# Requirement Specification, Analysis and Verification for Autonomous Systems

Alessandro Pinto
*Intelligent Systems*
*Raytheon Technologies Research Center*
Berkeley, CA
alessandro.pinto@rtx.com

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAC18074.2021.9586208

***Abstract*—The environments and the goals that an autonomous system needs to be able to understand and act upon are not fully known and clearly identifiable at design time, especially for those systems that interact with the physical world. Still, for applications that require a certain level of assurance, requirements, which include the definition of how a system is supposed to interact with its context, need to be defined, analyzed, refined towards an implementation that can be shown to deliver certain guarantees. We first broadly discuss the importance of the early stages of the design process where requirements are defined, and we then discuss the formalisms needed to capture requirements and refine them towards implementation for autonomous systems.**

***Index Terms*—Autonomous systems, design, analysis, requirements**

## I. Introduction and Background

Automation is a common feature of many products and services that we use on a daily basis. The availability of powerful computational resources, large storage capacity, and high-bandwidth data links, even on small portable devices, together with a better understanding of how to build approximate models for decision-making starting from data, has led to an increased sophistication of automation features towards autonomous systems that collaborate with humans. These systems have been developed for, and deployed in a range of applications from consumer electronics, to health and fitness, domotics, manufacturing, energy, and transportation. While autonomous systems have made these products more usable and useful, have improved productivity, and relieved humans from tedious and/or dangerous tasks, their implementation can exhibit unexpected behaviors. Examples include the emergence of echo chambers and other effects in the online world [11], [17], [26], car accidents, and cyber-physical attacks [9]. The inability to provide high-level of assurance for autonomous systems justifies the resistance that some industry sectors, such as commercial aviation, and others that serve mission critical applications, have shown towards the adoption of advanced automation[1]. The stark reality is that, in many realistic cases, the modeling and analysis methods available today cannot accurately predict the behavior of an autonomous system.

[1]In this article we will refrain from using terms such as "intelligence", or "artificial intelligence". We will also later discuss what automation means.

There is a need for new methodologies and tools to develop, deploy, and maintain advanced autonomous functions. Ideally, these new methodologies and tools should provide confidence that the system is safe to be placed in operation, perhaps with reduced capabilities or limited performance. New tools are also required to collect relevant operational data and evolve a system towards optimal performance and increased levels of automation, while maintaining safety and satisfying high-level requirements, such as unbiased decision-making. Such goal will likely require the definition of a new design process (or the adaptation of available ones that have demonstrated the ability to deliver high assurance systems) together with a set of tools that can jointly contribute to an assurance argument. *A critical step in a design process is the elicitation and definition of requirements* that specify the system boundaries, the expected behaviors, and non-functional constraints such as minimum performance requirements.

As an example, the development of commercial aircraft must follow design guidelines that have been developed over the past many years. These guidelines are codified in standards that cover safety assessment [23], system development processes (from concept to aircraft, systems, and items) [24], and hardware and software development processes [21], [22]. When followed, these guidelines deliver high assurance systems as proven by a very improbable occurrence of catastrophic events in aviation [1]. Particular emphasis is placed on the *generation, traceability, validation and verification of requirements*.

Clearly, safety assurance also relies on humans. In the aviation domains, for examples, pilots are instrumental in dealing with contingencies that may occur at any time and in any phase of flight. Dealing with contingencies requires a deep understanding of the world and of complex machines such as the aircraft. Flying a commercial airliner, in fact, requires a formal background and thousands of flight hours, indicating the need for a vast amount of aviation and common-sense knowledge. Re-allocating tasks from humans to machines requires developing autonomous functions with the same level of understanding of the world and of the physical systems they are operating. This task could become a roadblock. It is, therefore, crucial to carefully distill essential knowledge requirements for autonomous systems, and to develop tools to help extract high-quality knowledge from a variety of sources

including data and documents.

