Chapter 5
Verification Challenges for Autonomous Systems

Signe A. Redfield and Mae L. Seto

5.1 Introduction

Autonomy and artificial intelligence are quite different. Autonomy is the ability of a physically instantiated robot (autonomous system) to make decisions and reason about its actions based on its in-situ sensor measurements. The objective is to adapt to changes in itself or other systems it interacts with, the environment it operates in, or its tasking (mission). Artificial intelligence, in the broad sense, refers to abstract capabilities associated with problem-solving and does not necessarily require a reference to the physical world. An autonomous robot might use artificial intelligence tools to solve its problems but it is grounded in the physical environment it shares with other objects. An artificial intelligence itself might use an autonomous robot to implement a solution it devises or to gather data to solve a problem but it does not have to ground itself in the physical world for this. This chapter addresses challenges in transitioning autonomous robots, enabled with autonomy which may have artificial intelligence, from the laboratory to real-world environments.

Robotics has been a recognized interdisciplinary area of study since the mid-1900s. In the 1970s the first wave of industrial robots went from the research community to the factory floors (Engelberger 1974). These robots were automated. To overcome safety issues due to their sensory and processing limitations, they were segregated from their human co-workers in physical safety cages. Even with relatively predictable controllers governing their actions, it was not possible to verify their safety sufficiently to operate near humans. Today, robot systems are more capable (Miller et al. 2011), complex (Ferri et al. 2016), and thus less comprehensible

S.A. Redfield (✉)
U.S. Naval Research Laboratory, Washington, DC, USA
e-mail: signe.redfield@nrl.navy.mil

M.L. Seto
Dalhousie University, Halifax, NS, Canada
e-mail: mae.seto@dal.ca

W.F. Lawless et al. (eds.), *Autonomy and Artificial Intelligence: A Threat or Savior?*, DOI 10.1007/978-3-319-59719-5_5

to non-specialists. Research and development has pushed the boundaries on what autonomy can confer on robots in all environments. However, it has not similarly pushed boundaries for how to certify and assure these robots' functions.

Research addresses user needs at the design stage as motivation for an autonomous robot to address a problem. Aspects peripheral to the problem become a lower priority. However, with increase interest in long duration autonomy (Dunbabin and Marques 2012), complex missions, and driverless cars, one of these peripheral aspects have risen in importance. This is the requirement to verify the safety and bounds on the operational capabilities of autonomous systems.

This chapter introduces autonomy for autonomous systems, verification in general then verification implications for autonomy. Next, verification challenges applicable to most robot operating environments (land, sea, air, and space) are outlined with the simple ground robot as an illustrative example.

5.2 Autonomy

Autonomy adds complexity to autonomous systems which adds expense and uncertainty about the system performance, its safety, and when it should be used. Despite this, there are robot situations where autonomy provides a viable solution. These include situations that involve:

- uncertainty about the environment: for example, in rooms, doors may be opened or closed, they can contain people acting within it
- uncertainty about the robot state within the environment: inaccurate or incomplete sensor data on its self-position so that even with a complete map of the environment, the robot cannot navigate to a desired location, and
- communications latency: the robot does not have a human to interpret sensor data or make decisions in new or ambiguous situations.

Autonomy refers to a category of control mechanisms and behaviors (in the context of the behavior-based robot control paradigm) that provides robustness against this uncertainty and enables the robot to operate with little or no human intervention to interpret sensor data or make decisions.

The following terms are used to discuss elements of autonomous systems.

Definitions

System—immobot,[1] robot, group of immobots or group of robots, centralized or decentralized. The hardware, software, perception, actuation, communications, and decision-making that are abstracted as a unit and act in the world. For example: the robot that turns the doorknob is a *system*, but the doorknob is part of the environment rather than the system. A team of robots with a single stationary

[1] Immobot—a robot that is not capable of moving from one location to another within its environment but is capable of modifying its environment in some way, e.g. a smart house.

computer acting as a centralized controller includes both the robots and the computer in the system. A smart house is a system but the people inside of it are part of its environment. The user interface may or may not be part of the system but the human using it is not. The terms autonomous system, system, autonomous robot, and robot are used interchangeably here.

Autonomous system—a system that makes decisions based on local environmental information and **has an intractably complex interaction with its world (environment)**.

Behavior

1. *(robotics)*—the algorithms, software modules and/or actions of a robot in each context (designed and observed)
2. *(verification)*—the actions of a system in an environment

These definitions are unusually specific; a more typical definition of *autonomous* simply means the system makes decisions about its actions based on local environmental data (IEEE Standard Ontologies for Robotics and Automation 2015a, b). Since the focus is systems with no verification tools, the more specific definition for *autonomous* will be used.

Simple systems can fall into the 'autonomous' category while at the same time, complex ones may not. For example, robotic arms in a factory have their physical structure and/or environment constrained so the verification problem is tractable. Similarly, their instantiated behaviors are not subject to any constraints so their system architects build the autonomy as they see fit. However, the purpose of this chapter is to identify verification challenges for difficult cases where formal methods-based design tools are, for whatever reasons, not feasible. While there is complexity and cost to autonomy its benefits on-board autonomous systems are notable.

5.2.1 Benefits of Autonomy

One reason to deploy a mobile autonomous system for a task is the difficult environment (space, underwater, under-ice, etc.). In dynamic environments, autonomous systems operate with limited human interaction to control complex, real-time, and critical processes over long durations. In addition to enabling operations in adverse environments, autonomy also has the potential for increased capability at reduced operational cost. The number of human operators required, a major cost, is reduced. As well, the reliance on the communications link between the robot(s) and its operators is also reduced. An autonomous system is faster than an operator (especially given latencies due to distance or environment) and can function even when communications with its operator is poor. Communication has a cost (energy, at the very least) and is imperfect as channels can be directionally-dependent, lossy, range dependent, and introduce delays. Autonomy can mitigate some of this compromised

communications. However, another cost of on-board autonomy is in the complexity, reliability, and cost of the verification design and implementation.

Verification addresses whether the autonomy was designed and implemented correctly whereas validation is concerned with whether the autonomy (e.g. a robot behavior) meets its requirements to begin with. Verification is the focus in this chapter. Current verification and validation, or V&V, techniques struggle with existing autonomous systems. For example, in the past, the implementation of embedded systems was conservative and dynamic memory allocation was only permitted at start time. Now, the requirement is to verify and validate autonomous systems that exhibit large sets of interacting behaviors and not all of them deterministic.

Autonomy facilitates the autonomous system adapting to a larger set of situations—not all of which are known at design time. This is a key point as one of the purposes of autonomy is to provide contingencies for situations that cannot be specified precisely a priori. Unfortunately, current verification processes require a complete specification of what the system is expected to do in all situations.

