the embedded systems engineer was to implement the tasks to satisfy the given periodicity and deadline constraints. Such abstractions—where the control designer did not have to consider any implementation platform issues, and the embedded systems engineer implementing the controller did not have to understand how the controller was designed—provided a clean separation of concerns. But they were associated with several disadvantages. As mentioned above, they resulted in a model-implementation gap, which even when addressed by suitably designing the controllers, did not lead to efficient implementations. Clearly, designing the controllers first without accounting for all implementation choices, *followed* by implementing them, eliminates a number of optimization options. This is because the "structure" of the controller has already been fixed by the time implementation-level optimizations are being considered. In contrast, a CPS-oriented approach achieves optimality by *jointly* considering all controller design along with platform-specific implementation options.

As is probably apparent from this discussion, *cross-layer design* is fundamental to any CPS-oriented design method. Tighter interactions between the controller design (or modeling) layer and the controller implementation layer, open up new optimization opportunities that would not be available otherwise. While computer science has thrived on the principle of separation of concerns in order to tackle design complexity, there are many recent examples where redefining the traditional abstraction layers—and following a cross-layer approach—has shown immense benefits. Although we have illustrated the CPS concept in a relatively abstract fashion using a controller and its implementation on a computer, other concrete examples of CPS abound. They arise in the context of transportation—e.g., how to dynamically control toll prices to reduce traffic congestion, how to control traffic lights, and how to dispatch electric taxis so their charging demands are aligned with the availability in the power grid. In the context of a smart home, CPS-related questions could be: when to switch on the clothes dryer depending on the electricity demands of the home and the time-varying price of electricity, or how to automatically adjust heating and the opening of motorized windows to let in the fresh air, based on the quality of a person's sleep being monitored by her Fitbit-like activity tracker. Other domains where such CPS-oriented design questions are being asked include robotics, large electrical energy storage systems, smart electricity grids, smart manufacturing or Industry 4.0, medical devices, and autonomous vehicles.

In each of these domains, what constitutes the physical "plant" that needs to be controlled, what is an appropriate model of such a plant, how should the control laws be formulated, and what is the "computer" (or the cyber component) implementing the control strategy, are all very different. As a result, answering these questions and also how models, relevant performance metrics, and optimization techniques from the different design layers should be combined to enable a *cross-layer design*, requires significant innovation. Let us go back to our original example of mathematically designing a controller and implementing it in software running on a computer. Here, the models of plants and controllers should capture their evolution in continuous time and are typically differential equations. On the other hand, the computer implementing the controller operates in discrete time, and a suitable model of it might be a state machine. Relevant metrics during the controller *design* stage are stability, settling time, or peak overshoot, whereas during the *implementation* stage they might be the utilization of the processor or a communication bus. Similarly, methods during controller design might be Lyapunov techniques, whereas during the implementation stage they could involve schedulability analysis techniques from the real-time systems domain. How to combine these different *models*, *metrics*, and *methods* across multiple layers to enable cross-layer design, very much depends on what these three "M"s are for each of the layers involved. And these in turn depend on the CPS application domain, such as those outlined above.

This special issue brings together eight contributed articles that illustrate how such cross-layer design for CPS could be enabled in different application settings. The first article, titled "Cross-Layer Design of Automotive Systems," by Wang et al., shows that automotive control and software design has to consider multiple abstraction layers. The authors span all the way from the vehicular network layer dealing with vehicle-to-vehicle and vehicle-to-infrastructure communication, to the automotive in-vehicle hardware and software later. Such a cross-layer design is important

not only for optimization but also to ensure functional, as well as other properties like security. The second article, "Reconfigurable Pipelined Control Systems," by Sanchez et al., shows how appropriately structuring compute-intensive sensor data processing—like those from cameras—can mitigate large sensor-to-actuator delays in feedback controllers. Here, the main technique involves a cross-layer codesign of the controller and the sensor data processing on multicore processors. The third article, titled "Cloud-Ready Acceleration of Formal Method Techniques for Cyber–Physical Systems," by Khaled and Zamani, also deals with a similar computation bottleneck issue, but that arising when synthesizing controllers from formal specifications. Here, the authors show how resorting to cloud computing might provide a solution.

The fourth article, titled "Integrating Interobject Scenarios with Intraobject Statecharts for Developing Reactive Systems," by Harel et al., deals with an important topic of how to reconcile the differences between a model and an implementation. As we discussed earlier, this is a central problem in cross-layer design of CPS stemming from the different models and concerns at the design and the implementation stages. This article shows that it is, however, possible to have a single tool and method to support both requirement specifications and implementation. A common method and tool for both the phases results in a smooth transition between them while ensuring semantic consistency. The fifth article—at least at an abstract level—deals with the same problem of ensuring consistency between controller design and its implementation. Toward this, "Breaking Silos to Guarantee Control Stability with Communication over Ethernet TSN," by Mahfouzi et al., presents a codesign approach to implement distributed controllers on an Ethernet network.

The next or sixth article by de Chamisso et al., titled "Lifelong Exploratory Navigation: An Architecture for Safer Mobile Robots," proposes a layered architecture for robotics, with contract-based interfaces between these layers. The different layers of such an architecture are capable of autonomous adaptation in response to new environments. The next article, "Hardware Virtualization and Task Allocation for Plug-and-Play Automotive Systems," by Lin et al., addresses the complexity of automotive in-vehicle architectures by proposing a hardware virtualization layer using an OS hypervisor. This not only enables hardware-independent software development, but also allows plug-and-play features. Finally, "The AXIOM Project: IoT on Heterogeneous Embedded Platforms," by Filgueras et al., describes an IoT node that is low-power while still offering high performance. It describes the hardware architecture of this node, along with its software development flow, and two use cases.

These eight articles cover a diverse range of topics on the cross-layer design of CPS. We believe that the readers will find them interesting and gain new insights into this evolving area. We thank all those who submitted their research on this special issue. We also thank all the reviewers, the EiC, Jörg Henkel, and Sara Dailey, without whose help this special issue would not have been possible. ■

Samarjit Chakraborty is a William R. Kenan, Jr. Distinguished Professor with the Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA. His research interests include all aspects of designing hardware and software for embedded systems. Chakraborty has a PhD from ETH Zürich, Switzerland (2003).

Jian-Jia Chen is a Professor with the Department of Informatics, Technical University of Dortmund, Germany. His research interests include real-time systems, embedded systems, power/energy-aware designs, and distributed computing. Chen has a PhD from National Taiwan University, Taipei, Taiwan.

Anuradha Annaswamy is the Founder and the Director of the Active-Adaptive Control Laboratory with the Department of Mechanical Engineering, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, and is currently serving as the President of Control Systems Society (CSS). Her research interests span adaptive control theory and its applications to aerospace, automotive, and propulsion systems as well as cyber–physical systems such as smart grids, smart cities, and smart infrastructures. She is a Fellow of IEEE and IFAC.

Devendra Rai works at Bosch, Germany. His research interests include software architectures for automotive embedded systems, and in-vehicle architectures focusing on multi- and many-core processors. Rai has a PhD from ETH Zürich, Switzerland (2015).

■ Direct questions and comments about this article to Samarjit Chakraborty, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA; samarjit@cs.unc.edu.