Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 11 (2017) 989 – 995

Procedia MANUFACTURING

27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy

From the Internet of Things to Cyber-Physical Systems: the Holonic Perspective

Luca Pazzi*, Marcello Pellicciari

University of Modena and Reggio Emilia, DIEF, Via Pietro Vivarelli 10, I-41125, Modena, Italy

Abstract

The paper presents a distributed model for implementing Cyber-Physical Systems aimed at controlling physical entities through the Internet of Things. The model tames the inherent complexity of the task by a recursive notion of modularity which makes each module both a controller and a controlled entity. Modules are arranged along part-whole tree-like hierarchies which collectively constitute the system. The behaviour of each module is strictly local since it has visibility only on its controlled modules, but not on the module which controls it. Each behaviour can be thus checked locally at design time against safety and liveness formulas, which still hold when component holons are composed into more complex ones, thus contributing, without the need of additional checks, to the overall safety and liveness of the final system.

© 2017 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing

Keywords: Smart factories; Industry 4.0; Internet of Things; Cyber-Physical Systems; Holons; State-based Control; Safety engineering;

1. Introduction

The Internet of Things (IoT) is a growing networking infrastructure which is increasingly connecting computer-augmented physical systems worldwide, singularly known as Embedded Systems. Embedded Systems are traditionally aimed at the control of a single phenomenon by sensors and actuators and are often committed to a small number of related physical objects, usually performing a simple task or application by a single controller. Cyber-physical systems (CPS) can be seen as an extension of Embedded Systems by the availability of ubiquitous connectivity which can be used to build applications at the macro level (towns, villages, rural areas) as well as at the micro level (single houses or factories, the latter with emphasis on smart manufacturing and robotics), possibly integrating them. Useful applications at both levels include, but are not limited to, monitoring complex real-world phenomena and smart manufacturing. Cyber-physical systems are moreover required to exhibit a high degree of autonomy and adaptation [1][2][3] ideal for industry 4.0 applications.

The considerable success of Embedded Systems in monitoring and controlling physical processes induces a “false sense of confidence” [4], since they can be seen as black boxes, designed once and immutable. IoT-based cyber-physical systems will be instead larger and often evolving networked systems, which require integration of computation and control of physical processes at different abstraction levels. Communicating distributed nodes will have to possess computational autonomy but, at the same

* Corresponding author.
E-mail address: luca.pazzi@unimore.it

2351-9789 © 2017 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
Peer-review under responsibility of the scientific committee of the 27th International Conference on Flexible Automation and Intelligent Manufacturing
doi:10.1016/j.promfg.2017.07.204

time, be dynamically composable into more complex and scalable applications [5]. Other challenges of IoT-based CPS regard safety and reliability requirements, qualitatively different from traditional Embedded Systems [6], [7].

Such systems have to be robust and adaptive [8], being built out of potentially unreliable components, typically by encapsulating redundancy and failure policies at different levels of abstraction [9], since *the more control on safety we add, the more complexity we have to manage when dealing with pure physical control* [10]. Finally, general-purpose networking techniques themselves make global system behaviour exponentially complex and unpredictable as the number of node grows.

Introducing modularity and hierarchy in control theory poses many challenges [11], mainly in decomposing feedback loops and in maintaining distributed invariants [12]. At the same time, each module should possibly maintain an ongoing interaction with both controlled entities, humans and other non-foreseeable modules to be composed in the future, while, as observed, performing safety and reliability tasks. Modular communication and composition is therefore central for an effective theory of IoT-based CPS. What we need is therefore a modular, composable, interactive, correct and safe model of computation. Such a model should possibly extend the interactive paradigm in order to allow inter-modular communication. Composition should be moreover hierarchical, with some modules controlling other modules, in order to defeat complexity.

2. The holonic paradigm

We propose to structure IoT-based cyber-physical systems through *holons* [13], modular communicating units, arranged in part-whole hierarchies, which host a behavior, accessible through an interface, by which the holon plays recursively and at the same time the *twofold and complementary* role of *part* and *whole*:

(1) as *part*, the interface allows the behavior of the holon to be controlled remotely by other holons having it as component, through directed command events; at the same time the behavior of the holon emits notification events, through the interface, towards other holons acting as whole and having the current holon as part;
(2) as *whole*, the behavior of the holon controls remotely the behavior of other component holons by emitting command events; at the same time the behavior of the holon may be influenced by the notification events received by the interface of such component holons.

