Testing, Validation, and Verification of Robotic and Autonomous Systems: A Systematic Review

HUGO ARAUJO, King's College London
MOHAMMAD REZA MOUSAVI, King's College London
MAHSA VARSHOSAZ, IT University of Copenhagen

We perform a systematic literature review on testing, validation, and verification of robotic and autonomous systems (RAS). The scope of this review covers peer-reviewed research papers proposing, improving or evaluating testing techniques, process, or tools that address the system-level qualities of RAS.

Our survey is performed based on a rigorous methodology structured in three phases. First, we made use of a set of 26 seed papers (selected by domain experts) and the SERP-TEST taxonomy to design our search query and (domain-specific) taxonomy. Second, we conducted a search in three academic search engines and applied our inclusion and exclusion criteria to the results. Respectively, we made use of related work and domain specialists (50 academics and 15 industry experts) to validate and refine the search query. As a result, we encountered 10,735 studies, out of which, 195 were included, reviewed and coded.

Our objective is to answer four research questions, pertaining to (1) the type of models, (2) measures for system performance and testing adequacy, (3) tools and their availability, and (4) evidence of applicability, particularly in industrial contexts. We analyse the results of our coding to identify strengths and gaps in the domain and present recommendations to researchers and practitioners.

Our findings show that variants of temporal logics are most widely used for modelling requirements and properties, while variants of state-machines and transition systems are used widely for modelling system behaviour. Other common models concern epistemic logics for specifying requirements and belief-desire-intention models for specifying system behaviour. Apart from time and epistemics, other aspects captured in models concern probabilities (e.g., for modelling uncertainty) and continuous trajectories (e.g., for modelling vehicle dynamics and kinematics).

Many papers lack any rigorous measure of efficiency, effectiveness, or adequacy for their proposed techniques, processes, or tools. Among those that provide a measure of efficiency, effectiveness, or adequacy the majority use domain-agnostic generic measures such as number of failures, size of state-space or verification time were most used. There is a trend in addressing the research gap in this respect by developing domain-specific notions of performance and adequacy. Defining widely-accepted rigorous measures of performance and adequacy for each domain is an identified research gap.

In terms of tools, the most widely used tools are well-established model-checkers such as Prism and Uppaal, as well as simulation tools such as Gazebo; Matlab/Simulink is another widely used toolset in this domain.

Overall there is very limited evidence of industrial applicability in the papers published in this domain. There is even a gap considering consolidated benchmarks for various types of autonomous systems.

We further analyse the results of our coding for the sub-domains of robotics and autonomous systems and provide a refined picture of the research landscape as well as strengths and weaknesses in each subdomain.

Authors' addresses: Hugo Araujo King's College London, hugo.araujo@kcl.ac.uk; Mohammad Reza Mousavi King's College London, mohammad.mousavi@kcl.ac.uk; Mahsa VarshosazIT University of Copenhagen, mahv@itu.dk.

Manuscript submitted to ACM

CCS Concepts: • **General and reference → Surveys and overviews**; • **Computer systems organization → Robotic autonomy**; • **Software and its engineering → Software system structures**.

ACM Reference Format:
Hugo Araujo, Mohammad Reza Mousavi, and Mahsa Varshosaz. 2022. Testing, Validation, and Verification of Robotic and Autonomous Systems: A Systematic Review . 1, 1 (May 2022), 62 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Motivation

Robotic and Autonomous Systems (RAS) involve a rich integration of several disciplines such as control engineering and robotics, mechanical engineering, electronics, and software engineering. Validation and verification of RAS entails a non-trivial extension of traditional testing techniques to deal with their multi-disciplinary nature. In particular, for researchers and practitioners from the software testing community, extending the existing software testing techniques to RAS is a challenge that has led to a sizeable literature on proposing and evaluating different techniques and processes. This rich literature calls for a secondary study that puts a structure to this landscape and identifies relative strengths and weakness of available results. The present paper addresses this gap by performing a structured literature survey of RAS testing.

There are a number of earlier surveys on related topics; we provide an in-depth comparison of related work with our survey in Section 2. However, briefly speaking, some of these surveys have a different or more confined scope, e.g., considering machine learning components [79], formal specification and verification techniques [139] or driving data-sets [109], or do not aim to provide a structure overview of the field in order to answer concrete questions for a given audience [20]. To our knowledge, this is the first systematic secondary study that covers the breadth of results in testing RAS (see the Related Work section for other studies with different foci) and moreover, provides an analysis of such results with the aim of characterising the type of techniques, process and analyse their evidence of applicability (in terms of tools and type of case study).

1.2 Scope and audience

Our scope covers novel results (including techniques, process, tools, and applications thereof) that deal with testing robotic and autonomous systems. We call such novel results “interventions”, following the tradition in medical secondary studies, as well as recent systematic reviews in testing [3]. In our terminology, an intervention is “an act performed (e.g. use of a technique or a process change) to adapt testing to a specific context, to solve a test issue, to diagnose testing or to improve testing” [68]. The scope of our survey includes several validation and verification techniques, including physical testing, model-based testing, runtime monitoring, formal verification and model checking.

Our audience are both researchers and practitioners in software and systems engineering. Hence, we perform our analysis from two perspectives:

(1) researchers: to identify strengths and gaps in the research landscape of testing RAS, particularly concerning the traditional software testing taxonomies, are there new challenges not covered by software testing taxonomies, and
(2) practitioners: identify interventions that have the evidence of applicability given the environment and available resources.

We provide a precise definition of RAS in the remainder of this paper, in order to derive rigorous inclusion and exclusion criteria. But in a nutshell, for our interventions to be useful for the intended audience, we confine our scope to those interventions that

(1) address testing the computer systems integrated in RAS (as opposed to only physical, mechanical, or control parts) in their methodology; this is justified by the fact that our intended audience are researchers and practitioners in software and systems engineering,
(2) have some evidence of applicability, efficiency, or effectiveness on RAS; this is motivated by our scope (testing, validation and verification of RAS) as well as our goal to provide evidence of strength (or weakness) for researchers and practitioners, and
(3) take the system-level validation and verification into account and do not focus on a specific unit or component of such systems (e.g., a specific type of learning or planning algorithms or testing of physical or mechanical parts of such systems), this is motivated by the inherent multi-disciplinary of RAS and the requirement for accommodating it for any system-level testing RAS.

Next, we define a number of research questions that help us structure and analyse the existing interventions for the two groups of audience.

1.3 Research questions

As specified above, we would like to review and analyse those interventions that are applicable to testing, validation, and verification of RAS; in particular, we have an emphasis on those intervention that take into account the computer systems in RAS and their interactions with their physical environment and human users. In the remainder of this section and throughout the rest of the paper, we use the term testing to refer to various testing, validation, and verification techniques.

A structured method of testing, validation, or verification is often steered by models, describing the structure or the behaviour of the system under test. The type of models often determines the type of analysis that can be applied and hence, has a far-reaching effect on the applicability and effectiveness of the technique. However, not all included interventions are model-based (or even related to test cases) as we also consider other forms of verification such as runtime monitoring. Moreover, the metrics of effectiveness, efficiency and coverage used to evaluate the system under test and the intervention itself are both a major factor in determining the intervention's applicability and hence, form a major part of our research questions. Finally, the case studies performed to evaluate the technique are a major source of evidence for applicability. Based on these observations, our research questions are specified below:

(1) What are the *types of models* used for testing RAS?
We interpret the word "model" liberally as any information source or domain abstraction that is used to structure or steer the testing process or evaluate the outcome of testing. This helps us understand and decide about the type of abstractions that are commonly used or needed for testing RAS. They help both researchers and practitioners identify the types / aspects of RAS that can be addressed using the current testing interventions and also the type of information that need to be made available in order for these interventions to be applicable. It also points out to aspects of RAS that are currently not covered by the current interventions. In line with the above specified goals, we analyse two types of models: those that address the system under test or its environment, versus models that describe its quality attributes.

(2) Which *efficiency, effectiveness* and *coverage measures* were introduced or used to evaluate RAS testing interventions?
Efficiency refers to the amount of time and resource needed for an intervention to achieve its goal. Effectiveness refers to the type and the number of faults recovered by a testing intervention and coverage refers to any measure that is used to decide the adequacy and the stopping criteria for a testing intervention. Answering these two questions also provide researchers and practitioners with the available evidence for the strength and applicability of the existing techniques, process, and tools.

(3) What are the interventions supported by *(publicly available) tools* in this domain?
Tool support is a key enabler to the application of testing interventions in practice and their integration with other interventions in research contexts. We analyse the literature regarding this research question by providing information about the tooling available for and needed for each intervention; we call the first class of tools, i.e., those tools that are developed to support a particular intervention, *effect* tools, and those tools used and needed for the effect tools to function. The second category, called context tools, provides further information about what is needed for a particular intervention to be automated in its context. We also report about the license information, when available to facilitate decision-making.

(4) Which interventions have *evidence of applicability* to large-scale and industrial systems?
We gather evidence from the reviewed interventions in terms of case studies and classify them into small-scale, benchmarks, and industrial case studies.

1.4 Structure of the Paper

The remainder of this paper is structured as follows. In Section 2, we review related work, with a focus on secondary studies (literature surveys and reviews) on related subject matters. In Section 3, we define the scope of the paper and explain the background to this structured review. There, we report on the core set of results we started with as the seed for our search in order to shape the study. In Section 4, we review the methodology we used for the our systematic review; this include the description of our search and selection strategy, the development of the taxonomy used for coding the results, our data extraction and synthesis methods. In this section, we also reflect on the threats to our study. In Section 5, we present the results of our coding and analyse them to answer our research questions. In Section 6, we reflect on our analysis and provide concrete suggestions for our target audience, i.e., both for researchers and practitioners. In Section 7, we conclude the paper and present some directions of future research.

2 RELATED WORK

There are a number of literature reviews, surveys, and mapping studies conducted which cover different aspects of robotic and autonomous systems. In what follows, we give an overview of the ones that are most related and have the closest connection to our study (in chronological order).

Cortesi, Ferrara and Chaki [55] discuss the features of a number of analysis techniques, namely data-flow analysis, control-flow analysis, model-checking and abstract interpretation. The survey covers features such as automation, precision, scalability, and soundness for these techniques. The goal for the study is stated as providing robotics software developers hints to help choosing appropriate analysis approaches depending on the kind of properties of interest and software system. However, the interventions studied in this paper are not necessarily applied in the robotics domain already. Furthermore, the work is not a systematic review and does not claim providing any coverage on existing work on analysis techniques applied in its target application domain.

Helle, Schamai and Strobel [100] as well as Redfield and Seto [182] provide an overview of challenges in and available techniques and results for testing and verification of autonomous systems. Both studies only sample a small subset of available results and techniques and use them to identify the areas requiring future research. Our findings based on a much larger set, provide a much more refined view about the available interventions and the landscape for future research.

Koopman and Wagner [119] give an overview of challenges in the V model adapted to deal with the problems in the context of autonomous vehicles. The paper identifies five major challenge areas in testing according to the V model for autonomous vehicles, namely, driver out of the loop, complex requirements, non-deterministic algorithms, inductive learning algorithms, and fail operational systems. The paper covers solution approaches that seem promising across these different challenges including phased deployment using successively relaxed operational scenarios, and using a monitor/actuator pair architecture to separate complex automated and autonomous functions from simpler safety functions, and fault injection. Similar to the previous two papers, the work of Koopman and Wagner has a more restrictive scope than the present paper; moreover, the above-mentioned work is not a (structured) review of the literature.

Gao and Tan [79] provide an overview of the state-of-the-art in V&V for safety-critical systems that rely on machine learning techniques (based on deep learning) for autonomous driving. In this work, the researchers first extract a set of studies by conducting a search and identify a set of challenges by reviewing these studies. Then, the validity of the identified challenges is checked by setting up an industrial questionnaire to survey. Furthermore, a set of research recommendations are provided for future work in automated driving based on deep learning. The search query used in this study is more limited than ours in scope, because it focuses on testing for automated driving and deep learning, while we cover robotic and autonomous systems in a much broader sense. The articles covered in this study are published before 2017.

Knauss et al. [115] present an empirical study for investigating software-related challenges of testing automated vehicles. In the work two different kinds of data collection namely, focus groups (including eleven practitioners from Sweden) and interviews (including 15 practitioners and researchers from a number of countries) are used. The work provides insights about challenges such as virtual testing and simulation, standards and certifications, increased need to test nonfunctional aspects, and automation. This work is not a systematic mapping.

Rao and Frtunikj [181] identify three concrete issues regarding assessment of functional safety of neural networks used in automotive industry to initiate the discussion with industrial peers to find practical solutions. The issues include: dataset completeness, neural network implementation, and transfer learning.

Kang, Yin, and Berger [109] provide a survey of publicly available driving datasets as well as virtual testing for autonomous driving algorithms. A detailed overview of 37 datasets for open-loop testing and 22 virtual testing environments for closed-loop testing have been provided. A remarkable aspect of this survey is the involvement of an industrial domain expert. The scope and results of the paper is significantly different from ours: they focus on autonomous driving algorithms, while we include the whole domain of RAS; they focus on datasets and tools, while we focus on interventions and their effects, as well as their tools.

Beglerovic, Metzner, and Horn [20] provide a brief overview of methodologies used for testing in automated driving. The work provides recommendations about promising methodologies and research areas aimed to reduce the testing effort. The authors mention challenges such as complexity of automated driving functions, variation of scenarios and parameters, scenario selection and test generation. Furthermore, the work briefly touches upon validation, supporting tools in the validation task, and standardisation. This paper is significantly different in methodology from ours: it is not a mapping study and does not provide any detail about the coverage of existing work.

Luckcuck et al. [139] provide a survey of formal specifications and verification methods and tools used for autonomous robotics systems. The work covers a range of studies from 2007-2018. In their work, a number of challenges for formally modelling and verifying the environments that the robotic systems operate in in addition to the internals of such systems is provided. Their work differs from ours as it only covers formal specification and verification tools for such systems. Hence, techniques such as (non-exhaustive) testing and simulation are not covered in their work. Also, our work has a different methodological approach in that we pose and answer research questions as the result of our secondary study, while they focus on the literature review itself. We did use the studies reviewed by Luckcuck et al. to validate and refine our search query in the third phase of our research.

Gleischner, Foster, and Woodcock [90] provide an overview of the strengths, weaknesses, opportunities and threats in the application of integrated Formal Methods to robotic and autonomous systems. Some of their findings, such as the gaps concerning evidence of effectiveness and tool support, reinforce our findings and some, such as the challenges in training, are complementary to those of the present paper. We believe some of the complementary findings arise from the general experience and findings about the application of formal methods, which goes broader than the scope of a survey in the domain of robotic and autonomous systems.

Tahir and Alexander [208] perform a systematic literature review on coverage-based validation, verification, and safety assurance techniques for autonomous vehicles. The scope of their survey is much more confined than ours. They do code different coverage criteria as an answer for one of their research question, which has an overlap with our goal with identifying the coverage criteria. We have used their included papers to validate our search query as a part of our third phase methodology.

Rajabali et al. [179] perform an extensive and systematic literature review on software validation and verification for autonomous vehicles. Their scope is more restricted than the scope of the present study, but some of their research questions (such as identifying gaps in the literature) are common to ours. However, their methodology does not involve a detailed taxonomy as in the present study and hence, their conclusions are more abstract and at a higher level. We have also used this recent paper to validate the query and the final set of considered papers in the third phase of our research.