Analytic V&V techniques, and model checking, in particular, can provide solutions to design autonomous system control agents in a more efficient and reliable manner. This can mean earlier error detection and a more thorough search of the space spanned by all performance possibilities (performance space). However, the most suitable V&V approach depends on the autonomy tools used. In addition to purely reactive tools, these can include:

- planners
- executives
- fault detection isolation and recovery (FDIR) systems
- mission-based measurements
- navigation
  - terrain analysis and modeling
  - localization
  - path-planning.

It is expected that autonomy approaches require both verification techniques specific to the approach and those that apply across autonomous systems.

5.3 Verification

Verification tools build an assurance case, a body of evidence that, connected using provably correct assertions, enables one to say, within defined assumptions, that the system has certain properties. These properties define *what is desired* and can involve either safety or security. For autonomous systems, there are three categories of safety: self; objects and people it expects to interact with, and objects and people it is not intended to interact with.

Definitions

Verification and validation process: Firstly, *validate* the match between the purpose for which the system is designed and the system requirements (and presumably generate a model of the system or design a potential solution). Secondly, *verify* that the model/design meets the requirements (works correctly). Third, *validate* the system to be sure it accomplishes the purpose for which it was designed (does what the user needs).

Verification: The process of gaining confidence in the correctness of a design or ensuring that a system has certain properties. The fact that it may require exhaustive proof is a problem associated with verification of autonomous systems.

Validation: This refers to step three in the process above. This is the process of testing to ensure that the design decisions made in the initial validation process, to match purpose to requirements, are correct in terms of the system end-use.

Verification is particularly important as systems transition from the research laboratory because it is a critical element of the certification and accreditation process which gives credibility with potential users. Within research laboratories, verification is important because it enables other researchers to use a given algorithm as a component of their systems without concern about unexplored failure modes. For example, if the objective is to test a robotic path-planner around obstacles, the user wants the robot's obstacle avoidance algorithm to be solid and well-understood. Verification confirms the circumstances under which the obstacle avoidance algorithm fails as well as provides a methodology to assess the merit of the user's path-planner with the integrated obstacle avoidance algorithm. Given that, what are the verification implications of autonomy?

5.3.1 Verification Implications of Autonomy

As one of the verification objectives is to understand what the autonomous system is supposed to do, verification tools assume a system specification exists. However, defining the operational goals of an autonomous system is quite difficult making its verification difficult. Existing research addresses these issues, but there are more unexplored research challenges than there are underway research efforts. Section 5.4 identifies autonomous systems verification challenges and notes those with ongoing research efforts. Specific problems that verification tools like sequence and scenario-based testing could address are described next along with their limitations.

Traditional flight software on unmanned aerial and space systems have two components: the on-board software and a sequence. The on-board software is low-level methods or procedures for commanding aspects of the spacecraft hardware, while the sequence is a time-ordered list of commands where each command maps to a software method. Each command is tested independently. Traditional V&V flight

software on unmanned aerial and space systems achieve verification confidence from testing each sequence before up-linking and executing.

For these cases, potentially unanticipated command interactions and subtle problems are detectable by sequence testing. Sequences have singular execution paths (or at most, a few), which facilitates detailed testing and interaction analysis to focus on just those paths. This is a powerful approach but is only flexible when there are a small number of execution paths and those paths are known in advance. On the other hand, the autonomy of autonomous systems may be parallelized, distributed, and non-deterministic with interactions between commands. Consequently, V&V sequence testing does not work as well with these systems.

Autonomous systems are commanded by high-level goals or autonomic responses rather than explicitly scripted sequences. If the system is controlled by high-level goals, these goals are interpreted and further broken down into lower-level commands. The autonomy planner determines the lower-level actions for the robot to take to achieve its high-level goals. If problems arise during execution the autonomy could take corrective actions and find alternate ways to achieve the goals. In this way, there are many possible execution paths. Thus, it is impossible to identify the few that are actually needed and to exhaustively test those. Additionally, the exact command sequence cannot be predicted in advance without complete environmental knowledge, as the decisions are based on both on-board events and local environmental conditions. Autonomy's value is in its ability to close control loops on-board the robot instead of through human operators. However, strength also makes it challenging to adequately define the behavior specification. Consequently, this means sequence validation approaches do not work as autonomy driven processes are not necessarily sequential or deterministic. As well, the autonomy could be implemented as multiple parallel threads that interact.

In an autonomous system, even a simple one, sequence testing provides some confidence for each of the commands, but it does not address interactions between commands the way it has for scripted flight software sequences. These interactions can be subtle and their results, unexpected. They can also depend on the exact sequencing and timing of prior commands, subtleties of the robot state, the environment it interacts with, etc.

As autonomous systems close control loops and arbitrate resources on-board with specialized reasoning, the range of possible situations becomes exponentially large and is largely inaccessible to the operator. This renders traditional scenario-based testing inefficient and in many cases, is impossible to perform exhaustively. There are also scenarios that cannot predictably occur or deterministically reproduced in regular testing. They include race conditions, omissions, and deadlock. Omissions in the specification are problems like defining how long an agent should wait for a reply to a service request (within the publish-subscribe architecture assumed here) before timing out or pursuing another action. Deadlock occurs when two elements enter an infinite loop in the task allocation process and the process fails to yield a decision. This can happen as a result of a race condition or an unforeseen interaction between software components.

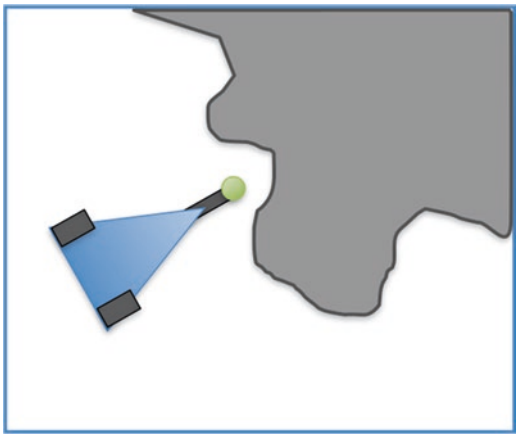

Fig. 5.1 Example robot system—triangular robot (*blue*) with three wheels (*dark gray*) and a downward looking sensor (*green*) in a flat environment (*white*) with cliffs (*light gray*)

A race condition is a behavior in electronics, software, or other system element where the output is sensitive to the sequence, timing, or order of other uncontrollable events. It creates a bug when events do not happen in the order intended. Race conditions can occur in electronics, especially logic circuits, and in multi-threaded or distributed software. An autonomous system can have all of these and is thus prone to race conditions.

Testing for race conditions is not straightforward since certain timing conditions must be satisfied for them to occur and these conditions may not manifest during regular testing. Because of this, regular testing alone cannot assure that race conditions do not exist. To determine whether race conditions exist, formal methods (not discussed here) must be used to model the interaction between agents/threads/subsystems.

In the detailed analysis of verification challenges with autonomous systems, it is instructive to have an illustrative example system.

5.3.2 Example System

The simple robot example, shown in cartoon form in Fig. 5.1, serves to illustrate subtleties that drive the variety of tools and research gaps that exemplify these problems.