Given its importance, this paper focuses on the requirement engineering process, and provides a list of promising techniques for requirement specification, validation, and verification. We discuss the typical structure of an autonomous system that serves as high-level functional architecture. We then discuss methodological guidelines for the definition of requirements, and conclude with a review of promising tools for validation and verification. Requirement engineering for autonomous systems, including those systems that are based on data-driven models, is a very active area of research involving academia, industry, and standardization authorities [2], [3], [12], [15], [19], [25], [27], [27], [27]–[29], [31].

## II. Three types of agents

An attempt at defining autonomous systems may quickly lead to controversies. It is useful, however, to have a small taxonomy in mind, as different types of automation pose different challenges to traditional design processes. We adopt here definitions similar to the ones proposed in [5] and [14]. The simplest autonomous agent is a *controller* that uses a predefined function to map the current state of the world to the action to be performed. The state of the world can be estimated accurately enough through models. These agents are used when the number of possible states and the actions can be enumerated, or when there is a computable and verifiable model that defines those maps. Several automation functions in aviation and automotive are implemented by controllers that belong to this class and that have been certified even under stringent safety requirements. For example, consider the control of power transfers on an aircraft [16]. The controller, albeit complex, is able to maintain critical loads powered under several fault conditions and can be certified as airworthy.

When the number of situations to handle is too large, agents are required to make decisions at run-time. *Planning* agents are given goals, constraints and cost functions, and compute plans as partially ordered sets of actions. The action models are given to the agent and are assumed to be invariant. The most advanced and complex case, however, is the one of *self-evolving* agents that are also required to learn models of the environment and/or of their own capabilities, and to use them towards improving their overall performance, or to implement new behaviors.

Common to the second and third class of agents, are three important features: (1) agents are expected to respond to a wide set of goals, too large to be enumerated at design time, and in a wide range of environments which is also too large to be enumerated, (2) the timescale of a plan is much longer than our ability to predict the evolution of the environment with a certain degree of certainty, and (3) not only the number of situations is too large to be enumerated, but there is a fundamental inability to describe the characteristics of the environment in a form amenable to model-based design or testing.

These three points challenge some assumptions on which the requirement generation process stands. The first problem

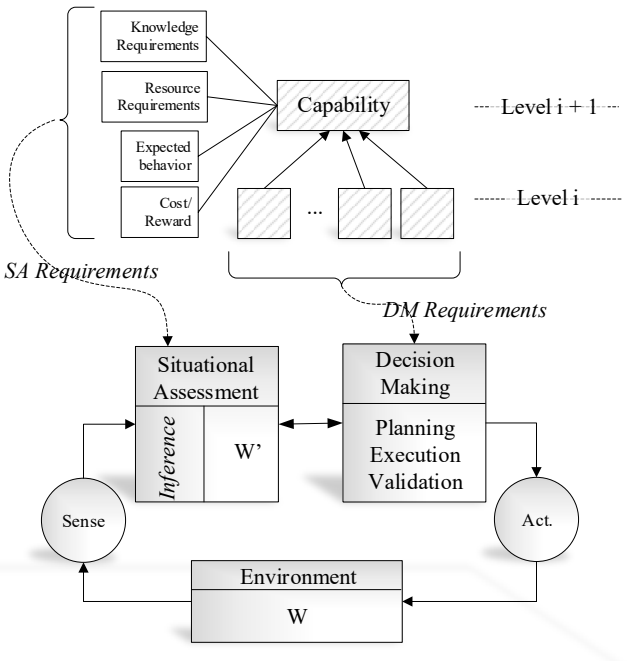


Fig. 1. Overview of the high-level functional architecture of an autonomous systems (bottom), the capability model, and the requirement allocation process (top).

is the identification of the basic functions of the system. Standards such as [23] provide a basic list of functions that are needed to control a system, or defines the steps to specify such list as part of the design process. Freedom to use the capabilities of as system in creative ways is left to the operator. Fore example, experience pilots can use the capabilities of an air vehicle in many ways and combinations to achieve a variety of goals such as transport passengers or cargo, help in fighting fires or rescuing lost skiers, land on a freeway or in the Hudson River. Each of these use cases comes with a range of high-level tasks to be executed and goals to be achieved. It is desirable for autonomous systems to have the same level of flexibility which requires on-line planning and execution. Given the time-scale of such plans, and the unpredictability of the environment, autonomous systems must also be able to assess and verify plans and policies at run-time and replan when needed. Finally, these systems may have to learn the environment after deployment. Learning changes the internal models of a system leading to a new system. This is typically not acceptable by current safety standards, and new tools are required to assure safe run-time evolution.