3 BACKGROUND AND RATIONALE

In this section, we provide an overview of the motivation behind this literature survey, and define its domain. Subsequently, we introduce the basic taxonomy that we have extended and adapted for coding the literature. We also review the pilot study that was used to shape our taxonomy (and later validate our search query, presented in the next section).

3.1 Motivation

Based on our study of the existing literature reviews and surveys, we identified the gap for a secondary study that 1) presents a structured review of the existing results on validation and verification of robotic and autonomous systems and 2) targets specific research questions regarding a) the types of models, b) measures of efficiency and effectiveness, c) available tools, and d) evidence of applicability to large-scale and industrial systems.

3.2 Robotic and Autonomous System

There are a variety of definitions for our domain, RAS; these definitions encompass aspects such as autonomy (including high-level decision making and planning), and adaptation (including artificial intelligence and machine learning) and interaction with human users and the physical environment (including perception, actuation, and mobility). In our view, the following definition provides a concise synthesis of these aspects:

> An autonomous system is an intelligent system that is designed to deal with the physical environment on its own, and work for extended periods of time without explicit human intervention. They are built to analyse, learn from, adapt to, and act on the surrounding environment.

This definition is inspired by and merges some complementary aspects in the earlier definitions given by the Royal Academy of Engineering [165] and the National Science Foundation [76]. We emphasise two important aspects of this definition: one is the system-level perspective; hence, modules or units of software and hardware that are not autonomous systems themselves will not be included in our studies; the second important aspect is the interaction with the environment; hence, autonomous systems that work on offline data and do not feature an interaction with their environment are excluded as well.

3.3 Testing and the SERP-Test Taxonomy

In this work, we consider a testing intervention as any structured approach to *validate* or *verify* the quality of a robotic and autonomous system. Validation concerns checking the system specification, design, or implementation against user requirements. Verification concerns checking the system specification, design, or implementation against another piece of specification, design, or implementation. In other words, validation checks whether we have built the right system (for its users), while verification checks whether we have built it correctly (with respect to other specifications and artefacts) [174].

Our classification of testing research is based on the SERP-Test taxonomy [68]. This taxonomy provides a very general framework for classifying and communicating software testing research and has been used and adapted for this purpose across different domains [3, 184]. It serves as a useful tool for researchers and practitioners to select a testing process or technique based on the available resources or the expected evidence of applicability, effectiveness, and efficiency. In SERP-Test, testing research is classified in terms of four facets: intervention, effect, scope, and context. Intervention pertains to the test techniques, their adaptation, and adoption in different context. Effect facet is used to identify the improvement or adaptation in a given practise as well as any insights gained through assessment. The scope specified whether the effect has been materialised in planning, design, execution, or analysis of tests. Context, as its name suggests, specifies the environment where the intervention takes place, in terms of people and their knowledge, the system under test, and the required models and other types of information.

In the next section, we report on the methodology of this study; namely, in Section 4.1 we discuss the seed papers that formed the basis of our search; in Section 4.2, we report on the final inclusion and exclusion criteria; and in Section 4.3, we report on the adapted taxonomy. In Section 4.4, we report on the search query and its validation with respect to the seed papers; Finally, in Section 4.5 we detail our strategy to extract data from the set of included papers.

4 METHODOLOGY

In this section, we present the methodology used throughout our study that encompasses three phases. In the first phase, a pilot study was conducted, in which, we gathered a set of seed papers (Section 4.1), developed a set of inclusion/exclusion criteria (Section 4.2) and refined our taxonomy (Section 4.3). In the second phase (Section 4.4.1), we performed the search, applied the exclusion criteria and coded the selected papers. In the final phase (Section 4.4.2), the search query was validated and refined via an analysis of the secondary studies on the subject and, also, in consultation with domain experts; a new search was performed and additional studies were included for review and coding. Finally, in Section 4.5, we present our strategy for further filtering papers by their content and an overview of the outcomes.

A repository containing artefacts of this study (namely, the seed papers, the result of the searches and the coding) is publicly available[1].

4.1 Seed papers

The set of seed papers contain 26 manually selected studies gathered in consultation with domain experts: three experts from academia with 32, 23, and 19 years of experience and one expert from industry with 26 years of experience in computer systems testing and verification domains. We reviewed this set as a pilot study with the following objectives:

(1) gathering keywords for the initial search query,
(2) sharpening the inclusion and exclusion criteria, and
(3) evaluating and adapting the SERP-Test taxonomy.

4.2 Selection strategy

To set the boundaries for the scope of our study, based on our research questions, we defined and used a set of inclusion/exclusion criteria as follows.

4.2.1 Inclusion Criteria. The criteria considered for inclusion of studies is as follows:

- The topic of the study is on Testing RAS (Robotics and Autonomous Systems),
- The context must consider the cyber and physical aspects of a system (as opposed to only physical, mechanical, or control parts.), and
- Evidence for applicability is provided.

In the scope of our study, *Testing* is interpreted in a broad sense, which includes formal verification techniques, static and dynamic testing, validation and non-exhaustive techniques.

4.2.2 Exclusion Criteria. The studies matching the following criteria are excluded.

- Not available online,
- Not in English,
- Short papers,
- Not peer-reviewed,
- Patents,
- Published before 2008 (in the second phase), published before 2014 (in the third phase),
- Not addressing robotics and autonomous systems,
- No research contributions to testing (incl. validation or verification),
- Only testing units in isolation; not considering the robotics and autonomous systems as a whole. (If the contribution for testing units are not specific to the system considered in the paper and can have applications in the bigger context, we included the study.),

[1] https://figshare.com/s/40bb82bba792d80bdbfa

- The study only considers the physical aspects of the system and not software components,
- Concerning human-controlled systems, e.g., UAVs and robots that are remotely controlled by a human, and
- For papers on the topic of simulation, as there are a large number of studies among the search results which do not have new contributions in the process or technique of testing interventions; we consider excluding such papers unless they provide clear contributions in the context of testing, validation and verification, have available tool, or provide evidence of applicability in industrial context.

4.3 Taxonomy

In order to consistently classify the set of included studies to extract the information required for answering the research questions described in Section 1.3, we follow a modified version of the SERP-Test taxonomy (see Section 3.3). We started with the high-level facets proposed in SERP-TEST taxonomy and throughout a number of iterations we defined and re-defined a number of categories based on the information obtained from coding the included studies. The extracted data from each facet has been used to answer the research questions and to identify strengths and gaps (provided to researchers and practitioners) as part of our analysis in Section 6. An overview of the final taxonomy, based on which the studies are classified, is depicted on Fig. 1.

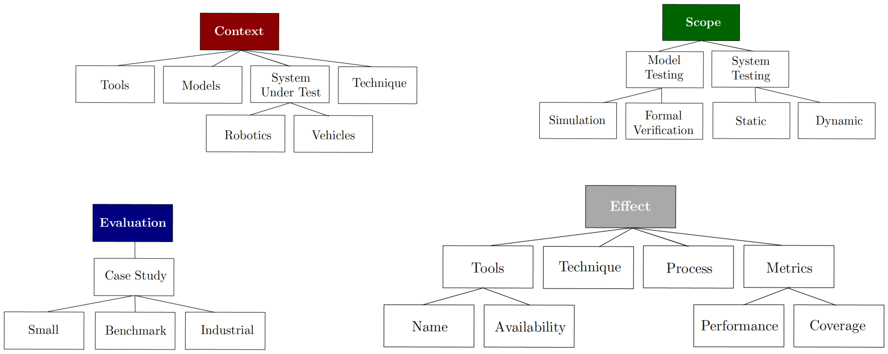

Fig. 1. Illustration of Taxonomy.

As follows, we provide a brief description of our taxonomy.

Context. For context we consider two main categories, namely, system under test and the technique.

- **System Under Test.** System under test describes the type of systems on which the testing technique is applied. In our study, we consider two main categories of RAS, namely, *Robotics* and *Autonomous Systems*. These two categories of systems are selected as they dominate the case studies and a broad range of systems that are considered in the studies concerning testing RAS.
- **Technique.** This is the second category that is considered under Context which represents the testing technique that is improved or affected as a part of the contributions of the work to testing RAS.
- **Models.** Different types of models can be used for describing the behaviour of a system under test. We consider this category to extract the information about the variety of models that are used in the work on testing RAS.
- **Tools and languages.** This category consists of details on tools and languages, under which, the subject systems are described.

Effect. We refine the *Effect* facet (see Section 3.3) further to four categories as follows.

- **Metrics.** This category encompasses the metrics used as a way of evaluating test adequacy or correctness of the subject, based on performance (i.e., efficiency and effectiveness) or coverage measures.
  - **Performance.** This category describes the effect of an intervention on the performance of the testing technique or the subject system. The performance covers a variety of measures concerning safety, quality and resources observed during testing.
  - **Coverage.** This category concerns the measures that indicate how comprehensive the testing technique is once performing in the context of RAS.
- **Process.** This category describes the kind of effects that impact the process of testing technique.
- **Technique.** The technique concerns methods presented as new testing methods or improvements for testing RAS.
- **Tooling** In this category, we extract information about the type of tools that have been used throughout each work. We further classify the tools according to their availability: (1) open source, that are tools for which the source artefacts are available, (2) publicly available, which are tools that are accessible to be used but the source code has not been provided and (3) private, which are tools that have not been made available for download or purchase.

Scope. This facet in SERP-Test taxonomy is further refined to two main categories as follows:

- **Model testing.** This category represents techniques which use a model of the system for testing. We define two sub-categories for such techniques:
  - **Simulation.** This category comprises different types of simulation techniques used for testing RAS.
  - **Formal verification.** This category describes formal verification techniques that use a model of the system to rigorously verify the behaviour.
- **System testing.** This category describes techniques which are applied on actual implementation artifacts of systems.
  - **Static testing.** This category describes techniques which perform testing of system without code execution.
  - **Dynamic testing.** This category describes techniques which check the functional behaviour by executing the implemented code for the system.

Evaluation. We define the case study category which has three main subcategories for this facet in SERP-Test taxonomy (see Sect. 3.3)

- **Case Study.** This category specifies the type of systems that has been used in evaluations of the selected papers. We categorise the case studies into three subcategories, namely, small scale, benchmark, and industrial.
  - **Small.** We consider examples that are developed solely for the purpose of evaluating the method in a specific study and are not applicable for evaluating other similar intervention (due to lack of available details, lack of genericity, or insufficient scale / number of subject systems) as small scale.
  - **Benchmark.** We consider a case study as benchmark if it represents a set of systems with sufficient level of details such that they are / can be used as a point of reference in the evaluations performed in the context of testing autonomous systems.
  - **Industrial.** We categorise a case study as industrial if the subject system is of industrial scale and the evaluation has been performed in industrial context.

4.4 Search strategy

A total of four searches have been conducted. Following after the initial search, three additional searches were conducted to account for our own internal validation and, also, external validation from domain experts. In addition to Google Scholar, two digital libraries, namely, ACM and IEEE, that broadly cover publications with topics in computer science and engineering fields, have been selected as search venue.

4.4.1 Initial query. From the seed papers, an initial set of keywords was extracted to form a search query; additional terms with close meanings and relation to the initial keywords were used to broaden the search. Our query is a conjunction of two main sub-queries: one that comprises terms relevant to our application domain, robotic and autonomous systems, and the other, contains the terms related to testing and verification. The initial query was as follows.

(“Robots” OR “Robotics” OR “Deep learning” OR “Machine Learning” OR “Artificial Intelligence” OR “Robot Simulator” OR “Autonomous Vehicle” OR “Autonomous Vehicles” OR “Autonomous Cars” OR “Image Classification Systems” OR “Neural Networks” OR “Unmanned Vehicles” OR “Unmanned Aerial Vehicles” OR “UAV” OR “Connected and Autonomous Vehicle” OR “CAV” OR “Automated Functions” OR “Drive Assist” OR “Multi-Agent Systems” OR “Autonomous Agents”)
AND
(“Testing” OR “Validation” OR “Verification” OR “Safety Case Analysis” OR “Runtime Monitoring” OR “Robustness” OR “Simulation” OR “Coverage” OR “Metaheuristics” OR “Search-Based” OR “Combinatorial” OR “SMT Solving” OR “SAT Solving” OR “Constraint Solving” OR “Model Checking”)

For this first search, we limited the scope of our search to papers published between 2008 and 2018. Its outcome was a set of 3030 studies.

4.4.2 Validated query. During our validation process, we made use of the seed papers, secondary studies (by checking papers that were referenced amongst included papers but were not an outcome of the search, i.e., snowballing technique) and domain specialists. We approached 50 academics and 15 industry experts in the domains of testing and verification to validate the outcome of the above-given search. They provide expertise in several areas, including verification and validation (31 experts with a median of 18 years of experience), artificial intelligence (8 experts with a median of 12 years of experience), human factors (5 experts with a median of 11 years of experience), and robotics and control systems (9 experts with a median of 14 years of experience). Of that group, we received detailed comments from 8 experts - 7 academics and 1 from industry with an average 18.25 and median 26 years of experience in the field. This resulted in three revisions of our search query.

In the first revision, we included additional keywords ("Robot", "Robotic", "Swarm", "Swarms", "UAVs", "Automated Driving", "ADAS", "Verifying", "Verifiably", "Assurance", and "Assuring"), removed keywords that did not result in coded papers ("Machine Learning", "Deep Learning", "Artificial Intelligence", "Image Classification System", "Neural Networks", "Robustness", "Coverage", and "Combinatorial") and swapped terms for more generic ones ("Autonomous Vehicles" and "Autonomous Cars" were swapped for "Autonomous"). Furthermore, we observed that from years 2008 to 2014, only a handful of papers were included; this led us to further focus the search to papers published between 2014 and 2018.

In the second revision, we added the terms "Driveless" and "Self-driving". Finally, in the third revision, to increase the relevancy of our results, we also included papers from 2019. The consolidated search query is as follows:

("Robots" OR "Robot" OR "Robotics" OR "Robotic" OR "Swarm" OR "Swarms" OR "Autonomous" OR "Unmanned" OR "UAV" OR "UAVs" OR "CAV" OR "Automated Functions" OR "Automated Driving" OR "Drive Assist" OR "Multi-Agent Systems" OR "Multi-Agent System" OR "Driverless" OR "Self-Driving" OR "ADAS")
AND
(""Testing" OR "Validation" OR "Verification" OR "Verifying" OR "Verifiably" OR "Assurance" OR "Assuring" OR "Safety Case Analysis" OR "Runtime Monitoring" OR "Metaheuristics" OR "Simulation" OR "SMT Solving" OR "SAT Solving" OR "Constraint Solving" OR "Model Checking" OR "Search-Based")

The validation process resulted in a total of 7679 additional and unique papers (i.e., the duplicates from the first search were automatically excluded).

4.5 Overview of the results

As discussed in Section 4.4, 3030 papers were obtained as a result of the initial query. As a result of the validation process, we obtained a further 7679 papers. This leads to a total of 10709 search results. Our data extraction methodology consisted as follows.

First, we went through the results and filtered papers based on their title; we obtained a total of 1247 potentially relevant papers. Second, the remaining studies were reviewed by abstract and we applied the exclusion criteria (see Section 4.2), which led to a final set of 195 studies. Third, this final set was coded according to our taxonomy and reviewed in detail as a part of this survey. Figure 2 shows a summary of the number of published articles clustered by year of release. We notice a steady yearly increase of studies included in our review.

