Creating a Foundation for Next-Generation Autonomous Systems

David Harel and Assaf Marron
Department of Computer Science and Applied Mathematics, Weizmann Institute of Science,
Rehovot 76100, Israel

Joseph Sifakis
Verimag Laboratory, Université Grenoble Alpes,
38401 Saint-Martin-d'Hères, France

Editor's notes:
This article advocates the need for a new autonomics foundation with a focus on decision-making logic and its processes for building trustworthy autonomous systems.

—Selma Saidi, TU Dortmund

Autonomous systems are already able to replace humans in carrying out a variety of functions. This trend will continue in the years to come, with autonomous systems becoming central and crucial to human society.[1]

Many organizations are already striving to develop the next wave of trustworthy, cost-effective autonomous systems. However, extremely high levels of complexity and criticality present fundamental new challenges.

Next-generation autonomous systems will be expected to operate under conditions that will often be unpredictable at the time of their development. Although test environments and simulation engines provide ever-increasing variation and realism [see CARLA (www.carla.org), HEXAGON MSC (www.hexagonmi.com/products/computer-aided-engineering-cae-software/msc-software), and COGNATA (www.cognata.com)], these are still constrained and synthetic. Engineers must be able to assure customers and regulators that the system will function correctly and safely, not only in a large variety of known critical scenarios but also in complex high-risk situations that were never even thought about previously.

There is a growing awareness that the challenges of developing next-generation autonomous systems will be difficult to accomplish due to weaknesses in established methods and processes. The combined community that consists of relevant groups in the industry, government, and academia, is in the process of launching road-mapping activities and large-scale collaborative projects, like DEEL (www.deel.ai) SAFETRANS (www.safetrans-de.org), HumanDrive (www.humandrive.co.uk), and the USA Department of Transportation Initiative: "Preparing for the future of transportation: Automated vehicles 3.0."

Still, we argue, this is not enough. The required trustworthiness mandates different, and more fundamental, advances in certain relevant fields, both for development tools and for final system implementation. To narrow the gap between the challenges in developing next-generation autonomous systems and the present state of the art, the research and industry community must construct a common engineering foundation. This foundation, which we term *Autonomics*, should address the unique challenges relevant to such systems, by providing new concepts, perspectives, and engineering principles, as well as the supporting methods and tools.

We believe that the availability of such a foundation has the potential to dramatically accelerate the

[1] This article abbreviates (and somewhat extends) our earlier paper, published in the Proceedings of the National Academy of Sciences (PNAS) [1].

Digital Object Identifier 10.1109/MDAT.2021.3069959
Date of publication: 31 March 2021; date of current version: 9 February 2022.

deployment and acceptance of high-quality, certifiable autonomous systems, built for the benefit of human society.

Next-generation autonomous systems

Preliminary definitions

Over the years, many definitions have been offered for autonomy (e.g., [2]). *Autonomic computing,* a term coined by IBM in 2001, focuses on systems capable of performing self-management, and, in particular, automating dynamic configuration. The research area of agent-based design and, in particular, *multiagent systems* offers a perspective of autonomy in paying special attention to the issue of combining local goals with collaboration rules and distributed algorithms to achieve system-wide overall goals. Autonomy is often associated with *self-awareness* (e.g., [3]), which implies the system's ability to perceive changes in the environment and use "knowledge" of its own states to react adequately so that a set of goals is achieved. *Symbiotic computing* studies how autonomous systems can interface and collaborate with humans and with complex organizations, considering the many technical, commercial, and ethical implications thereof.

To streamline the ensuing discussion, we offer, in this section and the next one, definitions for some basic concepts (see also [4] and [5]).

Systems are the artifacts that development teams are out to build. A system works within and reacts to, an external environment, and it consists of two types of components, agents and objects. The coordinated collective behavior of the system's agents and objects is designed to meet some global, system-wide goals.

Objects are those components whose programmed behavior is not affected during system development. Objects have *states*, that can be changed by agents or by other objects or can change "spontaneously," for internal reasons. These objects become part of the system's internal and external environments, respectively.

Agents are the main behavioral elements of an autonomous system. They are those designed (programmed, built) as part of the system's development process. Agents have *agency*: they are proactive and pursue specific goals which may change dynamically. Agents can monitor objects from the internal and external environments and can change their states. They can also coordinate their own actions with other agents.

