Autonomics: In Search of a Foundation for Next Generation Autonomous Systems

David Harel[1], Assaf Marron[1] and Joseph Sifakis[2]

[1] Weizmann Institute of Science, Rehovot, Israel

[2] Univ. Grenoble Alpes, Verimag laboratory, St Martin d'Heres, France

Abstract

The potential benefits of autonomous systems are obvious. However, there are still major issues to be dealt with before developing such systems becomes a commonplace engineering practice, with accepted and trustworthy deliverables. We argue that a solid, evolving, publicly available, community-controlled foundation for developing next generation autonomous systems is a must, and term the desired foundation *Autonomics*. We focus on three main challenges: (i) how to specify autonomous system behavior in the face of unpredictability; (ii) how to carry out faithful analysis of system behavior with respect to rich environments that include humans, physical artifacts, and other systems; and (iii) how to build such systems by combining executable modeling techniques from software engineering with artificial intelligence and machine learning.

1 Introduction

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. They will be broadly prevalent, and will include, e.g., vehicles of all kinds, medical and industrial robots, agricultural and manufacturing facilities, and distributed management for traffic, urban security, and electric grids.

Many organizations are already striving to develop the next wave of trustworthy, cost-effective autonomous systems, and researchers are busy building powerful tools and methods for the development process. However, the required trustworthiness is not achievable with current practices in software engineering and in artificial intelligence [26]. Extremely high levels of complexity and criticality present fundamental new challenges. Consider, for example, even a very modest autonomous system, a valet-parking robot – obviously a far cry from a full autonomous vehicle. Customers and regulators would be fully justified in asking whether the robot will be able to discover a child forgotten in the car, or notice that a pet dog is perched underneath it. Even if it is able to notice these, what will the robot do as a result? How will it react if a human attempts to stop it by pursuing it and yelling?

Such development challenges cannot be dealt with by merely enhancing the system's safety features, say, adding sensors and dedicated safety components, and carrying out richer test cases.

Next-generation autonomous systems will be expected to operate under conditions that will often be unpredictable at the time of their development, due to limited control over the system's environment, the dynamic emergence of new kinds of objects and events in the world, and the exponential growth in the number of composite configurations of such elements, old and new alike.

The literature contains interesting demos and discussions of unpredictability in autonomous systems; see, e.g., the SpotMini video [3] and the discussion of its demonstration environment [22]. Engineers must be able to assure customers and regulators that the system will function correctly and safely, even though many of the complex situations it might have to deal with are still unknown.

We believe that to close this gap between the challenges in developing trustworthy next-generation autonomous systems and the present state of the art, the research and industry community must construct a common engineering foundation for developing such systems. This foundation, which we term *Autonomics*, should address the unique challenges presented by next generation autonomous systems, providing engineering principles, methods and tools for their satisfactory development, as well as means for selecting among multiple design alternatives. We believe that such a foundation will dramatically accelerate the deployment and acceptance of high quality, certifiable autonomous systems, built for the benefit of human society.

2 Next Generation Autonomous Systems

2.1 Definitions

Over the years, many definitions have been offered for autonomy (e.g., [18, 11]). *Autonomic Computing* (e.g., [7]) focusses on systems capable of self-management, and in particular, automating dynamic configuration. The broad research area of *multi-agent systems* (e.g., [24]) pays 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., [15]), which implies the system's ability to perceive changes of the environment and use "knowledge" of its own states to react adequately, so that a set of goals is achieved. *Symbiotic Computing* (e.g., [8]) studies how autonomous systems can interface and collaborate with humans and with complex organizations, considering the many technical, commercial, and ethical implications thereof.

In order to streamline the ensuing discussion, we offer, in this subsection and in the next one, definitions for some basic concepts (see also [25, 27]), and illustrate them with an example of an autonomous vehicle for factory-floor and plant yard deliveries (termed here FFAV).