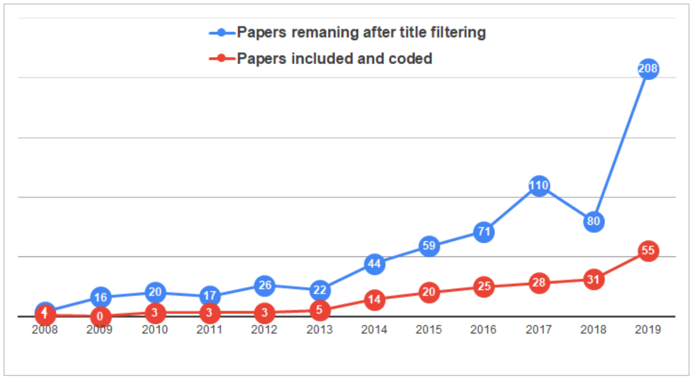

Fig. 2. Relevant and included papers by year.

5 RESULTS

In this section, we present the results of coding the literature in our taxonomy. We structure our results in terms of the four research questions. Regarding RQ1, we present the results concerning the different property specification languages and modelling languages and frameworks used for testing TAS. Regarding RQ2, we review the metrics used to measure the effectiveness, efficiency, and adequacy of testing interventions as well as the quality of systems under test. Regarding RQ3, we code the tools used to implement different interventions as well as any tools implementing the interventions themselves. Regarding RQ4, we present the evidence provided for applicability of the interventions in terms of the case studies and benchmarks used to evaluate the interventions.

5.1 RQ1: Models

In this section, we review the type of models and formalisms that are used for describing the behaviour of robotics and autonomous systems and their properties in testing interventions. Tables 1 and 2, show an overview of results of coding for models used in the studies included in this survey. We classify models according to their semantics (i.e., formal or informal), the domain in which they are employed (i.e., agnostic or domain-specific) and type (i.e., qualitative or quantitative).

We consider a model to be quantitative if it can represent measurable quantities such as probabilities or real-valued entities. Otherwise, the model is considered qualitative. This classification applies regardless of whether the results of the evaluation or the testing technique applied on the model is qualitative or quantitative.

5.1.1 Modelling Properties. Table 1 presents the models that have been used to represent properties and the studies that employ them. Among all studies included in our survey, less than one third use a model or logic to describe the properties of the subject systems. For this set of studies all models are classified as formal. Among those, we notice that, over two thirds employ logics to describe qualitative properties of systems [8, 9, 17, 19, 22, 23, 23, 41, 41, 64, 64, 69, 71, 75, 77, 81, 104, 108, 111, 120–123, 135, 137, 141, 159, 171, 176, 198, 204, 217, 221–223]. Linear temporal logic, first-order logic, and epistemic logic are examples of such logics that have been used in this set of studies. The remaining studies employ logics that can describe quantitative properties, e.g., describing stochastic or temporal aspects of systems [7, 8, 16, 37, 63, 88, 95, 111, 136, 138, 166, 172, 202, 235, 236]. We notice how there is a lack of languages that cater for specific domains; all property languages found in our survey have been domain agnostic.

Table 1. Models for system properties. The table maps the models used to represent properties to the studies that employ them. We classify the models according to their semantics (formal or informal), domain (agnostic or domain-specific) and type (qualitative or quantitative).

		Semantics		Domain		Type			
References	Model	Formal	Informal	Agnostic	Specific	Qualitative	Stochastic	Time	Continues Dynamics
[8]	First-order logic	■		■		■			
[9, 17, 19, 41, 64, 75, 77, 104, 108, 141, 159, 171, 176, 222, 223]	Linear Temporal Logic (LTL)	■		■		■			
[69, 204]	Computation Tree Logic (CTL)	■		■		■			
[41]	ForSALE	■		■		■			
[71]	Property Specification Language (PSL)	■		■		■			
[23, 120–123, 135, 137]	Epistemic temporal logics (ATL, ATLK, ACTL*K\X , IACTLK\X , CTLK)	■		■		■			
[23]	Epistemic strategy logic (ESL)	■		■		■			
[22]	Parametrised Data-Aware Multi-Agent Systems (P-DAMAS)	■		■		■			
[198]	Temporal Logic of Actions	■		■		■			
[217]	TRIO (Temporal Logic)	■		■					■
[64]	Rewriting logic	■		■		■			
[221]	Past time linear temporal logic (ptLTL)	■		■		■			
[7, 37, 172, 235, 236]	Probabilistic Computational Tree Logic (PCTL)	■		■			■		
[136]	Probabilistic Linear Temporal Logic (PLTL)	■		■			■		
[138]	Continuous stochastic logic	■		■			■		
[16, 95, 202]	Timed Computation Tree Logic (TCTL)	■		■				■	
[63, 88]	Signal Temporal Logic (STL)	■		■				■	
[166]	Metric Interval Temporal Logic (MITL)	■		■				■	
[111]	First Order LTL	■		■		■			
[89]	Differential Dynamic Logic	■		■					■
[81]	Graphical Structuring Notation (GSN)		■	■		■			

A review of the results presented in Table 1 shows there is a limited number of studies which consider analysis of properties of systems formulated using formal logics. Furthermore, quantitative properties are considerably less represented in the selected studies. Properties to verify stochastic, continuous and temporal aspects of the systems should play an important role when testing complex and real-time systems, such as RAS. This gap emphasises the need for quantitative logics that are tailored for the domain.

5.1.2 Modelling System Behaviour or Structure. In Table 2, an overview of models used for describing the behaviour or structure of robotics and autonomous systems is provided. Close to half of all of the included studies in this survey, employ system models in their testing strategy; mathematical and rigorously defined models, i.e., formal models, are used in most of such interventions. For instance, Petri Nets and a variety of their extensions [10, 19, 75, 192, 231], labelled transition systems and some of their extended versions [12, 37, 95, 138, 190], finite state machines and their extensions [94, 141, 141], and Markov chains [18, 158, 172, 200, 235, 236] are examples of such models. One observation is that among studies which use informal description of systems, models that are used in Gazebo and on ROS are more commonly used [13, 14, 42, 42, 51, 103, 118, 128].

Some studies employ a combination of models throughout their testing intervention; in particular, for some higher-level models, lower-level models can be used to specify their semantics [47, 157].

Of the studies that consider a behavioural model of their subject systems, around a third utilise qualitative models. Most of such models are employed in formal verification strategies, where correctness is evaluated via mathematical proofs or model checking. The remaining studies use models which describe different quantitative aspects of systems such as temporal and stochastic behaviour, e.g., using variations of Petri nets (e.g., stochastic and coloured) [10, 19, 75, 133, 191, 192, 231], probabilistic timed automata [12, 138], and Markov chains [18, 158, 172, 200, 235, 236]; system dynamics, using differential equations [6, 54, 70, 130, 140, 149, 166, 203], hybrid automata and their extensions [39, 40, 82, 226–228], functional mockup units [1] and various informal simulation models for dynamical systems [13, 14, 42, 51, 103, 118, 128, 173, 201, 233].

Compared to studies before 2019, we notice that there has been an increase in the use of stochastic models (from 4% to 14%). However, this number is still relatively small given the innate probabilistic aspects observed in RAS; hence, this might indicate the need for further stochastic models that are tailored for the domain. Furthermore, we observe a prevalence of qualitative models, despite the importance of quantitative aspects in the behaviour of RAS.

Table 2. Models for system behaviour or structure. The table depicts the studies which make use of models to specify the behaviour or structure of their subject system. Such models are further classified in terms of their semantics (formal or informal), domain (agnostic or domain-specific) and type (qualitative or quantitative).

		Semantics		Domain		Type						
References	Model	Formal	Informal	Agnostic	Specific	Qualitative	Stochastic	Time	Continues Dynamics	Geometry	Arithmetic	Dynamics
[37]	Labelled Transition System (LTS)	■		■		■						
[13, 14, 60, 62, 72, 108, 164]	Belief Desire Intention (BDI)	■		■		■						
[231]	Predicate Transition Re-configurable Nets (PrTR Nets)	■		■		■						
[9, 64, 99, 141, 176, 188]	Finite State Machine (FSM)	■		■		■						
[148]	Prolog (horn clauses)	■		■		■						
[214]	Series-Parallel Action Graphs	■		■		■						
[150]	Alloy	■		■		■						
[89, 219]	Higher Order Logic (HOL)	■		■		■						

Continued on next page

Table 2 – *Continued from previous page*

		Semantics		Domain		Type						
References	Model	Formal	Informal	Agnostic	Specific	Qualitative	Stochastic	Time	Continues Dynamics	Geometry	Arithmetic	Dynamics
[10]	Petri nets	■		■		■						
[133, 191, 192]	Coulored Petri Nets	■		■		■						
[94]	CEFSM - Communicating extended finite state machines	■		■		■						
[193]	Computation Graph	■		■		■						
[104, 222, 223]	Promela	■		■		■						
[202]	Kripke model	■		■		■						
[123]	Global Transition Systems	■		■		■						
[56]	Relational Functional Model	■		■		■						
[17]	Event-B	■		■		■						
[69]	DIME framework	■		■		■						
[206]	Finite State Abstract Model	■		■		■						
[209]	Ontology model	■		■		■						
[147]	Demand Compliant Design (DeCoDe-bsaed model)	■		■		■						
[123]	Open Interpreted Systems	■		■		■						
[232]	SysML	■		■		■						
[89]	Structured Assurance Case Meta-model (SACM)	■		■		■						
[31]	Soar cognitive models	■		■		■						
[18, 158, 172, 200, 235, 236]	Markov chain	■		■			■					
[159]	Markov Decision Processes (MDP)	■		■			■					
[19]	Stochastic petri nets	■		■			■					
[129]	Safety Automata	■		■			■					
[136]	Probabilistic agent templates	■		■			■					
[142, 143]	Process Algebra for Robot Schema (PARS)	■		■			■					
[6, 54, 70, 130, 140, 149, 166, 203]	Differential Equations	■		■					■			
[36]	Real valued functions	■		■					■			
[127]	Knowledge Association Graph	■		■					■			
[57]	Open scenario framework	■		■							■	
[46–48, 157, 183, 205, 229]	Communicating Sequential Processes (CSP)	■		■							■	
[12, 138]	Probabilistic timed automata	■		■			■	■				
[15]	Continuous time Markov chain	■		■			■	■				
[33, 134, 198, 204]	Extended Finite State Machine	■		■				■				
[7, 16, 31, 73, 95, 190]	Timed Automata	■		■				■				
[73, 75]	Timed Petri Nets	■		■				■				
[168]	ALICA	■			■						■	
[71, 108]	Gwendolen	■			■						■	
[60]	ETHAN (Gwendolen extension)	■			■						■	
[101]	Hybrid System model	■			■				■			
[39, 40, 82]	Hybrid Automata	■			■				■			
[226]	Periodic Controlled Hybrid Automata	■			■				■			
[227, 228]	Stochastic Hybrid Automata	■			■		■		■			
[46–48]	RoboChart	■			■		■	■				
[48]	RoboSim	■			■		■	■				
[58]	Meta-model (UML-like)		■	■		■						
[81]	Graphical Structuring Notation (GSN)		■	■		■						
[10]	UML class diagrams		■	■		■						
[163]	Tabular use cases		■	■		■						
[177]	DSLs		■	■		■						
[131, 132]	Stackelberg policies		■	■							■	
[1]	Functional Mockup Unit model		■	■								■
[1]	SystemC		■	■								■
[87]	Klaim		■	■					■			
[237]	Software Reliability Growth Model		■	■			■					

Continued on next page

Table 2 – *Continued from previous page*

		Semantics		Domain		Type						
References	Model	Formal	Informal	Agnostic	Specific	Qualitative	Stochastic	Time	Continues Dynamics	Geometry	Arithmetic	Dynamics
[87]	Mobile stochastic logic (MOSL)		■	■			■					
[25, 30, 162]	Matlab/Simulink models		■	■				■				
[75]	GenoM		■	■				■				
[169]	Configuration Network Language		■	■				■				
[221]	RoboticSpec		■		■			■				
[230]	DSL for cognitive models		■		■	■						
[173]	USARsim (Unreal Script)		■		■				■			
[13, 14, 42, 51, 103, 118, 128]	Unified Robot Description Format (URDF)		■		■				■			
[52]	Geometrical model		■		■					■		
[201]	DSL for traffic scenario		■		■							■
[222, 223]	Brahms		■		■							■
[233]	Roadview models		■		■				■			

5.2 RQ2: Effect

In this section, we review two different types of measures: the first type of measures coded and reviewed in this section are those measures used for evaluating efficiency, effectiveness and coverage of the various testing interventions. The second types of measures are the measures of quality used in testing the subject system; by reusing the terminology we classify them under efficiency (i.e., concerning timing and resources) and effectiveness (i.e., concerning safety and quality) of the subject system.

5.2.1 Measures for Interventions. Table 3 provides an overview of our coding of these measures, classified into efficiency (testing time or resources), effectiveness (testing quality), and coverage (testing adequacy). It is remarkable that about a third of the papers included for this survey used a measure of efficiency, effectiveness, or coverage to evaluate their results. This shows a significant gap in using well-defined measures to evaluate and compare various interventions.

It is also noteworthy that the interventions were measured against a vastly different range of measures. Apart from some very basic notions of efficiency (testing time, or state-space size) [13, 27, 53, 103, 111, 122, 148, 166, 172, 190, 193] and coverage (such as state and transition coverage) [10, 13, 14, 58, 94, 133, 192], most other notions are only used for a single intervention. This emphasises the need for coming up with domain-specific and more sophisticated notions of efficiency, effectiveness, and coverage that are used for benchmarking and comparing various interventions. Some exceptions that concern domain-specific measures are hypervolume (as a domain-specific measure for the searched space) and generational distance (as a measure of distance from optimal solutions) [25], cost of testing for autonomous vehicles in Euros per kilometre [35], feature interaction coverage [2, 33], situation coverage [144], and neuron- [212] and surprise adequacy [113] coverage.

5.2.2 Measures for Subject Systems. In this section, we review the measures of quality for the system under test that are used in various interventions, presented in Table 4. Unlike the previous section, there is more prevalence of domain specific measures; two commonly used measures are spatial distance from the intended trajectory (and variants thereof) [30, 37, 127, 128], collisions and obstacle avoidance [19, 25, 63, 131, 138, 141, 164, 214]. The remaining measures are sparsely used across many different interventions.

Table 3. Classification of measures considered in testing interventions into efficiency (testing time or resources), effectiveness (testing quality), and coverage (testing adequacy)

Measure		References
Effectiveness	Accuracy of the image recognition (failure rates)	[200]
	Hypervolume in fixed time (search-space coverage in time)	[25]
	Feature interaction failure	[2]
	Distance-based surprise adequacy	[113]
	Number and probability of faulty scenarios generated	[156]
	Reachability	[7, 18]
	Number of test cases	[114, 209]
	Number of failures	[176]
	Number of counter-examples	[206]
	Accuracy of the simulation	[199]
Efficiency	Precision	[160]
	Generational distance in time (distance to Pareto optimal solutions in time)	[25]
	Testing (est gen. and exc., model checking) time	[72, 77, 92, 103, 111, 122, 137, 142, 143, 148, 166, 168, 169, 183, 190, 206]
	Test case generation time	[13]
	Test execution and simulation time	[16, 32, 67, 70, 88, 118, 161, 172, 193]
	Reduced test case execution time	[27]
	Testing cost (€/ km)	[35]
	State-space size	[5, 10, 15, 72, 77, 111, 123, 136, 183, 190, 202, 222, 223, 235]
	Search time	[53]
Coverage	Hypervolume	[25]
	Structural coverage metrics (state, code, function, transition, path coverage)	[10, 13, 14, 58, 94, 133, 191, 192]
	Feature interaction (e.g., pairwise and n-wise coverage)	[2, 33]
	Neuron coverage	[212]
	Surprise adequacy coverage	[113]
	Situation (graph) coverage	[144]
	Requirement	[198, 210]
	Diversity	[160]

5.3 RQ3: Tooling

We gather and describe the tools that have been employed and introduced amongst included studies. We categorise tools as context and effect tools; a context tool is one that has been employed by the intervention but it is not a byproduct of its respective work. Effect tools, on the other hand, are the tools that have been developed by the academic community in our list of selected papers.