Each holon hosts a single behavior which plays the two roles at the same time. The role of whole is played directly through the complete specification of such a behavior, while the interface hides implementation details. Moreover, *the two roles are not symmetrical*, since the whole may totally control its components, but the parts do not control the whole. Parts simply emit notification events towards the whole, whose behavior specifies a reaction for each of such events. For example, an automated car acting as whole may command the opening of an internal door acting as part, but the part (the doors' subsystem) is not allowed to command any portion of the behavior of the whole in case either of success or failure in opening. Rather, the part emits either a success or a failure event towards the whole, which in turn reacts to it by triggering a specific behavior for each case. The rationale behind such an asymmetry is that the whole knows its parts through their interface, *but the parts totally ignore the whole to which they belong*. This in order to preserve self-containment and reusability in multiple design and implementation contexts.

In the paper, we adopt the state-based approach due to its generality and expressiveness. Besides the architectural aspects that we borrow from the theory of holons, we use a Statecharts variant, named Part-Whole Statecharts [14] that is strongly committed to the part-whole approach. Such a state-based language allows to specify behavior at multiple hierarchical levels by a "correct by construction" method.

By such holonic approach, the interface of each holon is a state automaton which has a set of states and state transitions. State transitions may be triggered through the interface or may happen autonomously. Two sets of events are provided by the interface, named *input* and *output* events which label, respectively, *triggerable* and *autonomous* state transitions. Input events are received by the holon and trigger the transitions to which they are associated to, output events are instead produced by the holon when it takes the autonomous transitions to which they are related. Said otherwise, the interface depicts the observable and prescribable behavior of the holon. Finally, input and output events are referred to as *external* features, since they describe the holon's external behavior.

Example: Figure 1 shows an holon's interface consisting of three states, Stop, Go and FailSafe and the state transitions linking them. Autonomous and triggerable transitions are depicted by arrows and can be distinguished since the formers starts with a white dot. Incoming arrow t_1 indicates state Stop as the initial state. Transition t_3 may be triggered by the holon receiving event go (we adopt the graphical notation of underlining trigger events), while t_4 may happen autonomously and emits the event fail_stop. We have therefore that go is an input event to the holon, while fail_stop is an output event.

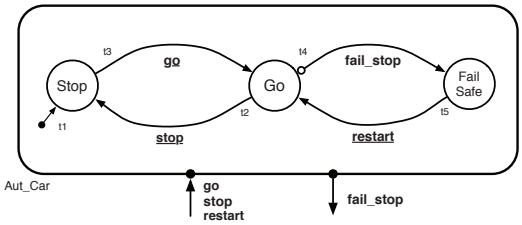

Fig. 1. The holon interface S of an autonomous railway car from [14]. Events stop, go and restart are named input events and label triggerable transitions t_2, t_3 and t_5, while fail_stop is an output event and is emitted by autonomous transition t_4.

Definition 2.1. The *interface* associated to holon h is a state machine consisting of a set of states, amongst which a set of initial states are distinguished, a set of input and output events, and of a transition function which determines, on the arrival of an input event, the next state reached by the automaton and the event which is sent outside.

Definition 2.2. An *assemblage* A(h) is a set of holons' interfaces which play the role of parts with respect to a single whole holon h acting as whole. Let A(h) = {h_1, h_2 . . . h_N} be the set of holons acting as parts with respect to h acting as whole. Each holon within an assemblage is univocally distinguishable through an identifier. The assemblage A defines a set of *named* input and output events, given by the input and output events belonging to the holon h in the assemblage, prefixed by their identifier and joined together. Finally, a *state configuration* of assemblage A(h) is a tuple of states each belonging to one of the state machines in the assemblage.

The behavior of the holon acting as whole, will be referred to either as the *complete* behavior of the holon or as the *implementation* of the interface, since it describes how the behavior of the interface is made available through the behaviors of other holons. Some interfaces however do not possess any implementation, since they are direct interfaces to the object or the phenomenon being controlled, either by sensing it or by acting upon it. In that case the assemblage controlled by the holon is empty.

By the approach, *implementing* an interface consists in adding constructs to its state diagram. Such constructs are named *internal* since they denote the holon's *hidden* (i.e. not shown by the interface) internal behavior. Conversely, removing such constructs from an implementation allows to obtain its interface.

Definition 2.3. Let h be an holon, and let A(h) be an assemblage of part holons; the holon's internal behavior is given by its interface state machine (Definition 2.1) extended by adding the following internal constructs to its state transitions:

(1) a *guard*, written [C], where C is a boolean valued proposition pertaining the state of one of the holons in the assemblage of parts of the current holon;
(2) a *list of command events*, where each command is an input events of the assemblage;
(3) an *internal trigger*, written **c.e**, associated to an autonomous transition, where c.e belongs to the input events of the assemblage.