## III. Requirements Elicitation, Validation, and Verification

In this section we describe a methodology to drive requirement definition for autonomous systems. We do not address a comprehensive list of requirements such as operational requirements, maintainability, sustainability, security, cost and durability. We also don't address completeness. Rather, the

focus of this section is on requirements that are generated from an analysis of the functional architecture of a typical system as shown in Figure 1. A system is a collection of agents, but the abstraction level at which we discuss requirements seats above such internal structure. Independently of the number of agents, an autonomous system has a natural functional decomposition which includes a *situational assessment* function, and a *decision-making* function (see the early work in [8], and our recent work in [18] among many others). The situational awareness function makes sense of inputs to create an understanding of the current state of the world, and to make predictions over a given time horizon. The decision-making function, instead, decides actions to take towards exhibiting certain behaviors and/or bringing about certain states.

For control agents, traditional methods already exist to design high-assurance autonomous functions. These methods rely on the ability to enumerate goals and states, and to prescribe policies that map environment states to actions. Control agents can be found at the lowest-level of the decision hierarchy and provide a set of dependable automation functions or capabilities which we will call *skills*. Skills have a known set of failure modes that can be modeled and exposed to the next level of the decision hierarchy, and taken into account during the planning process. The time-scale at which skills operate is small enough that predictions can be made with a known degree of accuracy. Examples of skills include low-level controls such as the trajectory following capability of an autopilot.

At the higher-levels of the decision hierarchy we find planning and self-evolving agents. At this level, the representation of the state space tend to be large and unstructured, and time scales tend to become longer. As a result, it is in general harder to provide accurate estimates or projections about the world. Consider the case of detecting stop signs on the road by relying on images from a camera [13]. The function that relates pixel values to the presence of a stop sign is not known. Neural networks can be used to learn a correlation between the domain and range of such function, but it is an approximation that may lead to run-time errors. Other examples of this kind abound such as detecting and avoiding other aircraft in the sky, or predicting the intent of humans.

At the fundamental level, an autonomous agent (Figure 1) partially senses the environment state $w \in W$, and uses models to create an internal representation $w' \in W'$. The situational assessment function establishes a relation between the two, that qualitatively can be expressed as $R_{SA} \subseteq W \times W'$. Each estimate $w' \in W'$ corresponds to potentially many actual situations $w \in W$. Because decision-making has only access to the internal representation of the world, defining requirements on $R_{SA}$ becomes essential. Notice that the estimate $w'$ does not need to be accurate, but only accurate enough to guarantee good decisions. The definition of such requirements, therefore, should start from an analysis of the decision-making process and the types of capabilities that the agents may decide to use.

To define capabilities and decision-making, we find inspiration in the capability approach from economics [20]. A capability represents a potential that an agent has to realize a behavior or to achieve a certain state. A capability can be used only in certain contexts, namely when the agent has enough resources and is confident that the capability will succeed in its intent. For example, the capability of moving without collisions requires access to a map of the environment, and enough power to mechanically move the agent. Decision-making is the process of selecting the capability to use in a given state of the environment to achieve goals (that can be desired behaviors of states of the environment). Capabilities can be structured hierarchically where a group of capabilities at level $i$ can be leveraged together in such a way to deliver a capability at level $i+1$.