5.3.1 Context Tools. As shown in table 5, tools for simulation are amongst the most utilised; their usefulness comes from a less costly method of visualising whether the design and process are satisfactory. The middleware ROS [178] combined with the 3D simulator Gazebo [116] form the most popular tool for robotics simulation. Furthermore, Simulink [65], a graphical extension of MATLAB [152], is the most used tool for modelling and simulation of dynamic systems.

In the context of autonomous vehicles, traffic simulators such as SUMO [21] and SYNCHRO [105] have also been employed by included interventions, along with vehicle simulators such as CarMaker [44] and Autoware framework [110].

Moreover, tools for formal verification are also extensively used, with model checkers being the most prominent type. The statistical model checker, Prism [124], provides modelling and analysis of systems of stochastic nature modelled in

Table 4. Classification of measures of quality for the subject systems used in testing interventions.

Measure		References
Effectiveness	Probability of time to collision	[129]
	Performance and safety properties	[70, 177, 221]
	Safety for human operators	[128]
	Satisfied performance properties wrt. number of robots	[12]
	Number of failures	[177]
	Requirements satisfaction	[37, 215]
	Spatial deviation of intended behaviour	[30, 32, 37, 50, 78, 127, 128]
	Endurance distance and stairs traversal of robots	[106]
	Accuracy of the image recognition	[200]
	Collisions & obstacle avoidance	[11, 16, 19, 25, 49, 54, 63, 131, 138, 141, 164, 214, 239]
	Stability	[214]
	Search depth	[71]
	Throughput	[12]
	Schedulability	[73]
	Positive and supportive interactions towards humans	[151, 159]
	Anthropomorphism measure	[189]
	Number of hazards and risk reduction measures	[217]
	Probability of mission success and failure	[15, 142, 143, 159, 169, 227, 228, 235, 236]
	Formal assertions (deadlock freedom, liveness)	[7, 227, 228]
	Criticality (complexity of scenario and dynamics)	[57]
	Vehicle performance (acceleration, speed, position)	[75, 88, 128, 209]
	Regret (Difference between rewards earned and achievable rewards)	[154]
	Severity of failure	[210]
	Probability of rare events	[167]
Efficiency	Number of collisions over time	[164]
	Time to collision	[114]
	Resource utilisation (e.g.: CPU)	[71, 173]
	Network usage	[173]
	Fuel consumption	[70, 138]
	Constraint violation rate	[132]
	Device utilisation	[12]
	Response time	[12]
	Training time	[32]
	Latency	[161]
	Idle time	[6]
	Task completion time	[83, 159]
	Time for hazard identification and risk reduction	[217]
	Median miles to next disengagement	[237]
	Battery life	[235]

markov chains or probabilistic automata. As for qualitative models, UPPAAL [126] offers formal verification for timed automata models that can be, however, extended to employ data types.

Table 5. Tools used in the context of testing interventions for RAS and description of tools.

			Tool Type						
References	Name	Description	Model Checker	Code Generator	Game/Simulation Engine	Theorem Prover	Machine Learning	Test case Genrator	other
[15, 18, 37, 136, 138, 158, 159, 176, 235, 236]	Prism	Probabilistic model checker	■						
[64, 231]	Maude	LTL model checker	■			■			
[7, 7, 12, 14, 16, 31, 73, 95, 108, 190, 227, 228]	UPPAAL	Timed automata model checker	■						
[1, 4, 25, 30, 33, 92, 101, 156, 198, 203, 207]	Matlab/Simulink	System design and simulation environment			■				
[13, 14, 34, 42, 50, 51, 67, 80, 149, 215, 216, 225, 239]	Gazebo	Robots simulator			■				
[96, 112, 170, 201, 207]	Unity-3D	3D games engine			■				
[1, 2, 156]	PreScan	ADAS simulation platform			■				
[164]	Jadex	BDI reasoning engine							■
[141]	JavaMoP	Runtime monitoring framework							■
[11, 13, 14, 34, 42, 50, 51, 103, 128, 149, 185, 220, 225]	ROS	Robotics middleware							■
[150]	Alloy analyser	Analysis of Alloy models							■
[148]	Acceleo	Code generator		■					
[80, 91, 173, 207]	SUMO	Traffic modelling and simulation			■				
[173]	UsarSim	Robot simulation			■				
[63]	Breach	Cyber-Physical Systems simulation			■				
[74, 75]	TINA	Circuit simulator			■				
[75]	Genom3	Robotic components development			■				
[63]	P	Robotic logics model checker	■						
[74, 75]	Fiacre	Formal verification for TOPCASE environment	■						
[10]	Pipe	Petri Nets editor and analyser	■						
[10, 129]	CADP	Design of communication protocols and concurrent systems	■						
[58]	PATeCa	Autonomous vehicles field testing analysis						■	
[46]	Wodel	Mutant generator						■	
[89, 219]	Isabelle	Automated theorem prover				■			
[19]	Cosmos	Tool for statistical models							■
[193]	haROS	Static analysis of ROS application code							■
[202]	SMV	Symbolic model checker							■
[70]	Cplex	Mathematical programming solver							■
[41, 77, 111]	NuSMV	Extension of SMV symbolic model checker	■						
[62, 71, 72, 108]	MCAPL framework	Prototyping and model checking BDI agents	■						
[46, 47, 157, 183, 205, 229]	FDR	CSP model checker	■						
[104]	Spin	PROMELA model checker	■						
[198]	TLC Model Checker	TLA+ model checker	■						
[217]	Zot	TRIO model checker	■						
[108]	Torcs	Car racing simulator			■				
[70]	Synchro	Traffic simulation and analysis			■				
[203]	Simscape	Multidomain physical systems simulator			■				
[1]	Veloce	Hardware simulator			■				
[1]	Simcenter Amesim	Analysis of digital systems			■				
[11, 88]	IPG CarMaker	Autonomous vehicle simulator			■				
[11]	Autoware	Simulation and analysis of self-driving vehicles			■				
[96]	Apollo	Autonomous driving platform							■
[198]	Supremica	Modelling and analysis of discrete-event control functions							■
[101]	Wolfram Mathematica	Computing system							■
[78, 170]	TensorFlow	Machine learning platform					■		
[78]	DeepDriving	Vision-based learning for autonomous cars					■		
[220]	YOLO	Real-time object detection system					■		
[198]	Spark	Programming language and verification toolset					■		
[220]	OpenSceneGraph (OSG)	3D graphics toolkit			■				

Continued on next page

Table 5 – *Continued from previous page*

			Tool Type						
References	Name	Description	Model Checker	Code Generator	Game/Simulation Engine	Theorem Prover	Machine Learning	Test case Genrator	other
[129]	CARLA	Autonomous vehicle simulator			■				
[154]	Multi-Attribute Task Battery II (MATB-II)	NASA workload simulation tool			■				
[69]	Gear	DIME model checker	■						
[142, 143, 169]	VIPARS	PARS model checker	■						
[168]	Clingo	Programming solver							■
[82]	ARIADNE	Reachability analysis of hybrid automata							■
[223]	BrahmsToPromela	Translation tool from Brahms to PROMELA							■
[225]	Orocos toolchain	Creation of real-time robotics applications		■					
[31]	X-Plane	Flight simulator			■				

5.3.2 Effect Tools. A total of 37 tools, publicly available or otherwise, have been introduced by the academic community as an effect of their intervention. Seven of them were not accessible at the time of writing this survey and were classified as private, including SSIM [118], which is a tool for simulating flight software employed in Mars rovers projects. The remaining tools, a total of 30, are available for the general public; 27 of those also have the source artefacts made public and have been classified as open-source. In Table 6, we also inlcuded the specific licence, if any, that the tools are under. We note how the source code of some tools were made available without a licence being specified. In this case, certain repositories, such as GitHub, consider that default (US) copyright laws apply.

Analogously to context tools, we notice a focus on development of tools for simulation and model checking. Tools for testing vehicles are amongst such tools, including in road [4, 54, 78, 166, 233], aerial [67, 140] and maritime [52] environments. As for robots, RoboTool [48] and Improv [6] offer formal verification alternatives for testing robots whilst ROSRV [103] provides a ROS extension for verification at runtime.

Table 6. Tools introduced by studies included in this survey for testing RAS.

References	**Name**	**Description**	**Availability**
[212]	DeepTest	Testing of DNN-driven autonomous cars	Open-source (GPLv3)
[166]	APEX	Formal verification of autonomous vehicle trajectory planning	Private
[75]	Translation tool	Translation tool from GenoM to Fiacre	Private
[233]	Roadview	Traffic scene simulator for Autonomous Vehicles	Private
[46–48, 157]	RoboTool	Formal verification and simulation of robots	Public (No licence found)
[140]	MAV3DSim	Simulation platform for UAV controllers	Public (No license found)
[4]	Florida Poly AV Verification Framework (FLPolyVF)	Verification of the decision making of autonomous vehicles	Open-source (MIT licence)
[118]	Simulator in Julia	Robots simulation	Public (No license found)
[52]	Stonefish	Simulation tool for marine robots	Open-source (GPLv3)
[67]	GzUAV	Framework to run multiple-UAV simulations in Gazebo	Public (No license found)
[54]	Move	Suite of tools to test autonomous vehicles	Open-source (GPLv3)

Continued on next page

Table 6 – *Continued from previous page*

References	Name	Description	Availability
[216]	SSIM	Simulation of flight software	Private
[6]	IMPROV	Tool for self-verification of robots	Public (No license found)
[16]	VerifCar	Framework for validation of decision policies of communicating autonomous vehicles	Public (No license found)
[18]	MCpMC	Statistical model checking of pMC	Open-source
[161]	Asynchronous Multi-Body Framework	Simulation of multi-body systems	Public (No license found)
[53]	RobTest	Tool for stress testing of Single-arm robots	Private
[78]	AsFault	Test case generation for self-driving cars	Public (No license found)
[234]	CyberEarth	Simulation of robots and cyber-physical systems	Public
[37]	Argos	Multi-physics robot simulator	Open-source (MIT licence)
[63]	Drona	Programming framework for robotic systems	Open-source
[103]	ROSRV	Runtime verification framework for ROS	Public (No license found)
[80]	Hybrid Simulation	3D simulation tool	Public (No license found)
[92]	Spot	Prediction of traffic participants	Open-source (GPLv3)
[101]	FROST*	Modelling and simulation of dynamical systems	Open-source (BSD 3-Clause)
[136]	PSV-CA	Probabilistic swarms verifier	Open-source
[176]	RoVer	Model Checker	Open-source (BSD 3-Clause)
[99]	Formal	Modelling and symbolic execution of CPS	Private
[149]	UUV	Gazebo extension for underwater scenarios	Open-source (Apache-2.0)
[185]	V-REP	Robots simulator	Open-source (Commercial or GPLv3)
[213]	MARS	Simulation environment for marine swarm robotics	Open-source (BSD 3-Clause)
[77]	Cruton	Translation from robotics DSL into NuSMV	Open-source (GPLv3)
[160]	Range Adversarial Planning Tool (RAPT)	Test scenarios generation	Public (No license found)
[196]	Pegasus	Autonomous vehicles simulation	Private
[199]	AirSim	Drone simulation environment	Public (Commercial licence)
[225]	Cosina	Simulation of real-time robotics systems	Public (No license found)
[137]	MCMAS	Multi-agent systems model checker	Public (No licence found)

5.4 RQ4: Applicability

Table 7 provides an overview of the case studies conducted amongst included papers. We classify them as small, benchmark, and industrial. Case studies designed specifically to evaluate a particular intervention, which lack sufficient details or generality to be employed for a general class of interventions, were classified as small. On the other hand, those cases studies that are sufficiently general and contain details to evaluate a range of interventions, provided that they are not used in an industrial context, are categorised as benchmark. Industrial case studies are those real-world (and hence, typically detailed and complex) cases conducted in a industrial setting.

Our observation identifies a significant gap in industrial evaluation of interventions; only 20 interventions ([1, 2, 25, 31, 70, 77, 88, 106, 114, 149, 154, 186, 196, 198, 198, 199, 209, 215, 216, 238]) have been evaluated in a industrial context. Understandably, the majority of cases studies have been fully conducted in academic settings. Of those, approximately half made use of small-scale models, which are often not representative of real systems. The other half, employed their proposed interventions on large scale subjects and data-sets, including physical systems.

Table 7. Classification of case studies considered in testing interventions as small, industrial, and benchmark.