Systems are the artefacts that development teams are out to build. A system works within an external environment, and it consists of two types of components, agents and objects (the latter, as explained below, are typically not built by the development team), which, as we shall see, operate within a common internal system environment. 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. For example, objects of the FFAV system may include ready-made components, such as the motor, a set of cameras, or a steering mechanism whose input is, say, the desired angle of the front wheels. A system often interacts with objects that are not part of it, but are part of the environment. Such are, in the case of the FFAV, machines on the factory floor (which may be mere obstacles or recipients of deliveries), and packages to be delivered. Objects have *states*, which can be changed by agents or by other objects, or can change ‘spontaneously’, for internal reasons.

Agents are the main behavioral elements of an autonomous system. They are those designed (programmed, built) as part of the system’s development process.[1] 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. Thus, the FFAV system may either have a single agent for all its functions, or separate agents for different tasks, such as work scheduling, route planning, travel control, gripping, and carrying. Agents can themselves be autonomous systems, and can, in turn, hierarchically contain other agents and systems.

The internal environment is the lower-level physical and virtual infrastructure used by the system's agents and objects. It may include the computer/processor/memory, batteries and other power sources, the operating system, communication hardware and software, and data-base management software.

The external environment of a system (often simply called the **environment)**, 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 any other physical or virtual entities that may affect, or be affected by, the system's behavior.

2.2 Autonomous Behavior

We say that a system or an agent (for simplicity, we shall stick to 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 from 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. Often perception has to deal with multi-modal inputs, such as vision, sound, heat, touch, radar, and data communication from other systems, all obtained using mode-specific sensors and input devices, and has to then amalgamate the received information. The second function is **Model Update**, which uses the information provided by Perception to create and constantly update an integrated run-time 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. This means that decisions consider many possibly conflicting goals, in a way that depends on the system’s current state and that of the environment. **Goal Management** chooses from among the set of goals the ones that are

[1] We ignore here the question of whether a given component, already developed and then incorporated ‘as is’ into the system, should be considered as an agent or an object thereof. Similarly, we sidestep the question of whether agents that are part of systems in the external environment, like those of autonomous manufacturing machines in the context of the FFAV, should be considered agents or objects.

relevant to the current state, and **Planning** computes a plan to achieve the set of goals produced by Goal Management, subject to state-dependent constraints. The plan 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.

2.3 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 the 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 a vehicle reaching a destination without collisions. They will typically face a far wider and more elaborate set of goals, as humans often do. Consider, the FFAV making a highly critical (and expensive) delivery, which may be at risk due to a safety issue. The situation is further complicated by the financial and legal considerations of its manufacturer.[2] For example, the owner of a chemical plan that uses an FFAV may want to allow the FFAV to (carefully) disobey a stop sign when making a very urgent delivery, but the FFAV's manufacturers might have programmed it for absolute compliance with the law, in order to reduce their liability.

Their environment is dramatically less predictable. Even autonomous systems of the present already have to deal with too many *known* environment configurations, and those we do not know about yet will obviously add a whole new order of magnitude to this difficulty. First-cut autonomous vehicles have to struggle with road-intersection topology, and varying traffic volume and speed, which can probably be addressed using existing technology. In contrast, there are far 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 weaving in and out of traffic on roads, sidewalks, and crosswalks; police instructions, spoken or signaled; poorly marked temporary diversions; emergencies that have not yet been handled by first responders, such as a traffic accident, a landslide/rock-fall or flooding; or an urgent request by a passenger to stop and step out, but where there is no safe place to do so. These kinds of difficulties are caused by the increased dependency on hard-to-predict physical aspects of the dynamic environment, compounded by the increased mobility, distribution, and sheer multitude of systems.

They require rich interaction with humans. Classical human-computer interaction (HCI) is typically geared towards trained users or operators controlling automated tasks. Future interfaces

[2] To some extent, the existence of multiple, possibly conflicting, goals reflects the transition from "weak AI" to "strong AI", which contrasts specific problem-solving with devising general decision methods for complex combinations of goals. We believe that strong AI is not a prerequisite to trustworthy autonomous systems of the future: individual systems will be able to solve complex issues without necessarily having at their disposal general strong AI solutions.

will have to deal with the overall behavior of the system as sensed by humans, with the way the system affects human behavior and with the way humans think about system behavior. Next generation autonomous systems will affect and put at risk a far wider circle of people, and will be increasingly exposed to both helpful and adversarial human actions, with the required communication and interaction that they necessitate. Suffices to think of a traffic jam caused by an autonomous vehicle on a busy highway, an autonomous crane on a busy construction site in a city center, or a medical-supply delivery robot, scurrying down a crowded hospital corridor.