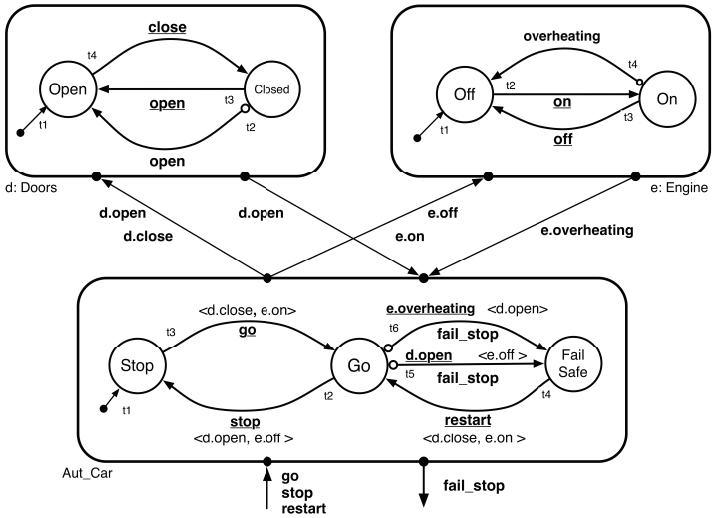

Fig. 2. The *complete* behaviour and the assemblage of components of the holon Aut_Car modelling an autonomous railway car [13] whose interface is shown in Figure 1.

Figure 2 shows the complete behavior and related assemblage of the holon Aut_Car whose interface is shown in Figure 1. The assemblage consists of the interfaces of two components, Doors and Engine. In other words, holon Aut_Car controls, through its implementation, the joint behavior of holons Doors and Engine. By comparing Figure 1 and 2, it can be observed, that the interface of the holon can be extracted automatically from a fully specified behaviour by simply hiding both its internal constructs and component holons. Observe that transitions t_5 and t_6 which emit the same event fail_stop in the implementation become a single transition in the interface. Transition t_3 is said *externally triggered*, i.e., it can be triggered by a generic holon (not specified yet) having the current holon as component. The arrival of the event go triggers transition t_5 and forwards as commands both events close to holon d (written d.close) and to holon e (written e.on). The external autonomous behavior of transition t_5, is explained by the fact that it is triggered by the internal notification event open from holon d: it is indeed said *internally triggered*. At the same time the state transition emits event off as command to holon e and event fail_stop as notification towards a generic and yet unspecified holon. In this case the transition implements a safety feature in the behaviour, that is *the car cannot travel with doors open*. Transition t_6 implements a similar mechanism.

3. Communication and synchronization

State machines within holons *operate* through an internal, never ending cycle, which iterates a basic computation step. During a computation step, signals sent to the machine are evaluated, and one, if any, state transition is chosen for execution. A state transition execution consists therefore in computing a new current state and in exchanging signals with other state machines. As observed in the paper, this process is not symmetrical. Two different transition patterns are in fact necessary and need to be implemented together with four kinds of communication ports (Figure 3), as shown in Section 3.1.

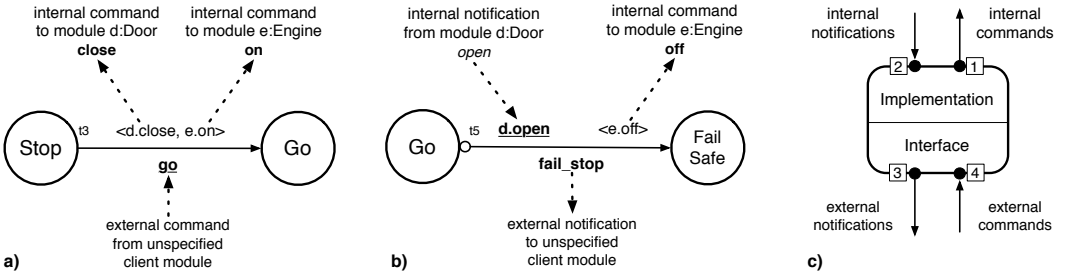

Fig. 3. Transition patterns for regular (a) and autonomous (b) transitions in holon Aut_Car. Four typologies of communication ports are therefore needed (c) in order to account for internal/external commands/notifications.