The environment of a system is the collection of all entities with which the system might interact. It may include other systems (with their objects and agents) and stand-alone objects, and of course humans—with their unpredictability, rationality, initiative, and power and authority to modify system behavior.

Defining autonomous behavior

We say that a system or an agent (for simplicity, we shall stick to the system below) manifests *autonomous behavior* if it embodies the following five behavioral functions, which are carried out with little or no intervention from humans or other systems.

Two functions are combined to enable the system to build for itself a useful representation of the state of the external environment. *Perception* is the function that inputs stimuli, interprets their basic meaning, and removes ambiguity, yielding relevant information. The second function is *model update*, which uses the information provided by perception to create and constantly update an integrated runtime model representing the system's environment and its states. This model will then be used in on-going decision-making.

Two other functions constitute the system's adaptive decision process. Decisions consider many possibly conflicting goals, in a way that depends on the current state of the system and its environment. *Goal management* chooses from among the set of goals the ones that are relevant to the current state. *Planning* computes a plan to achieve the set of goals produced by goal management, subject to state-dependent constraints; this is the agent's action in response to the current environment state and may consist of a sequence of commands to be executed by actuators.

The fifth function that characterizes autonomous behavior is *self-adaptation*, which caters for dynamic adjustment over time of the system's goals and the goal management and planning processes, through learning and reasoning, based on the evolving state of the system and its environment. Such adaptivity could come in many forms: very near term, e.g., using trial and error and recent experiences, as well as "life-long learning" by the system, constantly reevaluating its entire history to better achieve its goals in a dynamic unpredictable environment.

Next-generation autonomous systems are different

Next-generation autonomous systems, both those that are already beginning to emerge and definitely those of the future, differ from existing systems in several key aspects.

They have a large variety of possibly-conflicting system goals. A typical next-generation autonomous system will not be focused on a small number of well-defined goals, such as winning a game of chess or reaching a destination without collisions. It will typically face a far wider and more elaborate set of goals, as humans often do.

Their environment is dramatically less predictable. Even autonomous systems of the present already have to deal with an enormous number of *known* environment configurations, and those we do not know about yet will obviously add a whole new order of magnitude to this difficulty. While an autonomous vehicle's handling of varying and evolving road topologies and traffic volumes and speeds can probably be addressed using existing technologies, there are more complex issues, which humans handle routinely and which are still not adequately addressed. For autonomous vehicles, these include, e.g., the whims of bicycle and motorcycle riders; police instructions, spoken or signaled; poorly marked temporary diversions; dealing with passenger medical emergencies; understanding and reacting to changes in needs and new requests of passengers; abuse by humans inside and outside the vehicle; minor conflicts and accidents; the need to assist others in distress; and more. These kinds of difficulties are caused by the increased dependency on hard-to-predict aspects of the dynamic environment, compounded by the increased mobility, distribution, and sheer multitude of systems.

They require rich interaction with humans. The issue of interaction with humans goes much deeper than classical human–computer interaction. First, because future systems will operate in common human environments, having to interface with humans over whom the owner of the autonomous system has no control. They will be interfacing with humans in wholly new ways, even as part of normal everyday routines, such as a human and a robot negotiating the right-of-way through an office door, a human pointing out a spill on the floor to a passing robotic cleaning assistant, or a human and robot apologizing or arguing about a small accident/collision or misstep.

The human–computer interface itself will have to be far more extensive than a mere display and keyboard. It will encompass much of what the autonomous system understands and does, as well as aspects of the human's perception of the autonomous system and of the environment in which it operates. Special attention has to be paid to those parts of the interface that allow a human to interrupt the operation of the autonomous system or change it abruptly, e.g., when observing that an unexpected event or activity blocks the normal, preprogrammed operation of the system.

Why a new foundation?

We call upon the research and engineering community to create and evolve a foundation for developing such systems, which will recommend engineering practices and methods, point at tools and technologies, and offer open-source bases and examples. It will also include meta-information, such as reliable means for selecting among system design and development alternatives. Although this autonomics foundation should touch upon all aspects of system engineering, it will focus on "burning" issues (such as those we discuss in the upcoming sections) and propose ways to deal with them throughout development.