Decision-making includes planning, execution, and run-time validation. In general, planning results in policies (i.e., when and where to use capabilities) that are executed by the agent, which also uses run-time validation to assess how plan execution is going, and forecast potential issues that may arise in the future. There are two sets of requirements that can be derived from the analysis of the capability model. First, the agent must know if a capability can be used, and if and when it achieves its intended goals – both essential for plan execution, monitoring and validation. These requirements directly apply to situational awareness. Secondly, the agent must be able to deliver a capability at level $i$ using capabilities at level $i+1$. This second set of requirements directly apply to decision-making which has the freedom to select policies using $i+1$-th level capabilities. Feasibility in all conditions as expressed by the $i$-th level capability, as well as optimality while respecting constraints must both be verified. Notice that these requirements are in general hard to verify because it is common to rely on the shape of the cost functions, and on the requirements for using a capability to enforce safety constraints or other metrics. Verification, therefore, must check that under all conditions, and all applicable goals, the computed plan is valid.

Situational awareness requirements are expressed in terms of what needs to be known by the agent. This notion can be made precise in the context of modal logic [10] and its possible worlds semantics. Briefly, relation $R_{SA}$ defines the possible pairs $(w, w') \in W \times W'$. An agent knows that a fact $F$ is true if whenever $F$ is true according to the internal estimate $w'$, then it is also true for all $w \in W$ such that $(w, w') \in R_{SA}$. Clearly, the design of this relation plays a central role in determining what the agent knows. Starting from the capability models, and after collecting requirements regarding what the agent needs to know, an additional design step should be conducted to jointly derive requirements for $R_{SA}$ and inference methods. These requirements ultimately determine the sensor, perception and reasoning capabilities of the system.

Finally, a system may be decomposed into multiple agents. As a result, coordination requirements may be derived to share information and assign roles and capabilities. Coordination requirements may span both situational assessment, planning, execution, and run-time validation. In addition to situational

assessment and decision-making requirements, several other common requirements may need to be derived and assigned to agents, such as the ability to understand goals, to explain the reasons for a failure to find a plan or to achieve a goal, and to be able to achieve a safe state in any state of the world. Furthermore, validation and verification of a plan at run-time should not be based on the same models that are used in planning which could otherwise lead to common-mode failures. Independence requirements between planning and run-time verification should be considered. In the case of learning agents, several requirements should be included such as the ability to identify good events to learn from, to compute the extent of the revisions of the internal knowledge of the agent, to perform the update, and to safely resume operations. The overall requirement is that the system improves, meaning that its performance improves at least in some environments, while safety guarantees remain unaffected.

We conclude this section with a set of promising tools that can be used to model and verify requirements. Compositional frameworks are well suited to capture the requirements outlined above. Contract-based design [4] provides the formal foundation to model "what" a system is supposed to do at its interfaces in terms of assumptions and guarantees. Capabilities can therefore be represented formally as contracts. Relation $R_{SA}$ can also be represented by a contract that establishes a relation between $W$ and $W'$ under given assumptions on the environment. Assumptions and guarantees can be expressed formally using a combination of logics that include doxastic or epistemic modalities as done in previous works [6], [30]. In cases formal models are not available, verification will rely on simulation and testing. This is the case, for example, of the relation between camera images and semantic information such as the presence of a specific class of objects. Even in this case, however, promising tools are emerging in the area of falsification [7]. Finally, assurance for autonomous systems may also leverage qualitative human-driven methods as already done in application domains such as aviation [24].

## REFERENCES


[1] "Statistical summary of commercial jet airplane accidents: Worldwide operations 1959–2019," *Aviation Safety, Boeing Commercial Airplanes, Seattle, WA*, 2019. [Online]. Available: https://www.boeing.com/resources/boeingdotcom/company/about_bca/pdf/statsum.pdf

[2] R. Alexander, H. Asgari, R. Ashmore, A. Banks, R. Bongirwar, B. Bradshaw, J. Bragg, J. Clegg, J. Fenn, C. Harper *et al.*, "Safety assurance objectives for autonomous systems," 2020.

[3] A. Aniculaesei, J. Grieser, A. Rausch, K. Rehfeldt, and T. Warnecke, "Toward a holistic software systems engineering approach for dependable autonomous systems," in *2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)*. IEEE, 2018, pp. 23–30.

[4] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, K. G. Larsen *et al.*, "Contracts for system design," *Foundations and Trends® in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[5] J. C. Brustoloni, *Autonomous agents: Characterization and requirements*. Citeseer, 1991.