The issue of interaction with humans goes much deeper than classical HCI. First, because future systems will operate in common human environments, having to interface with humans who are neither users nor operators, and over whom the owner of the autonomous system has no control. Their behavior will not only have to *be* functional, efficient and safe, but will also have to *appear* to be so, in order to instill in humans the confidence that this is indeed the case. These systems will be interfacing with humans in wholly new ways even as part of normal everyday routines, such as negotiating the right-of-way through an office door, or pointing out a spill on the floor to a passing robotic cleaning assistant.

Second, 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. If the FFAV has to hand a fragile package to a human, and take from them another package, how does it communicate its readiness to hand over one and receive the other, its questions (e.g., has the human recipient already secured a hold on the first package?), its state (e.g., that it is now holding the second package safely so that the human's hold and attention is no longer needed), and so on.

Third, special attention has to be given to those parts of the interface that allow a human to interrupt the operation of the autonomous system or change it abruptly. If a worker just dropped a contact lens that the FFAV cannot see, how does he/she immediately stop it? If the FFAV was given the wrong package, how does the human call it back? If some emergency work blocks the normal, pre-programmed route of the FFAV, and a detour cannot be easily discovered, how does one give the FFAV an alternate, ad-hoc instruction, in real time and in a natural way, that will cause it to use a particular alternative route, say, the handicap ramp behind building C?

3 Why a New Foundation?

Our main claim in this paper is that developing trustworthy next-generation autonomous systems requires addressing fundamental issues that have not been dealt with adequately by present research or industrial experience. 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. While this Autonomics foundation should touch upon all aspects of system engineering, it should not aim at rewriting well-accepted system engineering principles. Instead it should be built to address the 'burning' issues 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 Neumann [21] and Bellovin and Neumann [2], whose main focus is the need to handle the impact of component vulnerability on full, composite system vulnerability. The literature dedicated to building and testing complex, autonomous, safety-critical systems (e.g., [1]) provides precious little in way of theories and tools for ensuring one's confidence (or trust) in the system's run-time behavior in face of unpredictable situations. In a closely related discussion of the challenges facing next generation systems, Chang [5, 6] argues that new solutions, and indeed a paradigm shift, are both required and possible, and that these can be enabled by new approaches to situation analysis.

To reinforce our argument about the depth and urgency of the required foundation, we shall focus here on one central aspect, which is at the very heart of autonomous system engineering – the decision-making. We present three partially-overlapping challenges in developing decision-making processes, for which satisfactory solutions have yet to come.

3.1 Challenge I: Specifying Behavior

Behavioral specification is needed in virtually all stages and activities of the development process: requirements, design, simulation, testing, verification, and validation. Behavior should preferably be specified in some rigorously defined language with agreed-upon dynamic semantics, but, at the very least, can also be done in in a way that can be mapped directly to precise and technically-oriented natural language descriptions. Although numerous diverse computer languages have been developed for this purpose – procedural, declarative logic-based, scenario-based, and more – we argue that in next-generation autonomous systems the very specification of desired, undesired, or even actual observed behavior introduces new issues that call for extensive research. These include means to specify individual goals and means to manage multiple goals and their inter-relationships.

When it comes to complex autonomous systems, the specification of even a simple single goal is hard. Assume that, for the first time, an employer wants to completely automate the floor-cleaning process. What kinds of specification are we after? Should it be action-oriented (e.g., where and how to sweep), 'object'-oriented (e.g., what kinds of dirt should be removed, and from where), or result-oriented (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 believe that what we need here are 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, extending current ontologies like the CYC project [16], Google's Knowledge Graph, the OWL web ontology language and others. The goal in this case is to answer simple questions with simple answers. For example, if the FFAV sees in its path an object it cannot recognize, it should be able to present a picture and perhaps sensor information to other facilities, such as a server in the cloud. The server might then inform the FFAV as to what the object is, so it can apply its existing rules, or it may instruct the FFAV to take certain actions based on the server's knowledge and logic.