Case studies		References
Small	Pedestrian detection	[81]
	Humanised robots	[231]
	UAV	[17, 18, 30, 46, 56, 60, 94, 134, 234]
	Cleaner agent	[164]
	Self-driving vehicle	[148]
	Sensor system	[35]
	Software functions	[218]
	Family of surgical robots	[150]
	Path planning and decision making	[51]
	Surveillance drone	[63]
	Lane-changing scenarios	[80, 166]
	Small robot	[62, 64, 219]
	Simple controllers	[19]
	Unmanned Surface Vehicles	[138]
	USAR robots	[10]
	Cooperative forklifts	[133]
	Agricultural robot	[190]
	Cruise control	[111, 131]
	Traffic environment	[132]
	Mullti-agent manufacturing controller	[229]
	AR.Drone	[42]
	Cooperative UAVs	[104]
	(Industrial scale) transport robot	[12]
	Platoon	[108]
	Robot swarm	[7, 7, 47, 120]
	iCub robot	[172]
	Collision avoidance scenarios	[4, 49, 129]
	Trained gate controller	[123]
	Autonomous vehicles scenarios	[130, 156, 227, 228, 232]
	Footbot	[48]
	Path following autonomous vehicle	[11]
	Autonomous parking	[50]
	Car following	[50]
	Single arm robot	[53]
	Ultimatum game	[189]
	LEGO EV3 robot	[205]
	Simple robot with LiDAR	[206]

Continued on next page

Table 7 – *Continued from previous page*

Case studies		References
	Border control system	[9]
	Military overwatch missions	[142]
	“AMiResot” robot platform	[147]
	Service robot	[168]
	Re-configurable autonomous trolley	[191]
	Surgical robot	[39, 40]
	Cruise control agent	[72]
	Paint spray robot	[82]
	Group of robots	[87]
	Communicating robots	[159]
	Search mission	[169]
Industrial	ADAS System	[25]
	Self-driving system	[2]
	Emergency response robot	[106]
	Test drive in a test track	[70]
	Mars Rover	[216]
	Automated breaking system	[1]
	Lateral State Manager	[198]
	ADAS scenarios	[88]
	Automated Emergency Break	[114, 209]
	NASA benchmark and user case studies	[154]
	RexROV and Desistek SAGA mini-ROV	[149]
	Cartesian impedance Control System in torque mode	[215]
	Care-O-bot	[77]
	Farming	[186]
	Quadrotor with Pixhawk controller	[199]
	Adaptive cruise control	[238]
	Autonomous CoPilot agent	[31]
Benchmark	Autonomous off-road robot RAVON	[177]
	Swarm of robots	[37]
	2 wheels differential drive robots	[128]
	UAV / Land vehicle cooperation	[33]
	Smores	[214]
	Udacity	[212]
	BERT 2	[13, 14]
	MIT and NIRA datasets	[180]
	Traffic sign database	[200]
	Benchmark	[173]
	Carina I	[58]
	Kobuki robot	[95, 193]
	LEGO EV3 robot	[141]
	RMP400 Robot MANA	[75]
	Landshark	[103]
	Alice autonomous vehicle	[226]
	Parallel delta robot	[36]

Continued on next page

Table 7 – *Continued from previous page*

Case studies		References
Benchmark	Jack ROV	[127]
	UAV	[162]
	Quadcopter controller	[74]
	Videos of pedestrians and vehicles	[32, 170]
	Traffic wave observations	[54]
	Leader and follower UAVs	[67]
	ROBNAV mobile robot	[73]
	Udacity, MNIST and CIFAR-10 datasets	[113]
	ATLAS robot	[118]
	Human-robot interactions	[6, 151]
	DaVinci research kits	[161]
	Turtlebot 2	[203]
	ZalaZone Smart City Zone	[207]
	Flexible Manufacturing System (FMS)	[217]
	Drone with Pixhawk flight controller	[220]
	WAYMO public road testing dataset	[237]
	Unmanned underwater vehicle (UUV)	[236]
	Windfarm drone	[235]
	Traffic Scenarios	[92]
	ATLAS and DRC-HUBO robots	[101]
	NAO robot	[176]
	NASA's Unmanned Ground Vehicle	[99]
	Hanse UAV	[213]
	iRobot vaccum cleaner	[15]
	KUKA LWR4+ and the Universal Robots UR5	[34]
	Underwater vehicle	[160]
	Chemical detector	[183]
	COUR-1 robot	[221]
	Care-O-bot	[222, 223]
	COMAN	[225]
	CoCar parking	[239]

6 SUGGESTIONS AND RECOMMENDATIONS TO STUDY AUDIENCE

In this section, we analyse the results of the previous sections in order to identify relative strengths and weaknesses regarding our research questions and for our two target audience groups: researchers and practitioners. We conclude this section by drawing recommendations from our analysis both for researchers and practitioners.

6.1 Analysis

6.1.1 Domain. Table 8 provides a concise summary of the domains covered by the reviewed interventions. A bulk of reviewed interventions do not pertain to any specific sub-domain of RAS. This indicates a clear gap for subdomain-specific research that considers the characteristics of each of these subdomains and takes them into account in their testing interventions. Most importantly, the subdomain of testing marine and submarine RAS as well as space RAS is under-explored (the only included intervention regarding marine and submarine robots [52, 149, 175, 213] and regarding space robots [154, 216] are not represented in the table for the sake of brevity). We note that there is a recently funded European project REMARO to fill in this substantial gap. [2]

Below we analyse the results gathered in Table 8 on a row-by-row basis:

Qualitative Despite the intrinsic quantitative nature of RAS, qualitative models also play an important role in the verification of such systems, particularly when they are abstracted away to be amenable to rigorous and exhaustive techniques in formal verification. In the case of vehicles, qualitative models abstract away from physical dynamics and rather focus on observable behaviour that can be modelled as discrete events. Overall, we noticed a gap in the qualitative analysis that focus on aerial vehicles and mobile robots (excluding road vehicles), this is likely due to the challenge of modelling movement without using continuous dynamics.

Road In the domain of autonomous cars, qualitative models have been used to reason about Human-Machine Interactions [163, 232] and high-level decision-making particularly regarding ethical concerns [60] and safety [81, 200]. most quantitative interventions propose use of formal models in their methodology [58, 60, 71, 72, 148, 163, 198, 206, 230, 232]. For instance, Yun et al. [232] propose an strategy that formalises Human-Machine interaction into SysML language to help steer the testing process. In the same vein, Naujoks et al. [163] provide a DSL using a taxonomy of use cases to cover transitions and modes in Human-Machine Interaction interfaces used in the verification process. Sun et al. [206] employ Satisfiability-Modulo-Convex encoding to build finite state abstractions of the learning component and formally verify it. Dennis et el. [60] focus on formalising and verifying ethical concerns in BDI agents. More informal approaches include the use of 3D simulation by describing scenarios in UML notation modelling [209] and the use of graphical notations for safety assurance analysis [81, 200]. A mix of formal and informal models is employed by Heitmeyer et al. [99] where they provide two new tools to be included into their toolset (FORMAL [98]). The first tool synthesises formal models from scenarios written in Event Sequence Charts and the second tool incorporates a 3D simulation tool (eBotworks [93]) into their toolset.

Aerial In the aerial domain, there are only a handful of interventions that use qualitative models for testing of aerial vehicles, mostly regarding safety and security concerns. For instance, linear temporal logic is used more than once to formalise safety assurance cases [17, 41]. Hagerman et al. [94] make use of finite state machines to extract security test suites and Bhattacharyya et al. [31] focus on formally verifying the boundaries beyond which the agent are designed to operate by translating models from a cognitive architecture (Soar [125]) into UPPAAL.

Mobile Only two studies have been found in this category. Andrews et al. [10] model autonomous systems and their environment using Petri nets to generate test cases and apply their technique to a case study in the human-robot interaction domain. Furthermore, in the context of software product lines, Mansoor et al. [150] conduct a case study on a family of surgical robots by employing formal analysis, feature modelling and testing; they discuss the key challenges and lessons learned from the case study.

Generic Most of the included interventions in this category concerned abstract representations of multi-agent autonomous systems and provided efficient algorithms for parametric (formal) verification or state-space reduction techniques [8, 22–24, 120–123, 135, 137, 168, 188]. Similar to the previous item, most of the interventions

[2] https://cordis.europa.eu/project/id/956200

Table 8. Testing, Validation, and Verification Interventions for Specific Subdomains of RAS

		Vehicles		**Robots**	
		Road	**Aerial**	**Mobile**	**Generic / Non-Physical / Immobile**
RQ1	**Qualitative**	[58, 60, 71, 72, 81, 99, 148, 163, 198, 200, 206, 209, 230, 232]	[17, 31, 41, 56, 94]	[10, 150]	[8, 9, 14, 22–24, 38, 62, 64, 69, 89, 120–123, 133, 135, 137, 147, 153, 164, 168, 188, 192, 193, 217, 222–224, 229, 231]
	Quantitative	[1, 16, 19, 25, 30, 51, 54, 57, 70, 88, 108, 111, 129–132, 134, 156, 158, 166, 171, 173, 201, 207, 226–228, 233, 237]	[18, 33, 42, 63, 104, 140, 162, 202, 235]	[12, 37, 138, 177]	[6, 7, 15, 34, 39, 40, 46–48, 52, 53, 73–75, 77, 82, 87, 89, 95, 101, 103, 118, 127, 128, 136, 141, 142, 149, 157, 159, 169, 172, 176, 183, 185, 190, 191, 203, 205, 213, 216, 219, 221, 225, 236]
	Formal	[16, 19, 58, 60, 71, 72, 99, 108, 111, 129, 134, 158, 166, 171, 198, 200, 209, 226–228, 230, 232]	[17, 18, 33, 41, 56, 63, 94, 104, 202, 235]	[10, 12, 37, 138, 150, 177]	[7–9, 14, 15, 22–24, 38–40, 46–48, 62, 64, 69, 73–75, 77, 82, 89, 95, 120–123, 127, 133, 135–137, 141, 142, 147, 157, 159, 164, 169, 172, 176, 183, 188, 190–193, 205, 217, 219, 221–224, 229, 231, 236]
	Informal	[1, 25, 30, 51, 54, 70, 81, 130–132, 148, 156, 163, 173, 201, 207, 233, 237]	[42, 67, 140, 162]		[6, 34, 52, 87, 101, 103, 118, 128, 149, 168, 185, 203, 213, 216, 225]
RQ2	**Effectiveness**	[2, 11, 16, 19, 25, 30, 32, 49, 50, 54, 57, 70, 71, 78, 113, 114, 129, 131, 156, 160, 167, 198, 200, 209, 210, 227, 228, 239]	[18, 63, 199, 235]	[12, 37, 138, 177]	[7, 15, 66, 73, 127, 128, 141, 142, 151, 154, 159, 164, 189, 214, 215, 217, 236]
	Efficiency	[16, 32, 35, 70–72, 88, 92, 111, 114, 132, 148, 160, 166, 173, 206, 237]	[67, 202, 235]	[10, 12]	[6, 15, 39, 40, 53, 66, 77, 83, 103, 118, 123, 136, 137, 142, 159, 161, 172, 183, 190, 193, 217, 229]
	Coverage	[58, 113, 144, 210]	[33, 94]		[14, 133, 191, 192, 212]
RQ3	**Open-source**	[4, 16, 54, 78, 80, 92, 212]	[18, 67, 140]		[6, 37, 52, 63, 77, 101, 103, 118, 136, 137, 149, 161, 176, 185, 213, 225]
	Public	[160]	[199]		[48, 234]
	Proprietary	[99, 166, 196, 233]			[53, 75, 216]
RQ4	**Small**	[4, 11, 19, 27, 30, 35, 49–51, 60, 71, 72, 80, 81, 96, 108, 111, 129–132, 134, 148, 156, 166, 218, 226–228, 232, 233]	[17, 18, 41, 42, 56, 63, 94, 104, 140, 197, 202]	[12, 138]	[7, 9, 15, 22, 24, 38–40, 46–48, 53, 62, 64, 69, 82, 83, 87, 89, 120, 121, 123, 133, 136, 137, 147, 159, 164, 169, 172, 185, 188–192, 205, 213, 219, 229, 231, 234]
	Industrial	[1, 2, 25, 70, 88, 114, 196, 198, 209, 238]	[31, 186, 199]	[186]	[77, 106, 149, 154, 195, 215, 216]
	Benchmarks	[32, 54, 58, 78, 92, 99, 113, 160, 170, 171, 173, 200, 207, 237, 239]	[33, 67, 162, 220, 235]	[10, 37, 150, 177]	[6, 14, 34, 36, 73–75, 95, 101, 103, 118, 127, 128, 141, 151, 161, 176, 180, 183, 193, 194, 203, 212, 214, 217, 221–225, 236]

used Linear Temporal Logic or variants thereof to model safety properties [89, 223]. Formal modelling and verification of human-machine interaction is also a common theme in this category [69, 222, 223].

Quantitative We see that quantities such as time (representing the real-time behaviour), probabilities (representing abstractions in communication networks or choices made by human actors), and physical dynamics (such as velocity and acceleration) are used for testing in various domains. Most of the developed techniques, such as compositional verification techniques or evaluation of intervention do not pertain to domain-specific instances of these quantities and consider general multi-agent and robotic systems. Below we review domain-specific interventions as well interventions that are developed for the general context of RAS, structured by the columns in Table 8.

Road In this domain we see a strength in integrating code-level abstractions (e.g., for individual components or functions) with system-level specifications (for vehicles and fleets of vehicles and using); such integration are then tested using simulation and formal verification frameworks. The quantitative models used for such testing intervention often pertain to vehicle dynamics and also probabilities arising from communication frameworks. AbdElSalam et al. [1] present a framework for verification of ADAS and autonomous vehicles which uses SystemC TLM models for virtual ECUs. Transaction-Level Models provide a high level abstraction of the SystemC components that are used in virtual ECUs. These models are then integrated with the vehicle and traffic models for simulation. Parametric modelling of CAVs as a network of timed automata is used by Arcile et al. [16]. In this work, VerifCar tool is applied to asses the impact of communication delays on the decision algorithms of CAVs and to check the robustness and efficiency of such algorithms. Similarly, variations of extensions of timed-automata, probabilistic timed automata and stochastic timed automata are used [12, 138, 227] for modelling the behaviour of autonomous vehicles to verify properties of different decision making and collision avoidance algorithms. Barbot et al. [19] use statistical model checking to verify an autonomous vehicle controller where the controller is specified in C++. A set of safety properties specified in HASL, a quantitative variant of linear temporal logic are verified for a controller. Betts et al. [30] compare the effectiveness of two search based testing methods, genetic algorithms and surrogate-based optimisation, for test case generation for UAV flight control software. There are several works in this category which provide and use simulation platforms [51, 51, 70, 88, 129, 132, 134, 173, 207].

Aerial In this domain timing information Human-machine interaction is formalised (through a formalisation of a cognitive architecture) in terms of networks of timed automata and UPPAAL is subsequently used for their verification. [18, 33, 42, 63, 104, 140, 162, 202, 235]

Mobile There are a few studies in this domain that consider quantitative models. Among these studies, the majority employ formal models [12, 37, 138]. Statistical model checking is one of the techniques that is used [12] to verify performance of transport robots based on behavioural models, stochastic timed automata, using UPPAAL SMC. Furthermore, model checking of Markov models is used [37] to verify PCTL properties of swarm robotics behaviour in the design phase. The models then are used as a blueprint for implementation and simulation. Probabilistic model checking of unmanned surface vehicles is another technique used [138]. PRISM model checker verifies PCTL properties of USVs on probabilistic timed automata as behavioural model. Other work in this category [177] use a designed DSL (graph-based models) to describe the system behaviour which is used as test model for generating test cases.

Generic As a general observation a considerable portion of all interventions (23% of papers) in this category have been an out-of-the-box application of model checking tools (mostly using PRISM [124], in some cases UPPAAL [126] and FDR [86]) to specify small-scale robotic case studies; another prevalent category (30% of papers) concerns theoretical papers on various logics and model-checking algorithms for multi-agent robotic systems. Notable exceptions of this general theme include languages and toolsets for rigorous simulation [47, 48, 157, 183] using formal verification as part of the design of robotic interaction protocols [64, 229], using formal verification to analyse human-robot interaction [69, 159, 176, 217, 222, 223], and generating test cases from formal models [14, 46, 133, 164, 193]. Another interesting intervention concerned comparison of different hybrid-systems solvers [39, 40] for formal verification of robotic applications. Also the theorem prover Isabelle/HOL has been used to formalise safety assurance claims [89]. Also model checking has been used to train policies in reinforcement learning [172]. Compositional techniques has been used to reduce the complexity of the model checking problem [171, 190]. Another noteworthy attempt is in formalising and verifying ethical concerns [60, 62].

Formal In summary, there is a relative strength in theoretical foundations of testing, validation, and verification, comprising various logical and specification formalisms and small-scale proof of concept exercise in model checking abstractions of RAS. There seems to be a recent trend, identified below in analysing human-machine interactions. There is a relative weakness in non-exhaustive testing, validation, and verification techniques and studying and improving their applications to large-scale and industrial systems. To give a more nuanced analysis, we distinguish our analysis further based on the domain:

Road There are a number of non-exhaustive testing from formal models [134, 209] and scenario generation [158] techniques proposed in this domain. Model synthesis from scenarios has also been studied [99]. The applications of different verification techniques such as formal verification based on supervisory control, model checking, and deductive verification has been studied in industrial context [198]. Also in this domain, a framework for validating ethical policies has been developed [230] and human-machine interaction for user interfaces have been validated [232].