In both cases, state machines *communicate* through some *communication medium* (CM), which again operates through one or more never ending cycles. Such cycles iterate basic communication computations, which consist, essentially, in delivering signals from one machine to another. The control model consists thus of three kinds of entities: the state machine acting as whole, referred to as *controller* in the rest of this section, the *controlled* interface machines within the component assemblage, and the communication medium connecting the state machines. Two very general models of synchrony may in first place be foreseen.

(1) In the *asynchronous model,* each machine is driven by a separate thread or by a separate processor. The communication medium is again driven by one or more threads or processors. The entities of the control model are therefore behaviourally independent and synchronize only through *communication ports*. A communication port is a block of memory which is shared among the different processes. The processes read and write control signals by a typical producer consumer pattern of execution. To prevent processes from reading or writing the shared data at the same time, one or more mutex or read-write locks are employed. Finally, the shared block of memory can be structured as a FIFO list, in order to have the producer not to stop in case a new control signal is produced before a previously produced control message has been consumed.

(2) In the *synchronous model,* both the controller and the controlled state machines, as well as the communication medium, are driven by a *unique* thread and execute in its main cycle, or through different processors and some sort of time-driven, or master-slave synchronisation is implemented.

The control model will be able not only to provide both kinds of synchronisation, but also to host a mix of them. In other words, given a controller state machine and a set of controlled state machines, it may be the case that some machines in the set are controlled through the asynchronous model, and the others through the synchronous one. As an example, the same computer processor may control some machines asynchronously through a field bus and, at the same time, control other internal state machines synchronously, like timers or adders, by the internal motherboard communication bus.

Control signals are used in order to coordinate the joint behaviour of the assemblage and of the controller (Section 4.1). Control signals are generated either by one of the assemblage components or by the controller, and processed by the

communication medium through the communication ports described in Section 4.2. Figure 4 depicts the two main stages of the process.

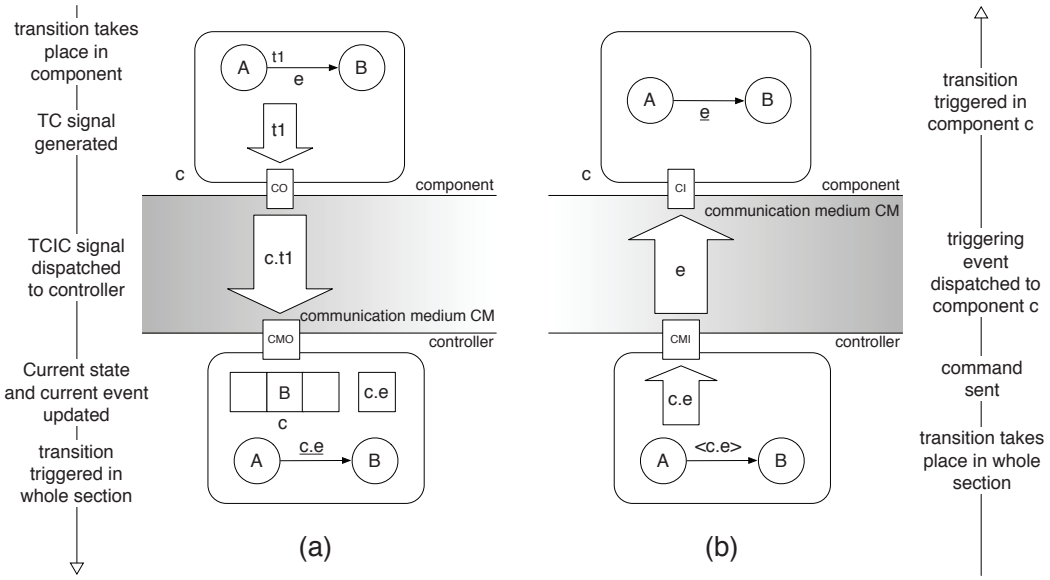

Fig. 4. Communication flow: component-controller (a) and controller-component (b) communication.

4. From the IoT to CPS

The Internet of Things already provides nodes which are connected to real-world objects, which can be either sensed or acted (or both) through an interface, let them be named P-nodes (Physical-nodes). We propose to add additional units of computations, called C-nodes (Cyber-nodes). By arranging them along holonic partonomies we obtain Cyber-Physical systems.