In the context of autonomous systems, some progress along these lines can be found in, e.g., Traffic Sequence Charts [9], USA NHTSA scenarios [20], Autoware software for AVs [13] and open source simulators like CARLA [10]. Each of these uses its own terms and concepts as building blocks in a 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. Even in a single given situation, and even if we allow the use of natural language for specification, it is often almost impossible to state what the system should or should not do. 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. Courts of law must rule on whether a decision made by a human was right or wrong, negligent or not, in line with what is expected of a reasonable person. This becomes significantly harder in the realm of autonomous systems, which, for example, have to determine where exactly to make the initial incision during a surgical procedure, or decide in a split second between two very bad alternatives in an emergency driving situation.

During simulations, stakeholders often encounter emergent properties and unexpected behaviors that were not mentioned in the development process. For example, when observing the behavior of an autonomous vehicle, such as an FFAV or a golf cart, one may notice that it sometimes repeats a path with unusual precision, creating unexpected wear on the floor, ground or grass. New requirements may be desired as a result, such as randomizing paths, documenting a limited set of supported surfaces, or even taking advantage of this kind of predictability in other parts of the system.

Making this happen is not easy. It is not even clear how to specify a formal version of a "trouble ticket", which describes an event, property or pattern that was noticed by the human observer but was not part of the original specification. Furthermore, one would want to automate the detection and articulation of emergent properties, since testing and simulation are likely to be highly automated, with limited opportunity for human observation. For anticipated behavior (desired or undesired), this would be quite similar to testing, but for unexpected behavior, automating the capturing itself in a formal, yet succinct and intuitive way, would appear to be close to impossible. In fact, we believe that Knuth's famous quote, "Beware of bugs in the above code; I have only proved it correct, not tried it," goes beyond recognizing the importance of testing given the limitations of formal methods and correctness proofs. It can be used to support our belief that testing is a must also because just observing the system in operation yields totally new insights about what the system does and does not do, and what it should or should not do.

Explainability and interpretability are especially relevant to emergent properties, particularly for neural nets and other 'black-box' solutions. In a way, explanations induce a model on the seemingly model-less machine-learning solution. And here too, summarizing such execution patterns automatically is a challenging problem that is the subject of active research.

The task of specifying and explaining behavior that must dynamically reconcile multiple goals can be aided by initially adding weights, priorities, and mutual constraints to goals, as well as just observing the system in operation and endowing the components responsible for the various goals with dynamic internal negotiation capabilities. A key role will also be played by mechanisms for specifying and handling contingent behavior, which reacts to and handles negative conditions directly related to the system's actual behavior. 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.

3.2 Challenge II: Analysis

By analysis we mean simulation, testing, formal verification, and system validation against stakeholders' tacit needs (STV&V for short). While these techniques will be of paramount importance for next generation autonomous systems, it is well-known that none of them provide complete assurances even for current systems, so they will have to be used in ways that complement each other.

The various techniques comprising STV&V 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 – desired, undesired, and not-yet-specified – under a variety of conditions. However, the simulated environment will always be an abstraction and simplification of reality. There are numerous simulation tools relevant to autonomous vehicles; see, e.g., [4, 8, 11, 17]. While these are effective and provide important features – albeit, spread across different tools – the foundation proposed here calls for additional important capabilities, such as far greater user control over environment variability and the ability to automatically detect and evaluate new emergent properties that were not in the original test specifications.

One of the main reasons that satisfactory STV&V analysis calls for foundational work, is the vast number of objects and variables involved in complex autonomous systems, and the even greater number and intricacy of the interactions between them. This is what we now discuss.