This toy system consists of a triangular robot with three wheels and a downward-looking range sensor on a forward telescopic pole to detect cliffs (stairs). The two rear wheels drive the robot. As the robot moves, it controls how far the downward-looking sensor is extended in front by extending or retracting the telescopic pole. Since the robot is physically instantiated it has a non-zero stopping distance. Extending the sensor pole further out allows the robot to detect cliffs earlier. This means it could travel at a higher forward speed. The robot uses dead-reckoning against an internal map to navigate to a waypoint and the downward-looking sensor

to avoid hazards on the way. It operates in a flat world with cliffs but no walls and its only task is to travel from one waypoint to another.

Though this is a simple robot, it provides context to demonstrate some of the challenges associated with autonomous system verification which is discussed next.

5.4 Challenges

The following four categories of research challenges in autonomous systems verification are identified as follows:

- *models:* development of models that represent the system,
- *abstraction*: how to determine the adequate level of abstraction and detail to which requirements are generated,
- *testing*: development of test scenarios, metrics and performance evaluation mechanisms; and the extension of simulations, test environments and tools to enable generalization from tests to conclusions about system performance, and
- *tools*: new tools or techniques that must be developed to enable autonomous systems verification.

The rest of this chapter introduces these challenges in more detail.

5.4.1 Models

With models, there are four identified challenges associated with how to model the autonomous system and the environment it operates in.

Challenge 1: *How is an adequate model of the system created?*

There are several types of models relevant to the verification problem. They include logical models that represent the desired or computed behavior, probabilistic models of the behavior, and mathematical and statistical models of the system. These models must be at a fidelity that captures interactions with the environment and predicts system performance. Software tools such as PRISM can verify behaviors that can be modeled probabilistically (Chaki and Giampapa 2013), but deriving these models and ensuring they represent the behavior is difficult, especially when the verification needs to generalize across environments. Estimations of the conditional probabilities in the model are difficult to arrive at when realistic environmental interactions are considered.

Challenge 2: *Common models and frameworks need to describe autonomous systems broadly enough so they can be used to standardize evaluation efforts and interfaces to the system.*

Beyond models that support verification for systems, models and frameworks (Challenge 1) that support evaluations across solutions are also needed. Such common models and frameworks are being developed from different perspectives. They

range from the development of ontologies and shared concepts describing autonomous systems (Paull et al. 2012) to architectural designs to mathematical models of robot behavior based on dynamical systems. External analysis tools that use variables as independent elements to characterize the system are generally inadequate. Dynamical systems approaches (Smithers 1995) attempt to produce more generalizable models using ordinary differential equations (ODEs) that include a nonlinearities due to time dependencies. However, these approaches, while able to describe some long term emergent behaviors, are not applicable to behaviors that are not modeled by ODEs. Developing models based on tools that measure differences across solutions and how to define a model type that supports evaluations and generalizes across solutions are unsolved problems.

In the toy problem, if the robot's high-level goals are broken down based on a framework like the OODA loop (observe, orient, decide, act), actions for the robot can be specified. Imposing this structure on the autonomous system ensures consistency between evaluation efforts and output standardization. However, a controller may or may not map well into that framework. A deliberative system might explicitly follow each step, while a reactive controller will not explicitly instantiate the 'decide' or 'observe' steps. In the reactive approach, the functions provided by the "decide" and "orient" steps are implicit in the "observe" and "act" steps and cannot be separated. This introduces problems when the framework is used to standardize evaluation efforts, since inaccuracies in the representation can lead to errors in the analysis. If the robot is not deciding, but is instead simply observing and acting, then verification tools designed to analyze the decision-making stage may not adequately capture the relationship between the sensors, actuators, and environment.

Challenge 3: *How should models of black box autonomous systems be developed and debugged? How is a mathematical and/or logical model suitable for formal analysis produced from empirical observations?*

Challenge 4: *How should one identify and model components that are not captured yet (and what are their properties)?*

When there is insufficient knowledge about a system to represent its autonomous behaviors with either logical, probabilistic or mathematical models, it is treated as a black box. Consequently, determining the level of abstraction is almost impossible. In that case, would observing the system's behavior yield sufficient insight into the level of abstraction to model the sensor data? For the example robot, is it sufficient to model the sensor output as a binary (floor/cliff) detection? Or, should the sensor output be modeled as a discrete or continuous-valued function describing the distance between the sensor and the closest object? Should noise, manifested as sensor variations or the frequency which transitions between the binary states occur, be included? Do the motor controllers need to be modeled or is it sufficient to generate a larger model of system actions as a function of sensor input? What principles are used to design a simulation or model, at the level of abstraction needed, to evaluate the feature or property of interest?

5.4.2 Abstraction

5.4.2.1 Fidelity

These challenges focus on the simulation fidelity rather than only the system model addressed in Challenge 1.

Challenge 5: *What determines the level of simulator fidelity to extract the information of interest?*

Insight into the fidelity a simulator requires for meaningful results makes it possible to identify scenarios where the system fails. Searches for scenarios where the system fails can be automated by developing adaptive learning techniques to focus simulations in performance space regions where failure is suspected (Schultz et al. 1993). However, these techniques are only partially effective. Along with development and tuning of learning algorithms, appropriate performance metrics to drive the learning process are needed. These learning techniques and performance metrics could also be used to identify which of several potential levels of fidelity capture the most failure modes.

Challenge 6: *How is the level of abstraction determined for the robot model, its behaviors, and the simulation that tests the model? How many environmental characteristics need to be specified? What are the aspects of the environment, the robot, and the autonomy algorithms that cannot be abstracted away without undermining the verification?*

The level of fidelity to model aspects of the environment as well as which aspects should be modeled is unclear.

The model of the autonomous behavior is given. But what is the fidelity of the model for the robot hardware that realizes the behavior? Can friction in the motors be abstracted away? What about other interacting behaviors in the system?

If a path-planning behavior is to be tested, the robot relies on an awareness of its position relative to the desired path or destination. What level of abstraction is adequate to capture that information? When that is known then the level of abstraction for the environment could be addressed.

For the example system, is it sufficient to define an environment "that contains cliffs"? Reaching the given destination implies the robot did not run out of power prematurely. Not falling off cliffs is easier if the cliffs are stair-like, rather than peninsular, since the robot has only one sensor and thus one measurement of cliff location at any time. The orientation of the robot to cliffs it might encounter or whether the road surface approaching a cliff impacts the robot's maneuverability is unknown. How could one verify the robot will be safe (i.e. not fall off a cliff) given its existing behaviors or determine the environment state space boundaries where the robot can be verified safe? Are there other aspects of the environment that affect the robot's performance that should be included in the environmental model or the robot's behavior model?