Aerial Apart from applying traditional model checking [104, 202] and simulation [56] techniques to this specific domain, we observe notable attempts to combat the huge state-space of domain-specific models by employing statistical model checking [18] and run-time monitoring [63]. Most papers in this domain have focused on constructing safety models / properties or even coming up with safety specification frameworks [17, 41]; however, energy-efficiency [235] and to a limited extent, security [94] have been addressed as well.

Mobile The landscape in this domain is much more sparse and scattered. As usual for this category, there are a number of applications of model-checking tools. There is a single paper on test description and test-case generation [177]; also variability is an under-studied aspect in robotics that has been handled in this context [150] and finally, there is an industrial case study on the application of model checking [12].

Generic Formal verification is prevalent in this category; besides parametric verification of multi-agent systems using variations of epistemic logic [8, 22, 24, 122, 136, 188], formal verification using timed automata has also been used in a strategy that composes verification problems into smaller ones [190] and in the verification of path planning [7]. Furthermore, Araiza-Illan, Pipe and Eder [14] use BDI models and model checking of probabilistic timed automata (in UPPAAL) to generate test sequences for human-robot collaboration tasks. Another use of UPPALL in this category is for model checking of ROS applications that makes use of an ad-hoc translation from ROS to UPPAAL [95].

Verification of probabilistic aspects can be found in a few studies. Zhao et al. [236] employ bayesian inference to estimate the distribution of parameters of Markov chains. Then, they combine formal verification, synthesis and runtime monitoring to check that the estimated parameters are not violated. Pathak et al. [172] make use of probabilistic properties of Markov chains for self-repair capabilities in robots and tie those into a formal verification process (of PCTL formulae). Araujo, Mota and Nogueira [15] apply probabilistic model checking to verify whether a robot trajectory (described in terms of an algorithm) satisfies specific behaviours or properties (stated as temporal formulas).

The combination of CSP and FDR can be seen in a few studies. Cavalcanti et al. [46] generate mutations on RoboCart models and feed the mutated CSP [102] (obtained from the RoboChart models) into FDR, which yields a counter-example that is used for testing. Yueng et al. [229] detail a process to contribute to the design of simulation experiments by analysing variations of timing parameters in CSP. They shown that the simulation experiment can yield different results in performance and in behavioural terms. Sumida et al. [205] demonstrate a case study in which they model a LEGO robot (EV3) and verify it in FDR for deadlock and livelock.

Several other studies employ different strategies to achieve different goals. Gainer et al. [77] synthesise formal models from control rules of robots in a DSL and input those models into NuSMV for model checking. Doan et al. [64] employ MAUDE to formally verify gathering of robots in a RING network. Santos et al. [193] work on generating unit testing for ROS components and property-based testing for ROS using the Hypothesis tool [155]. Bresolin et al. [39, 40] apply reachability analysis of Hybrid Automata in ARIADNE [26] to analyse the dynamics of surgery robots.

Informal In this category, there is a clear strength in generic simulation tools and architectures, followed by a strength in simulating road vehicles with domain-specific kinematic models. Some interventions focus on test-case generation and prioritisation as well as runtime monitoring both for generic RAS and for road vehicles. There is a clear weakness in domain-specific intervention for aerial vehicles, where there are only simulation tools for individual and connected vehicles reported in the literature and mobile robots (excluding road vehicles), where no intervention is included in our review. The simulation tools in various domains are often based on a combination of ROS [178] and Gazebo [116], Unity [211] and/or USARSim [45]. We refer to Section 5.3 for further explanation of these tools.

Road A majority of papers in this category introduce a simulation tool [1, 51, 54, 70, 130, 156, 207, 233] combining vehicle kinematics with other aspects of vehicle modelling such as communications [51], vision-based algorithm [130] and fuel consumption [70]. There are two interventions that use search-based testing and [25, 30]. Surrogate modelling, where a higher-level model is used to steer the search, is used in both approaches. Another approach uses past data to identify challenging situations and embed them into test-cases (using an XML structure) [201].

There are also a number of process interventions describing a process for safety assurance [81, 237] and testing Human-Machine Interfaces [163]. Some papers do use a well-defined syntax or a mathematical notation, but are classified as informal; in our classification, if a model does not have a rigorous formal syntax, semantics and reasoning method it is classified as informal. These include using XML as a formal model [201], mathematical descriptions of vehicle kinematics (see simulation tools above), and probabilistic descriptions for risks [237].

Aerial All interventions reported here concern simulation tools for modelling dynamics and control [42, 140, 162] and communication of aerial vehicles [67]. It is remarkable that these simulation tools rely on entirely different context tools, which will be analysed further in RQ3.

Generic In this category, the majority of interventions again propose simulation tools [118, 149, 185, 203] or a simulation architecture [225]. (Note that there are two simulation tools that address marine robots [149, 213], but since we did not have a separate class for such robots, we classified it here.) There are, however, a couple of interventions concerning test-case prioritisation [128] and automated unit-test execution [34], and run-time monitoring [103]. The run-time monitoring environment [103] provides an integration with ROS implementations.

Effectiveness Here we summarise the notions of effectiveness used in different domains; these notions comprise two sub-categories: the effectiveness of the RAS under test, which is the oracle or the property against which the RAS is tested and the effectiveness of the testing techniques, which provides an evaluation of the techniques, rather than the system under test:

Road Concerning the effectiveness measures for road vehicles, collision analysis is the most prominent metric, including analysis on the number of collisions [11, 16, 49, 54, 114, 131, 228], probability of collision [19, 129], and severity [210]. Furthermore, a few studies focus on the analysis of deviation from the intended path in terms of spatial and rotational deviation [30, 32, 50, 78]. As for measures for test adequacy, probability of faulty and rare events [156, 167], and the number of tests generated [209] have been studied; however, these latter measures are not domain-specific and similar measures have been used in other sub domains.

Aerial Only four studies have been included in this category. As measures of SUT efficiency, Desai et al [63] measure obstacle avoidance and plan execution. Similar to road vehicles, studies on probability of completing a task for aerial vehicles are found in the literature [18, 235]. Zhao et al. [235] takes a further step by analysing expected mission time and expected number of battery recharges during a mission. Conversely, as a test adequacy metric, [199] measures the accuracy of its simulation.

Mobile With respect to mobile robots, only a handful of papers collective metrics related to effectiveness. As in other sub-domains, collision avoidance [138] and probability of satisfying requirements [37] can also be found here. Brambilla et al. [37] also measures the improvement in the behaviour of a robot by analysing the number of objects retrieved. Arai et al. [12] also consider a similar measure of improvement in the behaviour but in terms of device utilisation. The only metric regarding testing adequacy is number of failures detected studied by Proetzsch et al. [177].

Generic As a unique measure of effectiveness, Ruijten [189] detects anthropomorphism in its subjects, which is a measure of human likeliness in robots. Several studies focus on analysing the probability of completing a mission successfully [15, 142, 159, 236]. Studies on safety [7, 164, 217] are also commonly found in this category. For instance, Viventini et al. [217] focus their efforts on analysis of hazards, such as the number of hazards identified, number of types of hazard and number of risk reduction measures taken.

Efficiency Similar to effectiveness, measures of efficiency can pertain to the system under test (measuring resource usage as a property to be checked or as an oracle for pass and fail) and the testing techniques (to measure the resources used in testing):

Road Mullins et al. use [160] precision, convergence and resolution for efficiency in testing. In a number of studies on verification of autonomous systems [71, 72, 111] the size of the state space as well as the total memory footprint [71] in their evaluation. is measured to evaluate efficiency. Sun et al. [206], in the verification of finite state models, use abstraction and verification time to estimate the efficiency in their work. Verification time is used in a number of studies [16, 71, 72, 92, 111, 148, 166] to measure efficiency. Gladisch et al. in [88] use simulation time to measure efficiency. Similarly Bi et al. use simulation time [32] for measuring the efficiency of the their work. Fayaz et al. [70] measure test duration in their evaluation. To evaluate the efficiency of the work [173], measure CPU usage and network bandwidth. Bode et al. [35] measure the cost (euros) of application of their approach as a notion of efficiency. Li et al. in [132] measure computational time in testing and comparing various autonomous vehicle decision and control systems.

Aerial Sirigineedi et al. [202] use verification time and number of states in Kripke structures, as system model, for measuring efficiency in their work. Urso et al. [67], use simulation time once using Gazibo as notion of efficiency. Zhao et al. [235], in their work that is using prism model checker, consider the expected measure time as a notion of efficiency.

Mobile Andrews et al. [10] consider the number of states and transitions as a measure of efficiency for the testing method. Arai et al. [12] presents results on the response time, throughput and device utilisation as measures of efficiency for the system under test.

Generic Verification time is commonly used in a number of studies [77, 137, 142, 172, 183, 190], to measure efficiency. The number of states is another common notion measured in evaluations [15, 77, 123, 136, 183, 229]. Furthermore, testing and simulation time is used to measure efficiency in a set of studies [83, 118, 193]. Althoff et al. in [6] measure the reduced idle time in self-verifying robots. Munawar et al. [161] measure latency and number of steps in simulation of surgical robots. Search time and optimisation time is used by Collet et al. [53] in evaluation of testing single arm robots. Gerstenberg et al in [83] measure task completion time in simulation. Vicentini et al. in [217] present the total time for risk reduction and hazard identification in their evaluation of formal verification techniques. Probability of task completion, number of executed instructions and time for completing the task are other measures used in evaluation of model checking autonomous systems [15]. Muhammad et al. [159] measure time and probability of task completion in probabilistic model checking of cooperative robot interaction.

Coverage Compared to the measures of efficiency and effectiveness, coverage measures are not as widely adopted.

Road Variations of structural coverage can be found in studies within this sub-domain. Neves et al. [58] developed a tool that conducts post-analysis (based on meta-models) on outputs collected from field testing of autonomous vehicles. Their tool aims to expand test coverage by exploring functionalities in the meta-model that were not covered during the field testing. Majzif et al. [144] devise a process that guarantees coverage of safety standards. They abstract results from component testing and make use of meta-models and situation graphs to compute a system-wide degree of test coverage and derive new scenarios to cover unexplored situations. Tatar [210] presents a method (implemented in TestWeaver [107]) for testing and validation of ADAS systems. The tool generates scenarios to cover relevant system states and feeds back previous executions to guide the next round of testing.

Aerial In the only paper in this category, Bicevskis, Gaujen and Kalnins [33] developed a new methods for testing and validation of autonomous processes collaboration. They build a collaboration model using an extended finite state machine and employ symbolic execution and feasibility tree analysis to check that all relevant states can be reached in the model. They have evaluated their strategy using a UAV case study.

Generic Tian et al. [212] propose DeepTest, a tool for testing Deep Neural Networks in autonomous vehicles. It generates tests that explore different parts of the DNN logic with the goal of maximising neuron coverage. Araiza-Illan, Pipe and Eder [14] propose a methodology for generating test cases that achieve high code coverage in human-robot collaborative tasks. They developed a testbench for ROS that makes use of belief-desire-intention (BDI) agents to generate valid and human-like tests. Structural coverage of Petri nets models have been utilised by Sagglietti in different contexts, such as in the generation of test cases for autonomous agents [133], and the verification of reconfiguration behaviour of autonomous agents [191, 192].

Open-source

Road There are a handful of open-source tools for formal verification [16], testing [54], and simulation [4, 80, 212] of autonomous vehicles. Testing and simulation seem to be gaining some strength with respect to open-source tools and we see tools for scenario generation, testing, and simulation of autonomous vehicles within various traffic scenarios. For connected vehicles, VerifCar [16] is a framework based on timed automata and is dedicated to modelling and verifying and validating connected autonomous vehicles policies.

Garzón and Spalanzani [80] present a tool that combines 3D simulation (for ego-vehicle control) with a traffic simulator (which controls the behaviour of other vehicles). The goal is to test the ego-vehicle in realistic high traffic situations. The FLPolyVF tool [4] connects functional verification, sensor verification, diagnostics, and industry/regulatory communication of autonomous vehicles whilst checking the effects of using different scenario abstraction levels. The MoVE tool [54] provides the possibility of modelling pedestrian behaviour. The framework focuses on testing autonomous system algorithms, vehicles, and their interactions with real and simulated vehicles and pedestrians.

Gambi, Mueller and Fraser present the AsFault prototype tool [78]. The tool combines procedural content generation and search-based testing in order to automatically create challenging virtual scenarios for testing self-driving car software. Tian et al. [212] propose DeepTest, a tool for testing Deep Neural Networks in autonomous vehicles. It generates tests that explore different parts of the DNN logic with the goal of maximising neuron co

Aerial All new tools reported here concern simulation tools for modelling dynamics and control [18, 140] and communication of aerial vehicles [67].

Lugo-Cárdenas, Luzano and Flores [140] introduce a 3D simulation tool for UAVs whose focus is on assisting the development of flight controllers. Analogously, D'Urso, Santoro and Santoro [67] also present a simulator for UAVs, called GzUAVChannel. The framework combines Gazebo, Autopilot, and NS-3 network simulator to provide a 3D visualisation engine, a physics simulator, a flight control stack and a network simulator to handle communications among unmanned aerial vehicles. On the stochastic side of software verification, Bao et al. [18] present a prototype tool for parametric statistical model checking that can cope with complex parametric Markov chains where state-of-the-art tools (such as PRISM) have timed out. They provide evidence of their tool efficiency by conducting an industrial case study.

Generic Several open-source tools have been proposed in this category, with a majority of them being simulators. The only exceptions are a formal verification tool [176] for human-robot interactions and two runtime verification tools [63, 103].

Rohmer, Singh and Freese introduce VREP [185] a popular robotics physics simulator that is now known as CoppeliaSIM. The tool uses a kinematics engine and several physics libraries to provide rigid body simulations (including meshes, joints and multiple types of sensors). Brambilla et al. have developed ARGOS [37], which is a multi-physics robot simulator that can simulate large-scale swarms and can be customised via plug-ins. In the Matlab environment, the FROST tool [101] is an open-source Matlab toolkit for modeling, trajectory optimisation and simulation of robots, with a particular focus in dynamic locomotion. Munawar and Fischer [161] present the Asynchronous Framework, which incorporates real-time dynamic simulation and interfaces with learning agents to train and potentially allow for the execution of shared sub-tasks.

For underwater robots, three new tools have been introduced: Manhaes and Rauschenbach present UUV simulator [149] which is an extension of Gazebo accomodating the domain-specific aspects of underwater vehicles. Cieslak et al. introduces Stonefish, a geometry-based simulator [52] that can be integrated with

ROS. Lastly, the MARS [213] tool provides simulation environments for marine swarm robots. As for tools in the human-robot interaction (HRI) domain, RoVer [176], provides visual authoring of HRI, formalisation of properties in temporal logic, and verification that the interactions abide by set of social norms and task expectations.

Huang et al. present ROSRV [103], which is a runtime verification framework that that can be used with ROS. Desai et al. [63] presents a runtime verification framework based on Signal Temporal Logic [145], where an online monitor checks robustness on partial trajectories from low-level controllers. In the context of surgical robots.

Public The tools included in this category are diverse; we report below on a test-scenario generation tool for road vehicles [160], two simulation tools for aerial [199] and generic [234] robots and a formal verification tool for generic robots (applied to a UAV case study) [46–48, 157].

Road Mullins et al. [160] developed a tool (RAPT - Range Adversarial Planning Tool) for generating test scenarios. The tools employs an adaptive search method that generates new scenarios based on the performance and results of the previous one. A clustering algorithm ranks the scenarios based on the performance type and how close they are to the boundaries of each cluster. The boundaries are based on notions of efficiency, diversity and scaling.