Autonomous systems will typically have to deal with numerous new elements, which are often ignored or simplified, or are controlled by other systems. Just think of an FFAV deployed at a busy outdoor factory yard, with people, equipment, and vehicles moving around, possessing distinct shapes, colors, reflection types, textures, sizes, locations, positioning and routes. Faithful simulation and effective testing of these systems and environments is a formidable task. And as to interactions thereof, here the problem becomes alarmingly worse. The system's "intentional" activities may be reasonably controlled, but the number and complexity of the possible interactions and indirect effects between all objects in the environment and the system, are mind-boggling. Consider testing a very small FFAV working its way through a throng of humans and machines, perhaps even other living creatures – as in a livestock show. The interactions it has to deal with are not limited to the obvious goals of, e.g., reaching a destination while avoiding collisions and abiding by traffic laws. How about the interactions between objects that arise when the FFAV tries to avoid splashing passersby when it crosses a puddle? How should it deal with cases where it might inadvertently startle humans when quietly and suddenly showing up at their side, or when it gets entangled in loose fabric or a cable?

Here too, we can list some of the issues that the Autonomics foundation should address regarding analysis. One, which is a pre-condition 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, in order to deal with the physical 3D space of real-world objects and their mobility. These libraries will be different for different application areas. Just think of the different kinds of environments that would be relevant to a medical system and a transportation system. The environment would have to be modeled using languages and tools that are able to describe knowledge of the environment and assumptions thereof, and to achieve a desired level of realism by controlling abstraction levels and simulation granularity.

The second issue the foundation will have to address, which is particularly important for analysis, involves the **infrastructure** needed for STV&V. We want to be able to test and simulate autonomous agents in interaction with the complex cyber physical environment for which they are being built. 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. For example, assume you want to figure out what the FFAV will do when it faces an obstacle consisting of two posts placed at a distance apart that is barely more than the width of the FFAV. Will it move between them or bypass them? Standard testing and simulation techniques call for actually placing the obstacles and observing the system's behavior. However, unless we also carefully check the feedback from the tested system as to what it 'thinks' it saw, one cannot be sure if it perceived the conditions as intended. Thus, the FFAV may indeed pass between the posts, but for the wrong reason: it might have misclassified one or both of them.

Such perception control is a particular case of state awareness, where during STV&V the infrastructure can report the system's state and that of the environment; can guide the operation of system components based on the states of others; and can report, and react to, things the system does and does not do, its execution paths, etc. 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 of the analysis part of the foundation has to do with controlling and measuring the **behavioral coverage** achieved via testing. Current testing methods for autonomous systems focus on achieving simple kinds of coverage: do the scenarios being tested "touch" each of the system's elements (e.g., all its potential actions, methods and lines of code), and all of its environment's known conditions and events. Some simply take the vastness of the testing efforts to be what counts: for example, makers of autonomous vehicles present the number of miles (real and simulated) that their products have driven, as though merely testing them for significantly more than what a typical driver would ever experience is sufficient (see, e.g., [17, 28]).

For the kinds of systems we have in mind, which will be deployed in large numbers worldwide, these approaches are inadequate. Even what appears to be the bare minimum here – a practical approach to measuring state coverage for both system and environment – is already hard enough. 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 run-time 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 physical changes. We also need fitting criteria for evaluating the testing process itself.

While tools like Generative Adversarial Networks (GAN) can discover (and help fixing) failures in deep neural nets, much work is still needed to enable systematic confirmation that a given AI-based autonomous system will handle a set of scenarios correctly with adequately high probability.

Finally, the Autonomics foundation will have to address **formal verification**. Even the best current verification methods can be successfully used for only single components or 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 assertions that describe the behavior we want to verify in terms that are readily aligned with the expectations of the human users and engineers. This is further complicated by the fact that whether some behavior is desired or not may not by a binary decision, but quantitative, residing on multiple scales[4].

Since many next generation autonomous systems will have components based on machine learning, the formal verification of neural nets, and the ability to supply adequate explanations of their internal 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. Accordingly, the next section takes a closer look at the challenge involved in incorporating such AI-based learning techniques into the foundation.