The task can be constrained so the robot only operates in an environment with stairs—not peninsular cliffs. Modeling the environment as stairs that are

perpendicular or parallel to the robot's travel direction is insufficient. However, including all possible orientations does not scale up to handle more complex environments. One cannot abstract away stair orientation if the objective is to characterize and model the robot's behavior near stairs. However, the sensor uses sound to detect its range to the floor so it is fine to abstract away the stair's color. The width of the stairs may affect the robot's ability to reach its destination before it runs out of power. Can the width of the tested stairs be bound?

Since the sensor is centered in front of the robot, some autonomous behaviors are likely to have a fault mode where the robot is approaching stairs at an acute angle. How would other locations in the robot state space, which may be fault modes, be identified? These fault modes are functions of the robot's physical configuration. For example, the separation of the rear wheels affects the angle when the robot falls off the cliff before it senses it.

5.4.2.2 Requirements Generation

Challenge 7: *Where is the transition from specifying system requirements to designing the system and how are principled requirements developed so they do not devolve into designing the solution?*

There are efforts towards requirements generation for autonomous systems (Vassev and Hinchey 2013), but they apply to space missions and highlight a problem with defining requirements for autonomous systems: defining the requirements often results in designing the system.

This is particularly noticeable in systems engineering requirements generation. Within the DoD Systems Engineering Fundamentals text (Defense Acquisition University Press 2001), IEEE Standard P1220 is quoted as defining a set of 15 tasks in the requirements generation process. Of these 15 tasks, one represents the desired capabilities of the system (the *functional requirements* which define and constrain the autonomy), one consists largely of elements that an autonomy designer would expect to be part of the design process (the *modes of operation*), three are currently unsolved research problems due to the inability to adequately define, in a testable and achievable way, what the robot ought to be doing (the *measures of effectiveness and suitability*, the *utilization environments*, and the *performance requirements*), and the rest define the context the autonomy is expected to operate. While they impose requirements on the autonomy, these additional constraints are not autonomy requirements themselves. In exploring the functional requirements generation process one finds the functional analysis stage encompasses the autonomy design process.

With the example system, the high-level requirement might be "the robot shall successfully reach its destination in an environment that contains cliffs". But even simply specifying the lower level requirements becomes rapidly difficult.

If a behavior is specified for the robot when it detects a cliff, it defines the system autonomy, not a functional or safety requirement. Sub-requirements of "the robot shall not fall off cliffs", "the robot shall reach its destination" and even "the robot

shall not take longer than X to reach its destination" could be specified, and have the autonomy balance the competing requirements. However, defining requirements below this level, again, quickly falls into designing the autonomy.

Different structures and models have been proposed to describe autonomous systems but none are widely accepted. There is no common taxonomy of goals that describes robot behavior and the goals of these systems. Typically, there are two ways these develop in a field as it matures: either they develop organically over a long period as different ideas are proposed, tested, and refined or rejected or, an organizing body selects or generates a standard to support and accelerate its acceptance.

Standards are being defined to support descriptions of both the hardware (IEEE Standard Ontologies for Robotics and Automation 2015a, b) and the tasks the systems are expected to accomplish, but this field is quite broad and there is little consensus on what tasks should be included or how they should be described to support their eventual implementation and use.

Challenge 8: *How is it ensured that the implicit and the explicit goals of the system are captured? How is a model of the system goals from a human understanding of the task goals, the system, and the environment created?*

To verify the system, implicit goals must also be captured in addition to the explicitly defined task goals. If the explicit goal is for the robot to gather information about a region, the implicit goal is to ensure the information returns to the operator. If the robot gathers the information but does not return it to the user, then as far as the user is concerned, the information has not been gathered.

Challenge 9: *How are performance, safety, and security considerations integrated?*

In the certification and accreditation communities, performance, safety and security considerations are separated. The safety ones are addressed by the certification authority and the performance and security ones by the accreditation authority. One of the major reasons autonomy is implemented on a system is to provide an extra layer of assurance for safety and security as the robot attempts to meets its performance requirements.

For the toy robot, safety includes "not falling off a cliff". If its task is "get from point A to point B", system safety is an implicit performance requirement, since falling off a cliff prevents the robot from reaching point B. If cliff locations are completely known, autonomy is not needed as the solution is to automate the optimal paths between a variety of A's and B's to avoid cliffs. Autonomy is needed if the cliff locations relative to the robot's actual path are not completely known and the desire is to react safely if it detects one. Safety is one of the reasons autonomy is in a system, and being able to avoid cliffs increases overall performance. In this case, safety is part of performance. If other aspects of safety are considered then, safety would include potential damage to the environment as a side effect of falling off the cliff (environmental safety) and potentially injured bystanders if the robot drives into or over a bystander's foot (bystander safety). The safety of the operator is not a consideration for this robot since the operator's interaction with the robot is minimal.

While these aspects do not directly affect the system performance, they do interact with its algorithms—an obstacle avoidance algorithm protects both bystanders and the robot itself while significantly affecting its ability to accomplish its task. Avoiding cliffs promotes safety for the robot and for the environment while improving its performance.

5.4.3 Test

Challenge 10: *At what point is there enough evidence to determine that an autonomous system or behavior has been verified?*

Even outside the robotics research community, the actual measures used to determine when enough evidence has accumulated to verify a system are often ad hoc or arbitrary. Since the original metrics lack firm principles or foundations, it is impossible to find a principled mechanism to support extending them to include autonomous systems. Just as there are no principled methods to determine what evidence is appropriate, there is no easy way to determine when sufficient evidence has been accumulated.

Challenge 11: *How does one ensure it is possible, in the physical world, to test simulated situations that result in boundary cases?*

This is a problem when a fault mode is triggered by a specific sequence of previous actions. There are trivial examples where the toy robot falls off a cliff if initialized in an untenable location, but setting up a physical environment where the robot will be induced to perform the same sequence of actions that lead to that failure is non-trivial. Without being able to repeatedly trigger the same failure in the physical world, there is no confidence that an applied remedy would be effective.

5.4.3.1 Scenarios

Challenge 12: *How would tests be designed so that passing them indicates a more general capability?*

NIST (National Institute of Standards and Technology) developed a suite of robot test arenas in their work with first responders and competition developers in urban search and rescue tasks (Jacoff et al. 2010). In this approach to capability evaluation, systems are tested on their ability to repeatedly accomplish a task within a given arena. Performance is based not only on simple binary accomplishment metrics but on reliability and robustness. This is an extreme version of the most common method developers use to engender confidence in their systems: ad-hoc execution of select design reference missions. Instead of developing an entire scenario, the NIST approach allows developers to test their systems on one capability at a time.