The existence of gaps between the state of the art and achieving the desired trustworthiness has been articulated, e.g., in the IEEE whitepaper [6] and Neumann [7]. To reinforce our argument we shall focus here on one central aspect, which is at the very heart of autonomous system engineering—the decision-making logic and its processes. We present three partially overlapping challenges in developing decision-making processes, for which satisfactory solutions have yet to come.

Challenge I: Specifying behavior

The behavioral specification is needed in virtually all stages and activities of the development process: requirements, design, simulation, testing, verification, and validation. We argue that in the next-generation autonomous systems the very specification of behavior introduces new issues that call for extensive research.

When it comes to complex autonomous systems, the specification of even a single simple goal is hard. Assume that, we want to completely automate a floor-cleaning process. What kinds of the specification are we after? Should it be focused on actions (e.g., where and how to sweep), on environment objects and entities (e.g., what kinds of dirt should be removed, and from where), or on states and results (e.g., what should the floors and shelves look like once the job is done)? How should one tell developers (and the system) about the need to move small objects or unplug devices that are in the way, or about dealing with such risks as breaking something?

We need here ways to describe the relevant "world" and its associated behaviors. For this, we propose to develop domain-specific ontologies of objects, properties, actions, and relations. This direction may extend or learn from current ontologies like the CYC ontology project, Google's Knowledge Graph, the OWL web ontology language, and others, but may also take on different design directions.

In the context of autonomous systems, some progress along these lines can be found in, e.g., Traffic Sequence Charts [8], the United States' National Highway Traffic Safety Administration (NHTSA) scenarios [9], Autoware software for autonmous vehicles (https://www.autoware.auto/) and open-source simulators like CARLA. Each of these uses its own terms and concepts as building blocks in bottom-up tool construction.

Beyond the issue of specifying single goals lies the extreme difficulty of specifying how the system should balance, prioritize, or weigh several, often competing goals under a bewildering multitude of circumstances. Many future generation autonomous systems will have to make complex decisions involving major human and business risks, and we doubt that stakeholders can prescribe in advance what the system should do in each case. And, of course, in addition to such technical issues of specification, there are also weighty ethical issues that are outside the scope of this article.

Dynamic changes in specifications constitute yet another complicating aspect. Humans most often deal reasonably well with goal updates in means to achieve them, or in assumptions about the environment. Autonomous systems will have to support such modes of communication and reactive behavior [4].

Explainability and interpretability are especially relevant to emergent properties, particularly for those parts of the solutions based on neural nets and other "black-box" approaches. In a way, explanations induce a model on the seemingly model-less machine-learning solution and summarizing such execution patterns automatically is a challenging problem as discussed in the "Challenge III: Combining 'model-based' and 'data-driven' approaches" section.

The ability to provide concise explanations of the system's decisions, both in real-time and after the fact, will be of great value, allowing developers, and the system itself, to judge the programmed decisions and adjust them as needed.

In fact, an entire subarea of the domain of specifications may emerge. How does one provide a description of what a system actually did in a particular case, in a way that is concise, clear, and has semantics that is agreed upon by everyone involved? Detailed event logs produced by the system, and/or videos of actual behavior, maybe too large, too detailed, or too arcane for the context; say, a discussion of legal responsibility in a court of law, or an engineering review of a functional change, together with a customer. Yet, a verbal summary prepared by a human is prone to error and bias. The same holds for the explanation of why a system behaved in a certain way: describing the rules, algorithms, and computations that drive the system, or the examples it was trained on, may not convey a satisfactory explanation. Again, a higher-level summary prepared by a human from highly technical information may omit important details or incorporate tacit assumptions.

Challenge II: Analysis

By analysis, we mean simulation, testing, formal verification, and system validation against the tacit needs of the stakeholders (STV&V for short). While these techniques will be of paramount importance for next-generation autonomous systems, it is well-known that they, alone or combined, cannot provide complete assurances even for current systems.

The various STV&V techniques all involve one manner or another of executing a system or a model thereof in a controlled fashion, and/or traversing or analyzing the resulting states. Simulation is perhaps the most "hands on" of these and facilitates observing emergent behaviors under a variety of conditions. However, the simulated environment will always be an abstraction and simplification of reality. As mentioned earlier, there are numerous simulation tools relevant for autonomous vehicles. While these are effective and provide important features—albeit, spread across different tools—the foundation proposed here calls for additional important capabilities to deal with the vast number of objects and variables involved in complex autonomous systems, and the even greater number and intricacy of the interactions between them.