3.3 Challenge III: Combining 'Model-Based' and 'Data-Driven' Approaches

In the ensuing discussion we use the term **model-based** to include classical software development approaches, all of which employ traditional programming languages and prescribe step-by-step processes and/or rules that are meticulously handcrafted and organized by humans. This also includes MDE techniques that use languages like the UMLK. However, we do not restrict ourselves to MDE. We have decided to use the term model-based in order 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.

In contrast, we use the term **data-driven** to encompass all techniques that involve machine learning (including, but not restricted to, deep neural networks), statistical analysis, pattern recognition, and all related forms of computing in which the system's behavior is derived from supervised or unsupervised observation. The latter can include observing input and output events and occurrences in the real world or in the processing of other systems, even that of earlier versions of the system under development. The desired behavior of the system we are developing is thus inferred; not prescribed as in model-based approaches.

There is a growing call to find ways to combine the two techniques, leveraging their relative advantages to complement each other [12, 23, 19]. Nevertheless, there is still no agreement on how to do this, the combination being very different from integration practices in classical engineering. And, of course, due to the new challenges involved, the problem is further exacerbated for next generation autonomous systems.

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 (like verification and explainability of neural nets).

The first difference involves the **general life cycle.** Traditional software engineering – in any of a number of classical life cycle methodologies – calls for requirements elicitation and specification, design, code, testing, and so on. In contrast, developing a module or system based on machine learning involves totally different stages, such as the collection, validation and sampling of training data, the actual training, evaluation, revision and retraining, etc.

The second difference concerns **specifying requirements**. Consider even very simple cases, such as requiring that an electrical switch must turn off when the temperature reaches 80 degrees, or that the brakes must be activated when a stationary obstacle is sensed and the stopping time at the current speed is less than 1 second. These are well defined, and engineers can translate them easily into working components, but for a system trained to handle excessive heat or 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.

Related to specifying requirements is the issue of after-the-fact **explainability and interpretability**; i.e., the ability to describe what the system does, articulate the underlying mechanisms it uses (algorithms, computations, etc.), and justify the decisions it makes. Despite the remarkable success of neural nets in performing many kinds of tasks, their internal workings are often a mystery. Current ideas addressing this problem are still a far cry from the situation with traditional programming, for which engineering practices recommend producing code that is easy to read and understand, and to enhance it with ample comments. 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 programming languages.

In addition, the difficulty in specifying the behavior of neural nets and explaining what they do and why they do it, makes it very difficult to analyze their behavior. While some initial work has been done on checking properties thereof (see, e.g., [14]), there is still much to be done on the testing and verification of systems based on learning.

Another striking difference, closely related to the ones above, involves **decomposability**, which is considered crucial in most stages of development. 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 **trustworthiness and certification**, which are clearly related to testing and verification. Many kinds of autonomous systems are highly critical – failure to meet their expected behavior can be disastrous. Critical system design calls for providing the trustworthiness guarantees required by the appropriate standards; e.g. 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, but components based on machine learning are not engineered in the same way. Achieving for them an accepted level of trustworthiness and certification requires wholly new technical solutions, since conventional testing and simulation techniques are wholly inadequate given the complexity and unpredictability associated with next generation autonomous systems. Complementary non-technical measures, such as risk management, the concept of insurance, or the use of the justice system as deterrent against negligence, are separate issues altogether and outside of the scope of this paper.

These significant differences illustrate the magnitude of the methodological and technical integration challenge, which 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 will have to decide which sub-problems should be solved using which of the two approaches. Sometimes the answer is easy: for example, reading traffic signs can be fully ML-based, while the decision to remain below a known speed limit can be model based. Staying in lane on a clean, well-marked highway might be model-based, while negotiating a road surface covered with sand could very well be ML-based. However, in many cases, the choice between the two approaches will be a lot more difficult. Besides, the means for deciding at runtime which situation is the relevant one, in order to activate the proper component (e.g., whether the road surface is clean or sandy), will also have to be developed, and which of the two approaches is to be used for that decision also has to be decided by the designers~~.~~.