It is more common for developers to test their systems against the entire mission it was designed to address. The mission is intuitively representative of a use case for

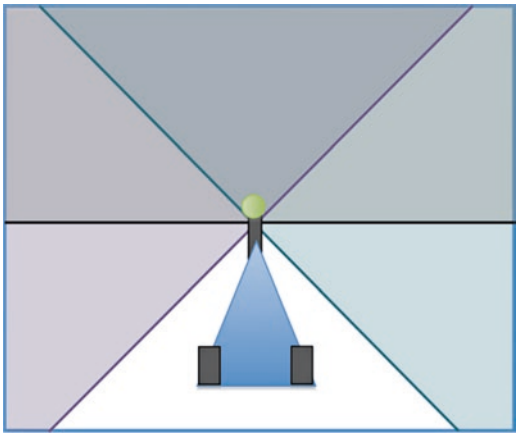

Fig. 5.2 Generalization of environments: if the robot can avoid cliffs that are straight and appear perpendicular (*gray*), at +45° (*purple*) or at −45° (*teal*), it is not imply it can avoid cliffs that cut through the *white* region

which the system was designed. In the best case, it would be a particularly challenging instance of the use case. The implication is that since the system can handle the demonstrated case it will also be able to handle other, similar cases the system will be subjected to operationally.

As an example with the toy robot, if the robot can avoid straight line precipices that are perpendicular, and at 45° (purple and teal lines in Fig. 5.2) to, its direction of travel (black line in Fig. 5.2) then it is valid to generalize and assume all orientations between perpendicular and 45° are likewise acceptable, as are orientations between perpendicular and −45° (all shaded regions in Fig. 5.2). However, it does not imply whether it will succeed against other cliff orientations (areas that cut through the white region in Fig. 5.2), other than to recognize that there is a point where it will lose stability before it detects the cliff.

While test cases demonstrate possibilities the challenge that autonomous robotics now faces is to produce test schemes that provide results which can be meaningfully generalized not only for specific capabilities but for system performance. Efforts have been made to address this challenge using automation to simply execute and analyze many scenarios (Barltrop et al. 2008; Smith et al. 1999; Pecheur 2000), but in each case these efforts required insight into the system under test, and the automation was still based on scripted scenarios that engineers deemed likely and not unanticipated ones.

Challenge 13: *How are challenging design reference missions selected so that performing well against them indicates a more general capability for the system rather than for specific system components?*

Even after generalizing from specific scenarios to regions of capability within the robot state space, methods are still needed to identify specific scenarios that provide the most general information.

In the toy example the environment was implicitly limited to only straight-line cliffs and perfect sensor or actuator performance between ±45°. There was no mention of unexpected obstacles or materials and conditions that cause errant sensor readings—all of which are sources of undesirable behaviors in autonomous systems.

For the toy example, a simpler scenario with cliffs oriented only perpendicular to the robot travel direction and no obstacles provides less information about the behavior robustness than a scenario with approaches to cliffs over a range of orientations and moving objects in the environment.

Challenge 14: *How can test scenarios be produced to yield the data required to generate mathematical/logical models or to find the boundary conditions and fault locations in the robot state space?*

This challenge addresses two points: (1) the development of test protocols and methodologies whose goals are not to evaluate the system but to generate a model of the system from external observations of system properties (flip side of Challenges 3 and 4), and (2) identify test scenarios in the robot state space that exist at performance boundaries. For example, the edge of the curb running along a sidewalk is a performance boundary for a robot that operates on sidewalks. In a simulation, this might be represented as a distance-from-sidewalk-edge parameter or as a position-on-the-map environmental feature, but in either case, it is a performance boundary.

Instead of test scenarios that characterize how well the system works, the purpose of these scenarios is to highlight both areas where the system fails to perform as expected and areas where there is a transition from one performance regime to another. As well, what techniques are needed to determine the parts of the performance space that should be characterized in a model of the black box autonomous system?

5.4.3.2 Metrics and Performance Evaluation

To evaluate the autonomous system using abstracted models, metrics, and measures that are proxies for the system, goals need to be defined. In some cases, these may be represented in the requirements, but the problem of defining metrics associated with implicit and less tangible goals is still difficult. Most work in this area focused on developing tools to measure the degree of autonomy in a system, rather than the effectiveness of the autonomous system as it attempts to accomplish its tasks.

Challenge 15: *Once an adequate model is created how is it determined whether all resulting emergent behaviors were captured and what are appropriate performance measurement tools for this?*

Most formal attempts to provide standards for autonomy have centered on the definition of "levels of autonomy", an effort to distinguish systems by their degree of independence from a human operator and level of capability. Examples include Draper Labs' 3D Intelligence Space (Cleary et al. 2001), the US Army's Mission Performance Potential (MPP) scale (Durst et al. 2014), the Autonomous Control Levels (ACL) put forth by Air Force Research Labs (Sholes 2007), and the National Institute of Standards and Technology's (NIST) Autonomous Levels for Unmanned Systems (ALFUS) (McWilliams et al. 2007; Huang et al. 2004, 2005), shown in Fig. 5.3.

Not only are different approaches largely incompatible with each other, even experts disagree on the taxonomy to categorize a system within a given approach.

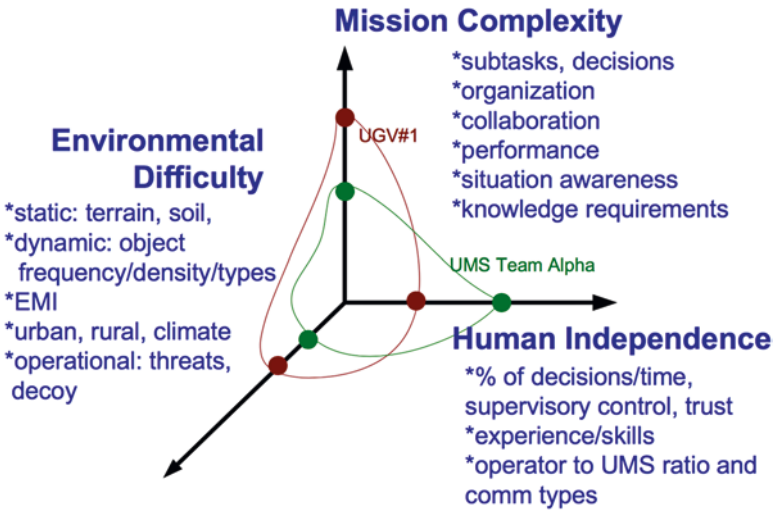

Fig. 5.3 The ALFUS framework for autonomy levels (Huang et al. 2005)

The overall effort to define levels of autonomy has devolved into a philosophical argument, and a 2012 Defense Science Board report recommended that the effort be abandoned in favor of the definition of system frameworks (Murphy and Shields 2012). The levels of autonomy used to certify autonomous systems are the exception because they only attempt to define specific characteristics relevant to their domain of interest.

To illustrate the difficulty in applying a subjective measure of autonomy to a robot, consider the toy example against the two of the axes of evaluation common to most levels of autonomy (Fig. 5.3)—situational awareness/environmental complexity and decision making/human independence. One might argue that the situational awareness of the robot is limited because it is only able to sense its own location and any cliffs in its immediate vicinity. However, it can also be argued that, for the intended environment (only has cliffs), this is all it needs, and the situational awareness is therefore, high. Likewise, since the only stipulated ability is to navigate to a destination and avoid cliffs, in the space spanned by all possible behaviors for all robots this is limited in its independence capability. On the other hand, waypoint following allows the robot to operate independent of a human operator in the traversal of those waypoints, so it could also be considered highly independent.