Each holon implements either P-nodes or C-nodes:

1. P-nodes are either:
   a. *sensors*, that is IoT nodes which are able to detect one or more aspects of a physical object or phenomenon. Such a detection happens through autonomous transitions in the interface associated to the objects;
   b. *actuators*, that is IoT nodes which are able to modify one or more aspects of a physical object or phenomenon. Such a modification happens through autonomous transitions in the interface associated to the objects;
2. C-nodes are essentially *controllers*, that is devices able to receive and send information to both C-nodes and P-nodes, that is sensors, actuators and controllers.

As observed, holons may not possess an implementation, in which case sensors and actuators are directly interfaced with physical objects and give rise to P-nodes. Sensed variations in the object or physical phenomenon result in autonomous transitions being taken by the corresponding holon. Triggered transitions in the interface perform instead changes in the object or influence the physical phenomenon. The distinction between sensors and actuators P-nodes may be blurred by considering that the same object may host both behaviors. For example, holon `Doors` in Figure 2 possesses triggerable transitions t_3 and t_4 which are taken once the input events `open/close` are received by the holon, resulting in the corresponding hydraulic operation of the doors. Manual opening of the doors for safety reasons is also possible, and results in the transition t_5 taken autonomously by the holon and in the corresponding output event open notified to the whole `Aut_Car`.

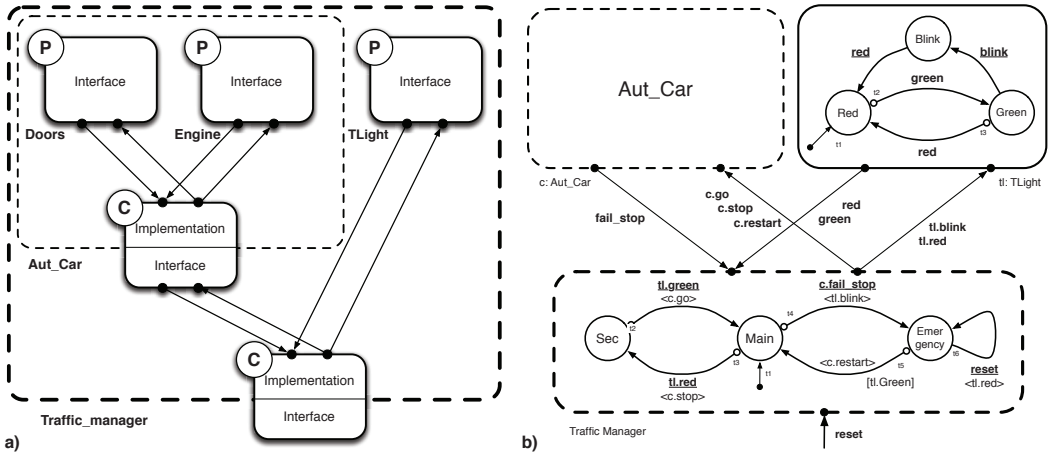

Fig. 5. IoT-based Cyber-Physical Systems (IoT-CPS) may be assembled from other already assembled IoT-CPSs and single holon modules, with P- and C-nodes identified by circled P and C (a). The complete behaviour of the IoT-CPS Traffic Manager, composed by the IoT-CPS Aut_Car and by the P-node TLight is shown in (b).

5. Conclusions

The interlinking of the physical and the cyber worlds by the Internet of Things requires to address the huge complexity that comes from the interaction of both worlds, that is of billions of computing, sensing, actuating and communicating nodes worldwide. Partitioning the network structure by smaller and flat substructures is a partial solution which only alleviates the problem: a more radical and organic solution is instead needed and must be sought by following real world structure more closely as possible. Since real world is naturally structured and connected, the idea is to understand carefully its organising principles in order to shape distributed cyber-physical systems accordingly.

In first place the world is not flat, that is each natural entity or artefact exhibits an internal structure, often composed by other components entities. Such a structure may be ignored for some tasks, but it may also be exploited for others. At the same time, each entity or artefact exhibits a set of features which act as an interface allowing to interact with it as a whole. The entity itself, by its internal mechanisms, coordinates the behaviour of its internal components and may, recursively, be component of larger entities. Albeit the global behaviour is very complex, at each level is locally rather simple and cognitively manageable.

In the paper, we modelled cyber-physical system by such a natural, part/whole, organisation. At each level, each subsystem is an autonomous entity named holon, since it may be both a part and a whole, from which the Janus characterisation after Arthur Koestler's intuition. The vertical and recursive composition of holons, each whole side matching the part side of one or more components, is called holarchy.