Autonomous systems will typically have to deal with numerous new elements, which are often ignored or simplified, or are controlled by other systems. As we did for the specification challenge, we list below some of the issues to be addressed.

One, which is a precondition to any kind of analysis, has to do with the *modeling of environments*. We envision using domain-specific libraries for various kinds of systems and tasks, to deal with the physical three-dimensional space of real-world objects and their properties, including mobility, effects on other objects, and associated risks. These libraries will be different for different application areas.

The second issue which is particularly important for analysis involves the *infrastructure* needed for STV&V. This would have to include mechanisms that adequately orchestrate and control executions. Furthermore, the infrastructure should be "state-aware" and transparent, being able to communicate with engineers using natural interfaces and logs that describe intuitively the state of the external environment, the internal state of the system and its agents, and the state of agents' perception of the external environment. The complexity of all this is amplified by the unpredictability of behavior. Even the relatively simple problem of determining which of the agent's states and interactions may occur in parallel with which others is extremely difficult.

The third major challenge has to do with controlling and measuring the *behavioral coverage* achieved via testing, whether virtual/in silico or by deployment in the real physical world. Such coverage refers to the space of all composite systems states, across multiple components, as well as of the paths and scenarios for reaching these states. Makers of autonomous vehicles sometimes present the distances (real and simulated millions of miles) that their products have driven (see web postings for Google's Waymo and Uber) as an indicator of behavioral coverage. Such metrics will have to be extended. Even what appears to be the bare minimum here—a practical approach to measuring overall composite state coverage for both system and environment—is already hard enough (see discussion in [10]). We envision having to develop techniques to automatically generate rich sets of scenarios, subject to criteria that can be external, i.e., from the environment and the real world, or internal, such as intricate behavioral combinations of specification and implementation entities. Furthermore, given the inability to exhaustively cover all runtime possibilities, we need support for accelerated metamorphic testing in physical environments; i.e., checking thoroughly that the system behaves correctly for a given scenario, and then quickly providing assurances for many other scenarios that differ from the basic scenario only by small environmental changes.

Finally, with regard to STV&V, the autonomics foundation will have to address *formal verification*. Even the best current verification methods can be used successfully only for single components or for greatly simplified models of the entire system. Also, not only is the behavioral specification of the system itself very hard, but it is no easier to specify the behavioral properties that need to be formally verified in terms that are readily aligned with the expectations of the human users and engineers. This is further complicated by the fact that some essential properties are quantitative ones, spanning multiple scales.

The increasing integration of components based on machine learning implies that their verification and the ability to supply adequate explanations of their behavior will become increasingly important. These problems are long recognized as being very difficult, and there is an emerging field of research around them, whose initial results look very promising such as the broad field of generative adversarial networks (GANs) and the Reluplex network verification method [11].

Challenge III: Combining "model-based" and "data-driven" approaches

We use the term *model-based* to emphasize the fact that the designer is required to build and provide a thorough technical description (a model) of the problem, its inputs, its outputs, and the required processing and behavior, in terms that are aligned with the problem domain. This includes classical software development approaches, all of which employ traditional programming languages and prescribe step-by-step processes as well as model-driven engineering (MDE) techniques.

In contrast, we use the term *data-driven* to encompass all techniques that involve machine learning (including, but not restricted to, deep neural networks; abbreviated ML hereafter), statistical analysis, pattern recognition, and all related forms of computing in which the system's behavior is derived from supervised or unsupervised observation.

There is a growing call to find ways to combine the two techniques, leveraging their relative advantages to complement each other [12]–[14]. Nevertheless, there is still no agreement on how to do this, the combination being very different from integration practices in classical engineering.

There are several differences between traditional software development and constructing solutions based on ML, which must be taken into account when trying to integrate the two. To better concentrate on the integration issue in this subsection, we disregard the still open-research problems in each of them.

The first difference involves the *general life cycle*. Traditional software engineering calls for requirements elicitation and specification, design, code, testing, and so on, whether in extended waterfall methodology development phases or short agile sprints. In contrast, developing a system based on ML involves totally different stages, such as the collection, validation and sampling of training data, the actual training, evaluation, and revision and retraining.