The foundation should describe and discuss various specific approaches to the integration. One example is a ‘pipeline’ approach, where some pre-processing is done by one method, and interim results are passed on to the other method. In the ‘divide and conquer’ approach, the problem is divided into sub-problems, and different techniques can be applied to each; the actual division can itself be model-based or data-driven. In another approach, one method (usually rules) serves as a ‘protective wrapper’ around the other (usually ML-based), constraining the latter to be within some decision domain. In additional variants, an approach that is mostly model-based can incorporate several data-driven black boxes to enrich sensor processing or to solve particular sub-problems. One can also apply both techniques to develop complete solutions, thus creating redundancy, and then use various composition or ‘voting’ techniques to yield the final system behavior.

4 Discussion

Next-generation autonomous systems are imminent, and many of them will be manifest some important form of criticality. They will have to cope with the uncertainty of complex, unpredictable cyber physical environments, and will have to adapt to multiple, dynamically changing and possibly conflicting goals. They will be expected to collaborate harmoniously with humans, giving rise to so-called “symbiotic” autonomy. 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. For example, due to the difficulty in explicitly specifying and then checking compliance with the rules and expectations that people may be concerned about, some public authorities allow self-certification for autonomous vehicles, transferring the responsibility to the manufacturers. 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. One can then only hope that the behavioral coverage was indeed sufficient, and, that someone other than the manufacturer indeed examined the tests and deemed them satisfactory. Another trust-related issue is the fact that critical software can be updated regularly, which creates the hope that certain kinds of failures of an AV would be immediately corrected in all AVs worldwide. However, this also raises the concern that updates might be deployed with less-than-adequate testing, causing problems of more critical impact than standard updates to the operating systems of smartphones or personal computers.

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 clearly outside the scope of this paper.

Our paper's contribution is twofold. First, we propose a basic terminology for next generation autonomous systems, and a framework capturing their main characteristics, and discuss how they are different from present-day autonomous systems. The framework provides insight into the spectrum of possibilities between automation and autonomy, and is intended to help in understanding the degree of autonomy of a system as the division of work between a system and human agents.

Second, we claim that the advent of next generation autonomous systems raises an extraordinary scientific and technical challenge, and advocate the need for a new foundation that will address the key open issues in their engineering. Such an Autonomics foundation will hopefully lead to trustworthy hardware/software systems. The degree of success in meeting this challenge will ultimately help determine the extent of acceptance of such systems, as a compromise between their estimated trustworthiness, the anticipated benefits of the automation they afford, and the required changes in other systems and in human behavior.

We anticipate that forming this foundation will require major and ground-breaking efforts in three main directions.

The first is to develop a rigorous theory and supporting tools for dealing with heterogeneous specifications. These should make it possible to characterize system behavior in a broad fashion, including the behavior of its individual agents, as well as the system's global behavior in terms of its overall goals and its emergent properties.

The second direction is aimed at providing sufficient evidence of a system's trustworthiness. We have emphasized the paramount importance of modeling and simulation: we need faithful, realistic modeling of behavior, as well as semantic awareness, so that the experimenter has access to a meaningful abstraction of the system's dynamics, allowing controllability and repeatability of the testing. The latter will also allow behavioral coverage, where we measure the degree to which relevant system configurations have been explored.

The third direction of the effort required for the foundation involves adopting a powerful “hybrid” design approach, seeking tradeoffs between the trustworthiness of classical model-based approaches and the performance of data-based ML ones. Taking better advantage of each approach requires the development of common architectural frameworks that would integrate modules characterized by their pure functionality independent of their design approach. Developing the theory for decomposability, interoperability and explainability of data-based modules is essential for reaching this goal.

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. This requires some aspects of general AI, which go beyond the objectives of ML-enabled intelligence. The extent to which we will ultimately use and benefit from autonomous systems will depend on how much we trust them, an issue which is the main reason for writing this paper.

Acknowledgements

The authors are grateful to Guy Katz and Orna Kupferman for valuable discussions, comments, and suggestions. 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 from his endowed William Sussman Professorial Chair of Mathematics at the Weizmann Institute.