IEEE Standard 1872–2015 (IEEE Standard Ontologies for Robotics and Automation 2015a, b) attempts to introduce clarity by defining autonomy as a role taken on by a process within the robot. Instead of attempting to define the system autonomy it allows the designer to make some aspects autonomous (e.g. avoiding cliffs) and others automatic (e.g. following a fixed sequence of waypoints).

Challenge 16: *Measurement and evaluation are generally poorly understood—operators can describe tasks for the robot but lack tools to quantitatively evaluate them. How should autonomous behaviors be*

measured so they consistently and accurately describe the capability embodied by a robot?

Efforts to create metrics generally result in tools with solid theoretical foundations that are not easily implemented or, focus on subjective evaluations from experts like system users (Steinberg 2006; Durst et al. 2014), and consequently, not easily comparable across experts. Standardized questions and forms use scales and techniques in an attempt to normalize subjective results (Billman and Steinberg 2007). However, the problem is the inability of evaluators to agree on the relative importance of subsidiary details within a task (e.g. whether the smoothness or the directness of the robot's path is more important) rather than the adequacy of the evaluation tools.

Challenge 17: *How is a metric defined for comparing solutions?*

Even if a metric is defined to evaluate whether a robot accomplishes its task, how are the different solutions compared? Start with the toy problem task: reach waypoint B by a given time. Two robots have the downward-looking sensor that identifies cliffs. Robot A has a sensor that tells it range and bearing to the waypoint and Robot B has a map and the ability to localize itself within it. Robot A uses simple heuristics that cause it to head straight towards the waypoint when there are no cliffs and to back up and turn a fixed amount when there is a cliff. Robot B has a more sophisticated behavior to characterize the cliff. Robot A has a motor controller that imposes smoother motion, while robot B's controller can stop abruptly and turn about an axis internal to itself. Would the metric to compare the solutions be based on how fast the robots reach the waypoint, or is it a function of the smoothness of the path? Is it a combination? If it is a combination, how are the metrics weighted? Is it measured with the same start and destination point every time or, is it sufficient to measure multiple random samples or, is the metric a function of path properties and independent of the specific path? Is the metric a simple binary of reached/failed to reach the waypoint? What if the user does not appreciate what the important aspects are? For example, the relative importance of path duration and efficiency or the reliability with which it reaches the destination.

Challenge 18: *How is the "optimal" defined against which the verification is performed? How is the solution shown to be in fact, optimal? How is the performance of the system measured?*

To verify a system one needs to know its properties and what it is supposed to do. The "optimal" solution for verification of autonomous systems can be a computably optimal solution to the problem (though the robot's limitations, environment, or practical considerations may drive it to a less optimal solution) or the desired behavior itself. The difference between the optimal solution and the robot's actual behavior can be a measure of system performance and used to compare against different solutions. Where a computable optimal solution exists, it is possible to determine whether the robot's performance was optimal, but in other cases, performance is more difficult to quantify. The problem is twofold. It is necessary to define what the behavior ought to be, and evaluate that against what the behavior actually is. Optimal can be defined in the context of the specific behavior (the best this robot is capable

of) or the task goals (what is the best that a system could do). General purpose metrics that compare behaviors using either is an active research area.

5.4.3.3 Intersection of Scenarios and Metrics

Challenge 19: *How is the performance from finite samples of the performance space generalized across several variables and parameters?*

This is similar to Challenge 12 (how to select tests that indicate a general capability), but the focus of this challenge is how to generalize performance given finite test results. For the toy robot, it is straightforward to generalize from the ±45 degree tests because the properties of the environment (cliff orientations), robot (nose sensor and wheel locations), and behavior (robot will not move outside the white triangle in Fig. 5.2 when it reacts to a cliff) are known. What is lacking are general principles, best practices, or theoretical structures that help determine how a given performance test result generalizes its performance throughout the robot state space. For example, if the width of the toy robot's wheelbase is changed, the limits on safe orientations to the cliffs changes. However, within this task and robot configuration, the general premise that the physical configuration is related to this aspect of performance holds. How are equivalent premises that hold for other tasks and scenarios determined?

Challenge 20: *Autonomy frameworks are unable to determine whether all the resulting emergent behaviors have been captured or to supply performance measurement tools.*

Even if it is possible to generalize performance samples to a range within a performance regime, the specificity of the samples limits their broader applicability, and thus does not address verification methods for entire systems. As a test scenario is designed to demonstrate one system behavior or feature, others may be simplified or ignored, which limits the broader applicability of the result. The test scenario only captures potential emergent behaviors related to the behavior or feature being tested, not emergent behaviors that occur when multiple behaviors or features interact. Autonomy frameworks define how interactions between the behaviors and functions of the robot are structured, but they do not define how to generate scenarios to measure the reliability of those interactions. Since the interactions between the robot and its environment is intractably complex this is a critical component of any test method since the performance space cannot be exhaustively searched.

5.4.4 Tools

Challenge 21: *What new tools or techniques need to be developed?*

If there are solutions to these challenges, an adequate system model and a reasonable abstraction of the environment for the simulation tools, there are still difficulties. In addition to correct-by-construction tools (Kress-Gazit et al. 1989), other

Fig. 5.4 NIST test arenas for urban search and rescue robots (from NIST's Robotics Test Facility website)

tools are required to: support analysis of black box systems and behaviors; categorize tasks and goals and connect them to platforms, environments, and capabilities; develop performance metrics; support the development and analysis of modelling and simulation approaches, and connect the different approaches to verification into coherence assurance cases. This is an incomplete list—as progress is made on the rest of these challenges, other gaps will be identified.

Challenge 22: *In general, how is the fitness of a physical robot structure for a given task or environment verified (e.g., a robot that cannot sense color or operates in the dark with an infrared sensor is unfit to sort objects based on color)?*

The NIST test arenas (Jacoff et al. 2010) shown in Fig. 5.4 use specific low-level capabilities (or skills) that can be combined to characterize a desired higher-level capability. The capabilities are determined against performance in a test. For example, the robot must manipulate objects with a required robustness in test X and reliably maneuver through environment Y.

Individual robots are tested against the full suite of test arenas and ranked per their performance in various categories (e.g., manipulation or maneuverability). As robots are generally specialized for a given task, once that task has been decomposed into the skills (or arenas) required, the suitability of a given robot for that task can be evaluated.

Although this approach is effective within this task domain, it has two major limitations: the process to define the arenas/skills was lengthy and expensive and the process to decompose tasks into amalgamations of skills is human-centric and does not generalize well from one task to another. Equivalent tests for the low-level skills could be developed so that any robot might be able to express. However, determining a complete set of basic capabilities was established and that each was adequately tested is more difficult.