The second difference concerns *specifying requirements*. Consider even the simple case, that the brakes must be activated when a stationary obstacle is sensed and the stopping time at the current speed is less than 1 s. The requirement is well defined, and engineers can translate it into working components, but for a system trained to avoid collision based on positive and negative examples, it is not at all clear how to use the requirements or how to incorporate them into the respective ML components. The fact that specified requirements are often associated with operational contexts further highlights the differences between the two approaches in this area.

Related to specifying requirements is the issue of after-the-fact *explainability* (also referred to *interpretability*). This calls for the ability to justify, or rationalize, a particular system decision using problem-related parameters and arguments, e.g., to describe what the system does and the underlying rules, algorithms, and computations. Despite the remarkable success of neural nets, their internal workings are often a mystery. Current ideas addressing this problem are still a far cry from the situation with traditional programming. Moreover, even if explainability and interpretability tools are eventually able to extract the tacit rules behind the operation of large neural nets, the way these rules relate to the net's actual mechanisms will be very different from the relation between natural language descriptions and source code in classical programming languages.

Additionally, this lack of explainability for neural nets makes it very difficult to analyze their behavior. While some initial work has been done on checking properties thereof (see [11]), there is still much to be done on their testing and verification.

An important related difference involves *decomposability*, which is crucial in most stages of development, e.g., for understanding and anticipating system behavior, finding and fixing errors, carrying out enhancements, and assessing the impact of changes. In model-based designs, most system artifacts can be hierarchically decomposed into well-understood functional and structural elements, the role they play in the full system being more-or-less clear. In contrast, the design of data-based ML solutions is typically accompanied by an end-to-end mindset—system-based or problem-based. Being able to decompose a machine-learning solution into meaningful parts appears to be an interesting challenge, which will, of course, bear upon explainability and verification.

Finally, we mention the differences between the two approaches with regard to their *trustworthiness and certification*. Many kinds of autonomous systems are highly critical and their design calls for providing appropriate trustworthiness guarantees for functionality and reliability. These are often specified in standards, like DO178B for avionic systems and ISO 26262 for electronic components in the automotive industry. In principle, model-based techniques give rise to predictability at design time. Achieving for components based on ML an accepted level of trustworthiness and certification requires wholly new technical solutions.

These significant differences illustrate the magnitude of the methodological and technical integration challenges that the autonomics foundation will have to address. To give a relatively straightforward example of this, consider a proposed system that is to ultimately consist of conventional model-driven components (based, e.g., on an object model, algorithms, scenarios, rules, and decision tables) and ML data-driven components (based, e.g., on neural nets). At some point, the engineers should have criteria to decide which subproblems should be solved using which of the two approaches, or perhaps use both, and choose among the various ways by which these components can and should be combined (see [1] for more details).

Discussion

Next-generation autonomous systems are definitely going to be built and will become commonplace in the years to come. Their predicted advent reflects the transition from "narrow" or "weak" AI to "strong" or "general" AI, which cannot be achieved by using just conventional model-based techniques or machine learning alone. Thus, classical software and systems engineering will have to be thoroughly enhanced.

Autonomous vehicles provide an emblematic topical case illustrating the challenge. Due to the lack of standards and compliance assessment techniques, some public authorities allow self-certification for autonomous vehicles, transferring the responsibility back to manufacturers, even though the public might expect otherwise, given the criticality of such systems. Another issue is evidence. Manufacturers will often publicize only partial information about their testing, such as the distance an autonomous vehicle has been test-driven. Another trust-related issue is the fact that critical software can be updated regularly, which raises the concern that updates might be deployed with less-than-adequate testing, causing problems of critical impact.

All this has generated lively public debates. Many important voices tend to minimize the risks from the lack of rigorous design methods: Some claim that we should accept the risks because the benefits will far outweigh them. Others accept the empirical methods and argue that rigorous approaches to complex problems are inherently inadequate. Some people are overoptimistic, arguing that we really do have the right tools, and it is just a matter of time. And besides all of this, we must take into account all relevant ethical/moral, legal, social, and political issues, a vast topic that is obviously outside the scope of this article.