[6] L. A. Dennis, M. Fisher, N. K. Lincoln, A. Lisitsa, and S. M. Veres, "Practical verification of decision-making in agent-based autonomous systems," *Automated Software Engineering*, vol. 23, no. 3, pp. 305–359, 2016.

[7] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional falsification of cyber-physical systems with machine learning components," *Journal of Automated Reasoning*, vol. 63, no. 4, pp. 1031–1053, 2019.

[8] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human factors*, vol. 37, no. 1, pp. 32–64, 1995.

[9] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1625–1634.

[10] M. Fitting and R. L. Mendelsohn, *First-order modal logic*. Springer Science & Business Media, 2012, vol. 277.

[11] K. Hao, "Facebooks ad-serving algorithm discriminates by gender and race," MIT Technology Review, 2019. [Online]. Available: https://www.technologyreview.com/2019/04/05/1175/facebook-algorithm-discriminates-ai-bias/

[12] J. Horkoff, "Non-functional requirements for machine learning: Challenges and new directions," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 2019, pp. 386–391.

[13] A. Karpathy, "Ai for full-self driving at tesla," https://youtu.be/hx7BXih7zx8?t=513, February 2020.

[14] A. Lapouchnian, S. Liaskos, J. Mylopoulos, and Y. Yu, "Towards requirements-driven autonomic systems design," in *Proceedings of the 2005 workshop on Design and evolution of autonomic application software*, 2005, pp. 1–7.

[15] A. Lapouchnian, Y. Yu, S. Liaskos, and J. Mylopoulos, "Requirements-driven design of autonomic application software," in *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, 2006, pp. 7–es.

[16] R. G. Michalko, "Electrical starting, generation, conversion and distribution system architecture for a more electric vehicle," Oct. 21 2008, uS Patent 7,439,634.

[17] C. T. Nguyen, "Escape the echo chamber," Apr. 2019. [Online]. Available: https://aeon.co/essays/why-its-as-hard-to-escape-an-echo-chamber-as-it-is-to-flee-a-cult

[18] A. Pinto, "An open and modular architecture for autonomous and intelligent systems," in *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*. IEEE, 2019, pp. 1–8.

[19] A. I. Roadmap, "A human-centric approach to ai in aviation," *European Aviation Safety Agency*, 2020.

[20] I. Robeyns and M. F. Byskov. (2016) The Capability Approach. https://plato.stanford.edu/archives/win2016/entries/capability-approach/.

[21] RTCA, *Software considerations in airborne systems and equipment certification*. RTCA, Incorporated, 1992.

[22] ——, "Design assurance guidance for airborne electronics hardware," *Radio Technical Commission for Aeronautics Inc*, 2000.

[23] SAE, "Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment sae international 12," 1996.

[24] ——, "Guidelines for development of civil aircraft and systems," *SAE International*, 2010.

[25] D. L. Silver and R. Poirier, "Requirements for machine lifelong learning," in *International Work-Conference on the Interplay between Natural and Artificial Computation*. Springer, 2007, pp. 313–319.

[26] Stuart Russell (42m, 00s), 2019. [Online]. Available: https://longnow.org/seminars/02019/feb/25/possible-minds/

[27] E. Vassev and M. Hinchey, "Autonomy requirements engineering: a case study on the bepicolombo mission," in *Proceedings of the International Conference on Computer Science and Software Engineering*, 2013, pp. 31–41.

[28] ——, "Autonomy requirements engineering," in *Autonomy Requirements Engineering for Space Missions*. Springer, 2014, pp. 105–172.

[29] A. Vogelsang and M. Borg, "Requirements engineering for machine learning: Perspectives from data scientists," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2019, pp. 245–251.

[30] M. Wooldridge and A. Lomuscio, "A computationally grounded logic of visibility, perception, and knowledge," *Logic Journal of the IGPL*, vol. 9, no. 2, pp. 257–272, 2001.

[31] M. A. Yahya, M. A. Yahya, and A. Dahanayake, "Autonomic computing: A framework to identify autonomy requirements," *Procedia Computer Science*, vol. 20, pp. 235–241, 2013.