Challenge 23: *Descriptive frameworks are either too specific and constrain the developer to specific tools when designing the autonomous elements of the system or, too broad and difficult to apply to specific cases. Tools are needed to analyze systems at both the specific and the broad levels.*

A variety of descriptive frameworks have been advanced to describe autonomous systems in a way that facilitates evaluation. However, when the framework follows too closely to a particular implementation, the solution is limited to only systems with that same implementation.

An example of this phenomenon is the application of formal methods to autonomy by simplifying system states and inputs to create a deterministic system. While this provides a verifiable solution, the simplifications limit the system, and the requirement for determinism precludes the use of more innovative techniques such as fuzzy logic, neural nets, and genetic algorithms. Broader models are more widely applicable, such as the classic OODA (observe, orient, decide, act) loop (Gehr 2009), but the lack of specificity makes them difficult to meaningfully apply. Attempts to find a middle ground between these two approaches include the Systems Capabilities Technical Reference Model (SC-TRM) (Castillo-Effen and Visnevski 2009), Framework for the Design and Evaluation of Autonomous Systems (Murphy and Shields 2012), and the 4D Realtime Control System (RCS) Reference Model Architecture (Albus et al. 2000). Each of these frameworks has its supporters and detractors, but no critical advantage has yet pushed one to widespread adoption. Once such a model is found, verification techniques can be developed for classes of components or capabilities rather than for the entire system at once, making the problem more tractable.

Challenge 24: *How is a structured process that allows feedback between the physical/ground truth layer and the formal methods verification tools developed?*

This is a specific tool among many that could be developed for Challenge 21. Formal methods verification tools can provide useful information about guarantees and properties of a given behavior. However, to verify the behavior as instantiated in a physical system, tools are required to enable test results in the physical system

to feed back into the formal verification tools. Then, this enables the formal verification tool results to feed back into the physical system.

Challenge 25: *How to disambiguate between cases where the specification was incorrect (task description abstraction failed to capture a required system action) from those where the environmental model was incorrect (environmental abstraction failed to capture some critical system-environment interaction)? How to identify not just individual situations but classes of situations where the robot fails to be safe or to achieve safe operation (e.g. a front wheel often falls off the cliff but the back wheels never do). How should unanticipated unknowns be accommodated?*

Challenge 26: *If an algorithm, or patch to an existing algorithm, was replaced can it be proven that no new failure modes were introduced without re-doing the entire verification process?*

If the problem to characterize the performance space of the original system was solved, is it possible to characterize the performance space of the new system without running every algorithm and system-level verification test again?

Does making an autonomous system modular reduce the verification burden when an autonomy algorithm component is changed, added or removed?

In the simulation tool the toy robot model sometimes falls off peninsular cliffs. If the robot is not intended to succeed against them, is this happening because the verification failed to realize that peninsular cliffs were not part of the task or because the simulation environment includes physically unrealizable cliffs?

It is important to evaluate the system at different levels of fidelity using analytic, simulation, and physical instantiation. The analytic tools provide confidence the robot will operate well in certain scenarios and poorly in others. The simulation tools, if abstracted to an appropriate level, can run sufficiently quickly to verify the analytic results in the good and poor areas and identify commonalities between failure modes for the boundary regions. The physical testing tools provide a means to explore the impact of the environment and robot physical structure on its performance in those boundary cases.

The two key challenges identified in testing methodology stem from the intractable complexity problem of operating a complex system within a generally unbounded environment. Firstly, how can all possible scenarios be meaningfully sampled to create a representative subset? Secondly, how can these subsets be generalized to provide confidence in cases that were not directly tested?

These challenges focus on aspects of the problem that are the most difficult to address. Autonomous systems are used in dynamic environments which are inherently unpredictable. Bounds or classes of situations can be defined within which the system is expected to operate a priori. The problem identified here is to define classes of situations within which the system demonstrates specific predictable properties. What tools could be developed to examine a large set of test or simulation data and then extract the common feature that is predictive of success or failure, safety or danger? Can tools be created to identify aspects of the environment which were thought irrelevant but are actually important?

5.5 Summary

Within this chapter some pressing verification challenges facing autonomous robotics were identified as important as robots make the transition from the research laboratory to real-world applications (Table 5.1). By identifying these challenges the lack of insight into certain aspects of autonomous systems are highlighted.

Table 5.1 Summary of autonomous system verification challenges discussed

Challenges	
1	How is an adequate model of the system created?
2	Common models and frameworks need to describe autonomous systems broadly enough so they can be used to standardize evaluation efforts and interfaces to the system
3	How should models of black box autonomous systems be developed and debugged? How is a mathematical and/or logical model suitable for formal analysis produced from empirical observations?
4	How should one identify and model components that are not captured yet (and what are their properties)?
5	What determines the level of simulator Fidelity to extract the information of interest?
6	How is the level of abstraction determined for the robot model, its behaviors, and the simulation that tests the model? How many environmental characteristics need to be specified? What are the aspects of the environment, the robot, and the autonomy algorithms that cannot be abstracted away without undermining the verification?
7	Where is the transition from specifying system requirements to designing the system and how are principled requirements developed so they do not devolve into designing the solution?
8	How is it ensured that the implicit and the explicit goals of the system are captured? How is a model of the system goals from a human understanding of the task goals, the system, and the environment created?
9	How are performance, safety, and security considerations integrated?
10	At what point is there enough evidence to determine that an autonomous system or behavior has been verified?
11	How does one ensure it is possible, in the physical world, to test simulated situations that result in boundary cases?
12	How would tests be designed so that passing them indicates a more general capability?
13	How are challenging design reference missions selected so that performing well against them indicates a more general capability for the system rather than for specific system components?
14	How can test scenarios be produced to yield the data required to generate mathematical/logical models or to find the boundary locations and fault locations in the robot state space?
15	Once an adequate model is created how is it determined whether all resulting emergent behaviors were captured and what are appropriate performance measurement tools for this?
16	Measurement and evaluation are generally poorly understood—Operators can describe the tasks for the robot but lack tools to quantitatively evaluate them. How should autonomous behaviors be measured so they consistently and accurately describe the capability embodied by a robot?