In summary, we are at the beginning of a revolution, where machines are called upon to progressively replace humans in their capacity for situation awareness and adaptive decision making. The extent to which people will ultimately use and benefit from autonomous systems will depend on how much they trust them. We believe that a new engineering and scientific foundation for how to develop autonomous systems will be an important contribution to such progress. ■

Acknowledgments

We are grateful to Guy Katz and Orna Kupferman for valuable discussions in the early stages of preparing this article, and to Carl K. Chang, Werner Damm, and Moshe Vardi for their most insightful and helpful suggestions and comments on the manuscript. This work was supported in part by grants to David Harel from the Israel Science Foundation, Intel Corporation, and the Estate of Emile Mimran, and his endowed William Sussman Professorial Chair of Mathematics at the Weizmann Institute.

■ References

[1] D. Harel, A. Marron, and J. Sifakis, "Autonomics: In search of a foundation for next-generation autonomous systems," *Proc. Nat. Acad. Sci. USA*, vol. 117, no. 30, pp. 17491–17498, 2020. [Online]. Available: https://arxiv.org/abs/1911.07133

[2] *Automated Driving Levels of Driving Automation are Defined in New SAE International*, Standard J3016, SAE International. Accessed: May 2020. [Online]. Available: https://cdn.oemoffhighway.com/files/base/acbm/ooh/document/2016/03/automated_driving.pdf

[3] S. Kounev et al., Eds., *Self-Aware Computing Systems*. Cham, Switzerland: Springer-Verlag, 2017.

[4] J. Sifakis, "Autonomous systems—An architectural characterization," in *Models, Languages, and Tools for Concurrent and Distributed Programming*. Cham, Switzerland: Springer-Verlag, 2019, pp. 388–410.

[5] D. Harel and A. Pnueli, "On the development of reactive systems," in *Logics and Models of Concurrent Systems* (NATO ASI Series), vol. 13. New York, NY, USA: Springer-Verlag, 1985.

[6] S. Boschert et al. (2019). *Symbiotic Autonomous Systems White Paper III*. Accessed: May 2020. [Online]. Available: https://digitalreality.ieee.org/images/files/pdf/1SAS_WP3_Nov2019.pdf

[7] P. G. Neumann, "How might we increase system trustworthiness?" *Commun. ACM*, vol. 62, no. 10, pp. 23–25, 2019.

[8] W. Damm et al., "A formal semantics for traffic sequence charts," in *Principles of Modeling*. Cham, Switzerland: Springer-Verlag, 2018, pp. 182–205.1

[9] W. G. Najm, J. D. Smith, and M. Yanagisawa, "Pre-crash scenario typology for crash avoidance research," Nat. Highway Traffic Saf. Admin., Washington, DC, USA, Tech. Rep. DOT HS 810 767, 2007.

[10] T. F. Kone et al., "Safety demonstration of autonomous vehicles: A review and future research questions," in *Proc. Int. Conf. Complex Syst. Design Manage.* Cham, Switzerland: Springer-Verlag, 2019, pp. 176–188.

[11] G. Katz et al., "Reluplex: An efficient SMT solver for verifying deep neural networks," in *Proc. Int. Conf. Comput. Aided Verification*. Cham, Switzerland: Springer-Verlag, 2017, pp. 97–117.

[12] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," 2017, *arXiv:1708.06374*. [Online]. Available: https://arxiv.org/abs/1708.06374

[13] J. Mao et al., "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," 2019, *arXiv:1904.12584*. [Online]. Available: https://arxiv.org/abs/arXiv:1904.12584

[14] D. Harel et al., "Labor division with movable walls: Composing executable specifications with machine learning and search (blue sky idea)," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 9770–9774.

David Harel is with the Weizmann Institute of Science, Rehovot, Israel. He is the Vice President of the Israel Academy of Sciences and Humanities, Jerusalem, Israel. His research interests focus on logic and computability, software and systems engineering, systems biology, and more.

Assaf Marron is a Researcher at the Weizmann Institute of Science, Rehovot, Israel. His research interests include software engineering, machine learning, and biological modeling. Marron has a PhD in computer science from the University of Houston, Houston, TX, USA.

Joseph Sifakis is emeritus CNRS Research Director at Verimag, Saint-Martin-d'Hères, France, a laboratory that he founded and directed for 13 years. His research interests cover fundamental and applied aspects of system design with a focus on autonomous systems.

■ Direct questions and comments about this article to Assaf Marron, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel; assaf.marron@weizmann.ac.il.