Aerial Shah et al. [199] introduce the AirSim simulator that generates training data for building machine learning models used in autonomous air crafts. It offers physical and visual simulation, including models of physics engine, vehicle, environment, and sensors.

Generic There are two tools included in this category: a formal specification and verification tool [46–48, 157] and a simulation tool [234]. Cavalcanti et al. [46–48, 157] introduce RoboTool, supporting graphical modelling, validation, and model checking (via FDR [85]) of robotic models written in RoboChart [157] and RoboSim [48]. Zhang et al. [234] introduce CyberEarth, a framework for program-driven simulation, visualisation and monitoring of robots. The tool integrates modules from several other open-source tool such as ROS [178] and OpenSceneGraph (OSG [43]).

Proprietary

Road The three tools included in this category comprise two tools for formal analysis [99, 166] and a simulation tool [233], each of which are explained further below.

Heitmeyer and Leonard [99] introduce two tools integrated into the FORMAL framework; the tools synthesise and validate formal models. The first tool syntheses a formal Software Cost Reduction (SCR) requirements model from scenarios and the second tool combines the existing SCR simulator [97] with eBotworks 3D simulator to allow for simulation of continuous components.

O'kelly introduces APEX [166], which is a tool for formally verifying the trajectory planning and tracking stacks of ADAS in vehicles. Zhang et al. present RoadView [233] a photo-realistic simulator that tests performance of autonomous vehicles and evaluate their self-driving tasks.

Generic The three tools included in this category are diverse and range from simulation [216] to formal verification [75] to model-based testing [53].

Verma et al. [216] present a Flight Software simulator that is used to simulate MARS Rover missions. The simulator assists in predicting the behaviour of semi-autonomous systems by providing the capability for human operators to check if their intent is correctly captured by the robot prior to execution in different scenarios and environments. Foughali et al. [75] implement an automatic translation from GenoM [146], a robotics model-based software engineering framework, to the formal specification language Fiacre [28], which can be fed into TINA [29] for formal verification. Collet et al./ [53] introduce RobTest, a tool for generating collision-free trajectories for stress testing of single-arm robots. It employs constraint programming techniques to solve continuous domain constraints in its trajectory generation process.

Small A considerable number of studies, among included in this survey, consider small case studies in their experiments. Here, we review the most prominent ones.

Road A few papers employ case studies with a focus on collision avoidance for road vehicles [4, 49, 71, 129]. They all focus on detecting imminent collision using built-in sensors. Similarly, Gaurehof, Munk and Burton [81] conduct a case study where a machine learning function detects pedestrian using a video analysis.

Several case studies concentrate on driving scenarios and manoeuvres, such as lane changing [19, 166], lane and path following [11, 233], merging [80], roundabouts [228], traffic scenarios [132, 156], parking [50], and overall cruise control [72, 111, 131, 148, 218]. A focus on the actual decision making and path planning can be seen in a handful of the case studies [50, 51, 227] as well. Hardware in the loop simulations [96] and human-machine interaction for driving assistance [232] can also be seen among the included case studies.

Aerial The small scale case studies in this domain only present theoretical or very limited models of UAVs, such as a surveillance drone [63] or a model for UAV launch [94]. Moreover, Brunel et al. [41] conduct a safety case analysis whilst Bu et al. [42] explore simulation and realistic testing of vision-based object tracking for UAVs. Aerial drones in co-operative scenarios are also the subject of two case studies [104, 202].

Mobile Only two small scale case studies were included here: Lu et al. [138] make use of PRISM model checker to investigate three collision avoidance algorithms in a unmanned surface vehicle model with a dynamic intruder and Arai and Schlingloff [12] employ a model checking technique on a transport robot model.

Generic Several case studies can be found within this category, with the vast majority being small models that are applied to demonstrate the respective intervention. Here, we briefly present and discuss some of them.

Walter, Täubig, and Lüth [219] provide an algorithm that increases safety through formal verification using the theorem prover Isabelle; the case study is a small robot. Nguyen et al. [164] provide a multi-step process to verify correctness of autonomous agents and apply it to a cleaner robot. Fu and Drabo [231] model a humanoid robot in an extension of Petri nets (called Predicate Transition Reconfigurable Nets - PrTR Nets) and formally verify it. Lill et al. [133] also make use of Petri Nets, however, they develop models of cooperative forklifts and simulate scenarios where the robots decide which one has the priority when passing through narrow pathways.

Farulla and Lamprecht [69] conduct a case study on human-robot interaction processes that have been modelled in DIME, and show how they can be verified with the GEAR model checker. Zhang et al. [234] have built a virtual simulation platform, CyberEarth, for robotics and cyber-physical systems. A visual coverage task for UAVs is also introduced to demonstrate the platform. Dennis and Fisher [62] apply an agent verification approach to verify the correctness of an agents ethical decision-making. Doan, Bonnet and Ogata [64] specify and formally verify, using the model checker Maude, a robotic gathering model.

Industrial

Road The industrial case studies involving road vehicles included in our survey typically involve verifying specific components of such systems.

In the context of advanced driver assistance systems (ADAS), Abdessalem et al. [25] generate test cases for a such system that can visually detect pedestrians. Zhou et al. [238] introduces a framework for virtual testing of advanced driver assistant systems that uses real world measurements. Kluck et al. [114] consider virtual driving scenarios for testing automated emergency breaks. AbdElSalam et al. [1] use Hardware Emulation-in-the-loop to verify Electronic Control Units (ECUs) for ADAS systems.

Fayazi, Vahidi, and Luckow [70] implement a vehicle-in-the-loop verification environment and conduct field testing in the International Transportation Innovation Center (ITIC).

Gladisch et al. [88] select case studies that use industrial automated driving (adaptive cruise control, lane keeping, and steering control scenarios) to evaluate their search-based testing strategy. Abdessalem et al. [2] generate test cases for the *SafeDrive* system, which contains the following four self-driving features: autonomous cruise control, traffic sign recognition, pedestrian protection, and automated emergency braking.

Aerial Shah et al. [199] build a model of quadrotor with pixhawk controller in their newly developed simulator, AirSim, that includes a physics engine and and supports real-time hardware-in-the-loop. Rooker et al. [186] demonstrate their validation framework for autonomous systems in a farming context with simulations and field testing. They employ both UAVs and ground mobile systems. Bhattacharyya et al. [31] apply formal verification methods to an autonomous CoPilot agent.

Mobile The only study in this category is the study conducted by Rooker et al. [186], which is also mentioned above. In summary, they demonstrate their validation framework for autonomous systems in a farming context with simulations and field testing.

Generic Jacoff et al. [106] conduct field testing for the performance evaluation of robots used in disaster scenarios. Verma et al. [216] present a Flight Software simulator that is used to simulate MARS Rover missions. They demonstrate their approach with a case study. Satoh [195] conducts a case study using a physical transport robot to demonstrate their framework that can emulate the robot's physical mobility.

Manhaes and Rauschenbach [149] model the Sperre SF 30k ROV underwater robot (RexROV) in the demonstration of the simulator for unmanned underwater vehicles. Uriagereka et al. [215] conduct simulation-assisted fault injection to assess safety and reliability of robotic systems. The feasibility of their method is demonstrated by applying it to the design of a real-time cartesian impedance control system. Gainer et al. [77] conduct a case study in the context of verification of human-robot interaction using the Care-O-Bot robotic assistant.

Benchmark

Road Many different case studies have been included in this category. We briefly discuss the most distinguished ones. For instance, Neves et al. [58] developed a tool that conducts post-analysis (based on meta-models) on outputs collected from field testing of autonomous vehicles. Five field testings involving a program to control the navigation of an autonomous vehicle, CaRINA I, were performed. Zofka et al. [239] present the framework Sleepwalker for verifying and validating autonomous vehicles and demonstrate the benefits of their framework using different instances stimulating an autonomous vehicle.

Mullins et al. [160] have developed a tool (RAPT - Range Adversarial Planning Tool) for generating test scenarios to be employed on the System Under Test. Their tool is applied to realistic underwater missions. Heitmeyer et al. [99] synthesise software cost-reduction models of multiple autonomous systems to be used in a simulator integrated with the eBotworks simulation tool. Gruber and Althoff [92] present a reachability analysis tool (Spot) that finds counter-example to property violations. Their tool is evaluated using the CommonRoad benchmark PM1:MW1:DEU_Muc-3_1_T-1.

Pereira et al. [173] employ several small case studies in their attempt to couple two simulators, namely SUMO and USAR-Sim. Pasareanu, Gopinath, and Yu [171] present a compositional approach for the verification of autonomous systems and apply the technique on a neural network implementation of a controller for a collision avoidance system on the ACAS Xu unmanned aircraft. Bi et al. [32] present a deep Learning-based framework for traffic simulation and execute several scenarios of intersections with and without pedestrians.

Aerial Not many studies have applied their intervention to benchmarks of aerial systems. Bicevskis, Gaujens, and Kalnins [33] develop models for the testing of UAV and UGV collaboration in the Simulink environment. Mutter et al. [162] also explore the simulation of UAV models in Simulink and discuss the results when combining the platform and environment models. D'Urso, Santoro, and Santoro [67] simulate leader follower UAV scenarios in their framework. Their goal is to combine four simulation environments: a 3D visualisation engine, a physics simulator, a flight control stack and a network simulator.

Wang and Cheng [220] present a hardware-in-the-loop simulator for drones that can generate synthetic images from the scene as datasets, detect and verify objects with a trained neural network, and generate point cloud data for model validation. They simulate and conduct filed testing on a physical UAV. Zhao et al. [235] model an unmanned aerial vehicle (UAV) inspection mission on a wind farm and, via probabilistic model checking in PRISM, show how the battery features may affect verification results.

Mobile Two studies fit this category. Proetzsch et al. [177] use a designed DSL (graph-based models) to describe the system behaviour of the autonomous off-road robot RAVON. The model is used is used as test model for generating test cases. Brambilla et al. [37] model a probabilistic swarm that is checked in PRISM to evaluate their property-driven design method.

Generic Several studies have applied their intervention to benchmark case studies of generic/immobile robots. We briefly discuss the most distinguished ones.

Tosum et al. [214] present a design framework that facilitates the rapid creation of configurations and behaviours for modular robots. They've demonstrate their framework on the SMORES robot. Halder et al. [95] use the physical robot Kobuki as a case study, over which properties are automatically verified using the UPPAAL model checker. The focus of their approach is to model and verify ROS systems using real time properties. Laval, Fabrese, and Bouraqadi [128] introduce a methodology to support the definition of repeatable, reusable, semi-automated tests and apply it to a 2-wheels differential drive robot.

Bohlmann, Klinger, and Szczerbicka [36] automatically generate a model of a parallel delta robot on-the-fly. Their method for model generation is based on machine learning and symbiotic simulation techniques. Mariager et al. [151] design and field-test a robot that interacts with adolescents with cerebral palsy. Althoff et al. [6] propose a framework (IMPROV) for self-programming and self-verification for robots, which is demonstrated on a physical robotic arm. Wingand et al. [225] have developed CoSiMA, which is an architecture for simulation, execution and analysis of robotics system. They conduct experiments on the humanoid robot COMAN.

In Table 8, we map the identified subdomains to the different aspects of our research questions as follows:

RQ1 Across all subdomains a majority of models have been formal and quantitative and substantial gaps can be detected (most notably in the aerial vehicles and mobile robots subdomain) regarding using qualitative and informal models for testing.

RQ2 Across all studied subdomains, there is a clear gap in using precise notions of effectiveness, efficiency, and coverage. Among these, some generic notions of effectiveness and efficiency (such as testing time and state space size) and the notion of coverage (such as node and transition coverage) are the most-used measure for quantifying the effect. Common, more sophisticated measures of effectiveness, efficiency, and adequacy such as Average Percentage of Faults Detected (APFD) [187] do not seem to have been adopted in or extended to the domain of RAS. We do see some recent trend towards domain-specific notions of effectiveness and coverage [2, 25, 33, 35, 113, 144, 212]; almost all of these notions have been applied to the autonomous vehicles domain, but most of them can be adapted to be applicable to other domains as well.

RQ3 There is a considerable gap concerning tool support for testing RAS. There are very few open source tools, mostly in the autonomous vehicles [4, 16, 54, 78, 80, 92, 212] and aerial vehicles [18, 67, 140] subdomains. No open-source tools support the domain-specific aspects of mobile robotic system. The same pattern with a more severe gap is present for proprietary tools. Very few public (but not open source) tools are developed or used in the reviewed literature.

RQ4 There is also a very severe gap across all subdomains in using industrial case studies for evaluating RAS testing interventions. The most notable exceptions are a handful of case studies, mostly in the autonomous vehicles [1, 2, 25, 70, 88, 114, 196, 198, 209, 238] and the aerial vehicles [77, 106, 149, 154, 195, 215, 216] sub-domains, performed in an industrial context. Many interventions used small case studies, mostly without any specific application subdomain (e.g., using generic models of mobile robots); in these cases, the models did not contain enough details to be part of a general benchmark. There have also been some evaluations performed on small case studies based on drones and UAVs.

Analysis for Researchers. **Gaps:** In our analysis of the studied subdomains, there is a clear gap in treating marine and sub-marine RAS. Also there is a relative weakness in treating aerial vehicles and mobile robots. Moreover, there is a relative weakness across subdomains concerning the treatment of informal and qualitative models. Developing a common set of notions of effectiveness and efficiency to compare different intervention is a worthwhile research challenge and there is a gap in the literature in tailoring them for specific domains. The same observation holds for the notion of test adequacy. Tooling, particularly tailored for specific subdomains, is a general weakness for all interventions. Moreover, applying the interventions in industrial context is an outstanding challenge.

Strengths: The road vehicle subdomain has a considerable strengths across all research questions. Also there are far more interventions that have been developed for generic RAS systems without treating the specific concerns of sub-domains. Formal and quantitative models are by far the strongest interventions both in terms of the number of techniques studies and the evaluations performed, even in industrial domains.

Analysis for Practitioners.

Gaps: Since most of the proposed interventions have not been evaluated in industrial context, evaluating their applicability, including studying factors such as the learning curve and training, remains a substantial gap.

Strengths: Due to the available strength in formal and quantitative models, developing such models provides a starting point to benefit from the developed and studied interventions. There is certainly more maturity in the area of road vehicles to benefit from in practice, but we can envisage that by tuning the domain-specific aspects, other sub-domains may also benefit from these strengths.

6.1.2 Cooperation and Connectivity. Verification methods are pivotal for the wide-spread deployment and public acceptance of autonomous systems. The need for such methods is intensified in the functions enabled by network services, due to the close interaction among the communication protocols, control software (e.g., for cooperation rules), and system dynamics. Existing (manual) analysis techniques typically do not scale to the huge design-space and input-space of these functions and, hence, in this work, we survey automated verification techniques found in the literature.

Table 9 provides an overview of the interventions used to test cooperation and connectivity in RAS. The interventions can be broadly categorised into swarm RAS, where an emerging behaviour is to be observed through cooperation of a large number of RAS, versus cooperative RAS where few RAS units engage in a well-defined interaction (possibly with their environment) to achieve a goal.