References

[1] ACM. ACM Transactions on Autonomous and Adaptive Systems (TAAS). URL: https://taas.acm.org/ Accessed 10/2019.

[2] Steven M Bellovin and Peter G Neumann. The big picture. *Communications of the ACM*, 61(11):24–26, 2018.

[3] Boston Dynamics. Boston Dynamics Spot Robot is Ready to Leave the Nest, 2019. URL: https://www.youtube.com/watch?v=jEr1kRf_FC0 . Video accessed 10/2019.

[4] Patricia Bouyer, Orna Kupferman, Nicolas Markey, Bastien Maubert, Aniello Murano, and Giuseppe Perelli. Reasoning about quality and fuzziness of strategic behaviours. *arXiv preprint arXiv:1905.11537*, 2019.

[5] Carl K Chang. Situation analytics: a foundation for a new software engineering paradigm. *Computer*, 49(1):24–33, 2016.

[6] Carl K Chang. Situation analytics—at the dawn of a new software engineering paradigm. *Science China Information Sciences*, 61(5):050101, 2018.

[7] Autonomic Computing et al. An architectural blueprint for autonomic computing. *IBM White Paper*, 31(2006):1–6, 2006. URL: https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf Accessed 10/2019.

[8] Stuart Mason Dambrot, Derrick de Kerchove, Francesco Flammini, Witold Kinsner, Linda MacDonald Glenn, and Roberto Saracco. Ieee symbiotic autonomous systems white paper ii, 2018.

[9] Werner Damm, Eike Möhlmann, Thomas Peikenkamp, and Astrid Rakow. A formal semantics for traffic sequence charts. In *Principles of Modeling*, pages 182–205. Springer, 2018.

[10] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

[11] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 21–35. Springer, 1996.

[12] David Harel, Assaf Marron, Ariel Rosenfeld, Moshe Vardi, and Gera Weiss. Labor division with movable walls: Composing executable specifications with machine learning and search (blue sky idea). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9770–9774, 2019.

[13] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, and Tsuyoshi Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, 2015.

[14] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[15] Samuel Kounev, Jeffrey O. Kephart, Aleksandar Milenkoski, and Xiaoyun Zhu, editors. *Self-Aware Computing Systems*. Springer, 2017.

[16] Douglas B Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[17] Waymo LLSC. JOURNEY. URL: https://waymo.com/journey/ . Accessed 10/2019.

[18] Michael Luck, Mark d'Inverno, et al. A formal framework for agency and autonomy. In *ICMAS*, volume 95, pages 254–260, 1995.

[19] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

[20] Wassim G Najm, John D Smith, Mikio Yanagisawa, et al. Pre-crash scenario typology for crash avoidance research. Technical report, United States. National Highway Traffic Safety Administration, 2007.

[21] Peter G Neumann. How might we increase system trustworthiness? *Communications of the ACM*, 62(10):23–25, 2019.

[22] Simone C Niquille. Regarding the Pain of SpotMini: Or What a Robot's Struggle to Learn Reveals about the Built Environment. *Architectural Design*, 89(1):84–91, 2019.

[23] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[24] Onn Shehory and Arnon Sturm. *AGENT-ORIENTED SOFTWARE ENGINEERING*. Springer, 2016.

[25] Joseph Sifakis. Autonomous systems–an architectural characterization. In *Models, Languages, and Tools for Concurrent and Distributed Programming*, pages 388–410. Springer, 2019.

[26] Joseph Sifakis. Can we trust autonomous systems? boundaries and risks. In *International Symposium on Automated Technology for Verification and Analysis (ATVA)*, pages 65–78. Springer, 2019.

[27] Joseph Sifakis et al. Rigorous system design. *Foundations and Trends in Electronic Design Automation*, 6(4):293–362, 2013.

[28] Venturebeat. Uber's 250 autonomous cars have driven 'millions' of miles and transported 'tens of thousands' of passengers. URL: https://venturebeat.com/2019/04/11/ubers-250-autonomous-cars-have-driven-millions-of-miles-and-transported-tens-of-thousands-of-passengers/ Accessed 10/2019.