We then adapted state behaviours in such a way that each module may both implement a behaviour and be, at the same time, part of a more complex behaviour itself. This requires behaviours at the different levels to cooperate by exchanging semantically qualified event events, since they have to carry behavioural information at the different levels. Four ports are therefore needed in order to allow bidirectional communication towards higher and lower holons in the holarchy. Modules at the top implement only two ports, and host physical sensors and actuators.

It is finally possible to view each holon as the synchronization structure upon which model checking of temporal logic formulas is performed. In this way, model checking may be partitioned into low-complexity automata. The overall composition rules given in the paper allow to compose each holon into more complex hierarchies without having to recheck it once composed. In this way, we give safety and liveness rules a strict locality, encapsulating their validity within each module. We believe such a feature may be of paramount importance, given the safety critical tasks in which most cyber- physical system will be employed.

6. Acknowledgments

This paper is supported by European Union's Horizon 2020 research and innovation program under grant agreement No. 688807, project ColRobot (Collaborative Robotics for Assembly and Kitting in Smart manufacturing).

References

[1] Gervasio Varela. 2013. Autonomous Adaptation of User Interfaces to Support Mobility in Ambient Intelligence Systems. In *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '13)*. ACM, New York, NY, USA, 179–182.

[2] Albert Esterline, Chafic BouSaba, Barbara Pioro, and Abdollah Homaifar. 2006. Hierarchies, Holons, and Agent Coordination. In Proceedings of the Second International Conference on Radical Agent Concepts: Innovative Concepts for Autonomic and Agent-Based Systems (WRAC'05). Springer-Verlag, Berlin, Heidelberg, 210–221.

[3] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. 2015. A Survey on Engineering Approaches for Self-Adaptive Systems. *Pervasive Mob. Comput.* 17, PB (Feb. 2015), 184–206.

[4] Edward A. Lee. 2010. CPS Foundations. In *Proc. of the 47th Design Automation Conference (DAC)*. ACM, 737–742.

[5] Henry Muccini, Mohammad Sharaf, and Danny Weyns. 2016. Self-adaptation for Cyber-Physical Systems: A Systematic Literature Review. In *Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '16)*. ACM, New York, NY, USA, 75–81.

[6] Arvind Easwaran and Insup Lee. 2008. Compositional Schedulability Analysis for Cyber-Physical Systems. *SIGBED Rev.* 5, 1, (Jan. 2008).

[7] Jin Heo and Tarek Abdelzaher. 2009. AdaptGuard: Guarding Adaptive Systems from Instability. In *Proceedings of the 6th International Conference on Autonomic Computing (ICAC '09)*. ACM, New York, NY, USA, 77–86.

[8] Joerg Henkel and Lars Bauer. 2010. What is Adaptive Computing? SIGDA Newsl. 40, 5 (May 2010), 1–1.

[9] Tomas Bures, Danny Weyns, Christian Berger, Stefan Biffl, Marian Daun, Thomas Gabor, David Garlan, Ilias Gerostathopoulos, Christine Julien, Filip Krikava, Richard Mordinyi, and Nikos Pronios. 2015. Software Engineering for Smart Cyber-Physical Systems – Towards a Research Agenda: Report on the First International Workshop on Software Engineering for Smart CPS. *SIGSOFT Softw. Eng. Notes* 40, 6 (Nov. 2015), 28–32

[10] N.G. Leveson. 2011. Engineering a Safer World: Systems Thinking Applied to Safety. MIT PRESS.

[11] Hendrik Van Brussel, Luc Bongaorts, Jo Wyns, Paul Valckenaers, and Tony Van Ginderachter, 1999. A Conceptual Framework for Holonic Manufacturing: Identification of Manufacturing Holons, Journal of Manufacturing Systems, Elsevier, Vol. 18/No. 1.

[12] Luca Pazzi. 2015. Control Theory Meets Software Engineering: The Holonic Perspective. In *Proceedings of the 1st International Workshop on Control Theory for Software Engineering (CTSE 2015)*. ACM, New York, NY, USA, 34–41.

[13] Arthur Koestler. 1970. Beyond reductionism; new perspectives in the life sciences (Koestler and Smythies eds.). Proceedings of the Alpbach Symposium. (1970).

[14] L. Pazzi and M. Pradelli. 2012. Modularity and Part-Whole Compositionality for Computing the State Semantics of Statecharts. In Application of Concurrency to System Design (ACSD), 2012, 12th International Conference on.

[15] Luca Pazzi. 2014. Modeling Systemic Behavior by State-Based Holonic Modular Units. In Model-Driven Engineering Languages and Systems, Lecture Notes in Computer Science, Vol. 8767. Springer International Publishing, 99–115.