(continued)

Table 5.1 (continued)

Challenges	
17	How is a metric defined for comparing solutions?
18	How is the "optimal" defined against which the verification is performed?? How is the solution shown to be in fact, optimal? How is the performance of the system measured?
19	How is the performance from finite samples of the performance space generalized across several variables and parameters?
20	Autonomy frameworks are unable to determine whether all the resulting emergent behaviors have been captured or to supply performance measurement tools
21	What new tools or techniques need to be developed?
22	In general, how do we Verify the fitness of a given physical robot structure for a given task or environment (obviously, a robot that cannot sense color or is operating in the dark with an infrared sensor is unfit to sort objects on the basis of color)?
23	Descriptive frameworks are either too specific and constrain the developer to specific tools when designing the autonomous elements of the system or, too broad and difficult to apply to specific cases. Tools are needed to analyze systems at both the specific and the broad levels
24	How is a structured process that allows feedback between the physical/ground truth layer and the formal methods verification tools developed?
25	How to disambiguate between cases where the specification was incorrect (task description abstraction failed to capture some required system action) and those where the environmental model was incorrect (environmental abstraction failed to capture some critical system-environment interaction)? How to identify not just individual situations but classes of situations where the vehicle fails to be safe or to achieve safe operation (e.g. a front wheel often falls off the cliff but the back wheels never do). How should unanticipated unknowns be accommodated?
26	If an algorithm, or patch to an existing algorithm, was replaced can it be proven that no new failure modes were introduced without re-doing the entire verification process

While there are areas where progress is being made, and a few more with promising directions for future research, there are many problems that are not addressed. As verification of autonomous systems becomes a more pressing need for industry and a more mainstream research topic, we are optimistic that these challenges will be addressed and new tools and principled approaches will become available to support the safe transition of advanced autonomy and artificial intelligence into commercial autonomous systems.

Acknowledgements The authors would like to thank Andrew Bouchard and Richard Tatum at the Naval Surface Warfare Center in Panama City, Florida, for their help with early version of this paper, and the Verification of Autonomous Systems Working Group, whose efforts help define the terminology and identify these challenges. Thanks, are also due to the United States Naval Research Laboratory and the Office of Naval Research for supporting this research.

References

IEEE Standard Ontologies for Robotics and Automation (2015a), IEEE Std 1872-2015, 60 pages.
IEEE Standard Ontologies for Robotics and Automation (2015b), P1872/D3, 55 pages.
Albus, J., Huang, H. M., Messina, E., Murphy, K., Juberts, M., Lacaze, A., et al. (2000). 4D/RCS: A reference model architecture for unmanned vehicle systems version 2.0.
Barltrop, K. J., Friberg, K. H., & Horvath, G. A. (2008). Automated generation and assessment of autonomous systems test cases. Aerospace Conference (pp. 1–10). IEEE.
Billman, L., & Steinberg, M. (2007). Human system performance metrics for evaluation of mixed-initiative hterogeneous autonomys systems. Proceedings of 2007 Workshop on Performance Metrics for Intelligent Systems (pp. 120–126). ACM.
Castillo-Effen, M., & Visnevski, N. A. (2009). Analysis of autonomous deconfliction in unmanned aircraft systems for testing and evaluation. Aerospace Conference (pp. 1–12). IEEE.
Chaki, S., & Giampapa, J. A. (2013). Probabilistic verification of coordinated multi-robot missions. In Model Checking Software (pp. 135–153). Springer.
Cleary, M. E., Abramsom, M., Adams, M. B., & Kolitz, S. (2001). Metrics for embedded collaborative intelligent systems. NIST Special Publication.
Defense Acquisition University Press. (2001, January). Systems Engineering Fundamentals. Retrieved 2016, from http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-885j-aircraft-systems-engineering-fall-2005/readings/sefguide_01_01.pdf
Dunbabin, M., & Marques, L. (2012, March). Robots for environmental monitoring: Significant advancements and applications. IEEE Robotics and Automation Magazine, 19(1), pp. 24–39.
Durst, P. J., Gray, W., Nikitenko, A., Caetano, J., Trentini, M., & King, R. (2014). A framework for predicting the mission-specific performance of autonomous unmanned systems. IEEE/RSM International Conference on Intelligent Robots and Systems, (pp. 1962–1969).
Engelberger, J. (1974). Three million hours of robot field experience. Industrial Robot: An International Journal, 164–168.
Ferri, G., Ferreira, F., Djapic, V., Petillot, Y., Palau, M., & Winfield, A. (2016). The eurathlon 2015 grand challenge: The first outdoor multi-domain search and rescue robotics competition - a marine perspective. Marine Technology Science Journal, 81–97(17).
Gehr, J. D. (2009). Evaluating situation awareness of autonomous systems. In Performance Evaluation and Benchmarking of Intelligent Systems (pp. 93–111). Springer.
Huang, H. M., Albus, J. S., Messina, R. L., Wade, R. L., & English, R. (2004). Specifying autonomy levels for unmanned systems: Interim report. Defence and Security (pp. 386–397). International Society for Optics and Photonics.
Huang, H. M., Pavek, K., Novak, B., Albus, J., & Messina, E. (2005). A framework for autonomous levels for unmanned Systems (ALFUS). Proceedings of the AUVSI's Unmanned Systems North America.
Jacoff, A., Huang, H. M., Messina, E., Virts, A., & Downs, A. (2010). Comprehensive standard test suites for the performance evaluation of mobile robots. Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop. ACM.
Kress-Gazit, H., Wongpiromsarn, T., & Topcu, U. (1989). Correct, reactive, high-level robot control. Robotics and Automation Magazine, 18(3), pp. 65–74.
McWilliams, G. T., Brown, M. A., Lamm, R. D., Guerra, C. J., Avery, P. A., Kozak, K. C., et al. (2007). Evaluation of autonomy in recent ground vehicles using the autonomy levelos for unmanned systems (alfus) framework. Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems (pp. 54–61). ACM.
Miller, S., van den Berg, J., Fritz, M., Darrell, T., Goldberg, K., & Abbeel, P. (2011). A geometric approach to robotic laundry folding. International Journal of Robotics Research, 31(2), 249–267.
Murphy, R. & Shields, J. (2012). Task Force Report: The Role of Autonomy in DoD Systems. Department of Defense, Defense Science Board.

Paull, L., Severac, G., Raffo, G. V., Angel, J. M., Boley, H., Durst, P. J., et al. (2012). Towards an ontology for autonomous robots. IEEE/RSJ International Conference on Intelligent Robots and Systems, (pp. 1359–1364).

Pecheur, C. (2000). Validation and verification of autonomy software at NASA. National Aeronautics and Space Administration.

Schultz, A. C., Grefenstett, J. J., & De Jong, K. A. (1993, October). Test and evaluation by genetic algorithms. IEEE Expert, 8(5), 9–14.

Sholes, E. (2007). Evolution of a uav autonomy classification taxonomy. Aerospace Conference (pp. 1–16). IEEE.

Smith, B., Millar, W., Dunphy, J., Tung, Y. W., Nayak, P., Gamble, E., et al. (1999). Validation and verification of the remote agent for spacecraft autonomy. Aerospace Conference. 1, pp. 449–468. IEEE.

Smithers, T. (1995). On quantitative performance measures of robot behavior. In The Biology and Technology of Intelligent autonomous Agends (pp. 21–52). Springer.

Steinberg, M. (2006). Intelligent autonomy for unmanned naval systems. Defense and Security Symposiukm (pp. 623013–623013). International Society for Optics and Photonics.

Vassev, E., & Hinchey, M. (2013). On the autonomy requirements for space missions. IEEE 16th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, (pp. 1–10).