In general, this turns out to be an understudied area of testing RAS and little focus has been put in testing cooperative and connected scenarios in the literature. For the very few interventions reported in the literature, there is scarcely any evidence of efficiency or effectiveness available. The handful of reported evaluations are only performed on small-scale case studied and are not accompanied by open source tools. In our analysis, we focused on cooperation among robots; however, only in 2019, we encountered some papers that study cooperation from a human-robot interaction viewpoint [6, 151, 189].

Table 9. Testing Cooperation and Connectivity in RAS

		Swarm	**Cooperative**
RQ1	**Qualitative**	[56, 120–122]	[64, 133, 192]
	Quantitative	[7, 37, 47, 136]	[16, 33, 104, 108, 159, 191]
	Formal	[7, 37, 47, 56, 120–122, 136]	[16, 33, 64, 104, 108, 133, 159, 191, 192]
	Informal		[67]
RQ2	**Effectiveness**	[7, 37]	[16, 159]
	Efficiency	[136]	[16, 67, 159]
	Coverage		[33, 133, 191, 192]
RQ3	**Open-source**	[37, 136]	[16, 67]
	Public	[48]	
	Private		
RQ4	**Small**	[7, 47, 56, 120, 121, 136]	[64, 104, 108, 133, 159, 175, 191, 192]
	Industrial		[186]
	Benchmarks	[37]	[33, 67]

Overall there is very little about the stochastic details of communication protocols. The studies in this category mostly focus on verification of movement of robots (i.e., gathering and merging).

Qualitative

Swarm With respect to swarms, a number of theoretical studies [120–122] focus on scaling up the parametrised model checking problem to large swarm sizes. They employ various types of epistemic extensions of CTL as property specification languages. Their models include case studies on clustering of swarms, which synchronise to gather in a certain area. Cybulski et al.'s contribution [56] to the field is a simulation framework for the behaviour of UAV swarms. The framework also allows for performing simulations with a user-defined map of the environment.

Cooperative Regarding the use of qualitative models of cooperative systems, our search only resulted in three studies, two of which employ variations of Petri Nets as their models. Lill et al. [133] make use of Petri Nets to develop models of cooperative forklifts. The forklifts communicate to decide which has the priority when passing through narrow pathways. Sagglieti et al. [192] employ Coloured Petri Nets and classify cooperation in three distinct levels: perception-based, reasoning-based and action-based cooperation. In order to demonstrate their strategy, they model platooning-like scenarios where different robots follow each other. The third study, by Doan et al. [64], is a more theoretical study of parametric model checking employs model checking for multiple small robots gathering in circular configurations using a ring-based topology. Their study focuses on model checking the underlying distributed system and the properties written in LTL.

Quantitative

Swarm Three out of four papers in this category deal with probabilistic behaviour: Lomuscio et al. [136] perform model checking on probabilistic LTL, whilst Anmin et al. [7] and Brambilla et al. [37] both describe properties in PCTL. Cavalcanti et al. [47], on the other hand, models timed dynamics in CSP.

Cooperative The studies that have been classified in this category employ variations of temporal logic as their properties, such as LTL [104, 108, 159] and CTL [16]. As an exception to that list, Bicevskis et al. [33] provide a simulation environment in Simulink.

Formal

Swarm Kouvaros et al. [120–122] provide several theoretical studies in the field of formal verification and model checking of autonomous systems. They have demonstrated the applicability of their strategy on a case study of gathering of UAVs swarms. With respect to probabilistic systems, three studies have been found: Lomuscio et al. [136] offer a strategy for parameterised model checking of probabilistic LTLs. Furthermore, Brambilla et al. [37] provide a property-driven design method for probabilistic swarms that is checked using Prism. Lastly, Amin et al. [7] verify probabilistic behaviour expressed via PCTL properties using UPPAAL. They assert deadlock freedom, safety liveness checks, and reachability computations.

Cooperative The majority of studies employing formal methods in analysing RAS used model checking [16, 33, 64, 104, 108, 159, 159]; after model checking the most frequently used technique is model-based testing [133, 191, 192]. Arcile et al. [16] and Kamali et al. [108] investigate car platooning manoeuvres. In the former, the vehicles are modelled as timed automata and UPPAAL is used as a model-checking tool. In the latter, joining and exiting operations are modelled in Belief Desire Intention models and model checked using AJPF. They focus on abstracting a formal (untimed) model from an agent (timed) model and checking the correspondence between the agent model and the code. With respect to probabilistic model-checking, Muhammad et al. [159] model robots that synchronise to position themselves in an attempt to guarantee coverage of a certain area. The models are Markov Decision Processes that are checked using PRISM. Humprey et al. [104] make use of the model checker Spin to investigate cooperation between UAVs and sensors and collaboration among sensors as well.

Informal

Cooperative The only study in this category [67] provides a simulation environment that integrates existing solutions for simulation of multi-UAV applications such as Gazebo (for robotic simulation), ArduPilopt (for UAV control algorithms) and NS3 (for network simulation). Their case study is a model of a leader-follower application for large convoys of UAVs.

Effectiveness As noted before, there are very few measures of effectiveness used for evaluating the system or the testing technique:

Swarm Amin et al. [7], use very generic notions of effectiveness for the system under test, namely, deadlock freedom, safety, liveness, and reachability. Brambilla et al. [37] go beyond that and in addition to some domain-agnostic properties, such as probability of satisfying the requirements (safety), they measure aggregation time of the swarm, and improvement of the behaviour (in terms of objects retrieved).

Cooperative Muhammad et al. [159] measure the probability of task completion, and human interaction to measure efficiency in wireless sensor networks. Arcile et al. [16] measure the number of collisions in its vehicle verification approach.

Efficiency

Swarm Lomuscio et al. [136] count the number of sates and transitions as a measure of efficiency of their formal verification methodology.

Cooperative Muhammad et al. [159] also measure the time of task completion, and human interaction as a measure of efficiency for the system under test. Arcile et al. [16] measure the travel time for the system under test, and the verification time as efficiency metrics for the respective testing technique. D'Urso et al. [67] measure simulation time as a measure of efficiency of their integrated simulator.

Coverage

Cooperative Saglietti, Winzinger, and Lill [192] consider state coverage in the analysis of their model-based reconfiguration testing strategy, whilst Bicevkis et al. [33] conduct testing of collaborative UAV and UGV and consider "the complete test set" as a measure test coverage. Lill and Saglietti [133] model Petri nets entities and address the maximisation of interaction coverage whilst minimising the amount of test cases.

Open-source

Swarm With respect to swarms, two open-source tools have been reported: the PSV-CA tool [136] can model check probabilistic PLTL properties for swarm systems and ARGOS [37] is a multi-physics robot simulator, which can simulate large-scale swarms and can be customised via plug-ins.

Cooperative Only two open-source tools have been found for cooperative robots: VerifCar [16] allows for fault injection in models for UPPAAL model checking. GzUAV [67], on the other hand, is a simulation tool for connected UAVs.

Public

Swarm The only public, non-open source tool that as been employed in the testing of swarms is the RoboTool by Cavalcanti et al. [48]; RoboTool supports modelling, and model checking (through the FDR tool [86]). The tool has been applied to a UAV swarm case study.

Small

Swarm Kouvaros and Lomuscio [120, 121] study parameterised verification of robot swarms against temporal-epistemic specifications and model a small, theoretical, robot swarm. Cavalcanti et al. [47] introduce RoboChart, which facilities for modelling and verifying collections of interacting robots. Cybulski [56] provide mathematical models of a UAV swarm that can be simulated in their proposed framework. Amin et al. [7] presents a formal verification approach using timed automata for the verification of path planning of robot swarms.

Cooperative Poncela and Aguayo-Torres [175] conduct a case study where they test underwater robots wireless communication. Lill et al. [133] make use of Petri Nets to develop models of cooperative forklifts and simulate scenarios where the robots decide which one has the priority when passing through narrow pathways. Humprey et al. [104] make use of the model checker Spin to investigate cooperation between UAVs and sensors and collaboration among sensors as well. Saglietti, Winzinger, and Lill [192] use coloured Petri Nets to model interacting autonomous agents and generate test cases for reconfiguration scenarios.

Benchmarks

Swarm The only reported benchmark for this category is by Brambilla et al. [37], where they investigate aggregation and foraging manoeuvres on large scale swarms of multiple sizes.

Cooperative D'Usrso et al. [67] evaluate their methodology on a number of test programs using different UAV sizes. They aim their evaluation at testing (i) the scalability of the solution and (ii) its performances by comparing the simulation time with respect to physical execution time.

Industrial

Cooperative The only reported industrial case study is by Rooker et al. [186]. They make use of a simulation tool in the smart farming domain. Land and air robots are modelled using real dynamics and cooperate to complete farming tasks.

RQ1 Regarding the models used for analysing cooperative scenarios in RAS, we notice that formal probabilistic models (based on variations of temporal logic [120–122], process algebra [48], and timed automata [7, 16]) are the most used types of models. Often these models are used for the purpose of model checking abstract models of cooperative scenarios. Informal models are used less frequently, and only as input to simulation tools [67]. Qualitative and informal models are used far less often in this context.

RQ2 Most notions of effectiveness and efficiency are the generic notions such as state-space size, verification time, and test coverage [33, 133] for the technique and deadlock freedom and probability of temporal logic formula satisfaction for the system under test. The only exceptions where domain-specific notions of efficiency and effectiveness were used concern aggregation time of the swarm [37] and effectiveness of human-robot interactions [159].

RQ3 There is clearly a lack of tools for testing cooperative and swarm scenarios in RAS. The only exceptions are public model checking tools [16, 37, 48, 136] and a simulation tool for connected UAVs [67].

RQ4 Very few studies have evaluated their interventions on industrial-scale case studies [186] and benchmarks [33, 37, 67].

Analysis for Researchers. **Gaps:** An analysis of the included studies reveals that in cooperative scenarios for RAS, the role of communication networks and protocols and their effect on functionality, safety, and reliability of the RAS system is severely understudied. Integrating the body of knowledge available in communications with the testing and verification of RAS is clearly an area for future research. The very few available studies do not provide domain-specific measures of efficiency and effectiveness that pertain to the cooperative aspects and the emerging cooperative behaviour. Moreover, there is a lack of sufficient evidence of strategies being applied to industrial-scale case studies and benchmarks.

Strengths: There is certainly a strength in abstract theories for parameterised model checking of swarms. Apart from that there is no other concentrated area of strength.

Analysis for Practitioners. **Gaps:** As noted above, we do not think we have reached sufficient maturity in the research results for cooperative and swarm robots to be able to apply them in practice. Even the existing techniques have not been applied to many industrial case studies yet and no stable tool-sets are available at the moment. Working with researchers to define meaningful notions of efficiency and effectiveness as well as providing benchmarks and industrial case studies could lead an impactful future research agenda.

Strengths: There are no practical areas of strength in testing cooperative and swarm RAS scenarios.

6.1.3 *Testing strategy.*

Table 10 provides an overview of the testing strategies used for RAS. By far the most widely used strategy is formal verification, followed by simulation and runtime monitoring, respectively. Model-based testing is the least researched strategy.

Qualitative

Table 10. Overview of the testing strategies used for RAS.

		Simulation	**Model Based Testing**	**Formal Verification**	**Runtime Monitoring**
RQ1	**Qualitative**	[56, 99, 193]	[10, 14, 94, 133, 147, 164, 192]	[8, 9, 17, 22–24, 31, 38, 41, 60, 62, 64, 69, 71, 72, 89, 120–123, 135, 137, 148, 168, 188, 198, 206, 217, 222–224, 229, 231]	
	Quantitative	[1, 37, 42, 48, 52, 54, 87, 101, 118, 127, 130–132, 140, 149, 156, 157, 162, 173, 185, 201, 207, 213, 216, 225, 233]	[25, 30, 53, 134, 177, 191]	[6, 12, 16, 18, 19, 33, 37, 39, 40, 46–48, 57, 63, 73–75, 77, 82, 87, 89, 95, 104, 108, 111, 136, 138, 142, 157, 159, 166, 169, 171, 172, 176, 183, 190, 202, 205, 219, 226–228, 235, 236]	[63, 88, 103, 141, 171, 203, 221]
RQ2	**Effectiveness**	[11, 32, 37, 49, 50, 54, 78, 127, 131, 156, 167, 199, 214, 215]	[25, 30, 160, 177]	[12, 16, 18, 19, 37, 57, 63, 71, 73, 138, 142, 159, 198, 217, 227, 228, 235, 236]	[63, 141]
	Efficiency	[35, 67, 83, 118, 132, 161, 173, 193]	[10, 32, 53, 160]	[6, 12, 16, 39, 40, 71, 72, 77, 92, 111, 123, 136, 137, 142, 148, 159, 166, 172, 183, 190, 202, 206, 217, 229, 235]	[88, 103]
	Coverage		[14, 94, 133, 191, 192]	[33]	[144]
RQ3	**Open-source**	[6, 37, 52, 54, 67, 78, 80, 101, 103, 118, 140, 149, 161, 185, 213, 225]		[4, 6, 16, 18, 63, 77, 92, 136, 137, 176]	[63, 103]
	Public	[48, 199, 234]	[160]	[48]	
	Private	[99, 196, 216, 233]	[53]	[75, 166]	
RQ4	**Small**	[11, 35, 42, 48–50, 56, 83, 87, 96, 130–132, 140, 156, 185, 213, 233, 234]	[10, 30, 53, 94, 133, 134, 147, 164, 191, 192]	[9, 12, 17–19, 22, 24, 38–41, 46–48, 60, 62–64, 69, 71, 72, 82, 87, 89, 104, 108, 111, 120, 121, 123, 136–138, 148, 159, 166, 169, 172, 188, 190, 202, 205, 219, 226–229, 231]	[63]
	Industrial	[1, 149, 186, 196, 199, 215, 216, 238]	[25]	[31, 77, 198]	[88]
	Benchmarks	[32, 36, 37, 54, 67, 78, 99, 101, 118, 127, 161, 162, 170, 173, 193, 203, 207, 214, 220, 225]	[14, 160, 177]	[6, 33, 37, 73–75, 92, 95, 171, 176, 183, 217, 222–224, 235, 236]	[103, 141, 171, 221]

Simulation Heitmeyer et al. [99] synthesise state-based formal models (Software Cost Reduction tabular models) from scenarios specified in Mode diagrams (extensions of Message Sequence Charts). The models are used in a simulator integrated with the eBotworks simulation tool. Cybulski et al. [56] developed a simulation tool for UAV based on class- and activities diagrams. Further, their framework allows for user-defined maps of the environment.

MBT Two search-based testing approaches are employed in this category: Lill and Saglietti [133] employ genetic algorithm to maximise coverage in the Petri nets models for generating test cases. Analogously, Nguyen et al. [164] provide a multi-step process to verify correctness of autonomous agents. They makes use of multi-objective evolutionary algorithms to cover stakeholder soft goals. Araiza et al. [14] and Andrews et al. [10] focus on human robot interaction. The former generate test cases from BDI models whilst the latter focuses on coverage to generate test cases from Petri nets. Another model-based testing tool that uses Petri nets is [192], which uses Coloured Petri Nets and structural coverage metrics to generate test cases for reconfiguration scenarios. Finally, Hagerman et al. [94] combines a behavioural model with attack and mitigation analyses to generate a security test suite for UAVs.

Formal Verification The vast majority of papers in this category perform formal verification based on properties specified in variations of temporal logic; since autonomous systems specifications typically involve aspects such as beliefs and intentions, several studies are dedicated to studying the theoretical boundaries (e.g., (un)decidability) of verifying epistemic extensions of temporal logics [120–123, 135, 137]; a notable tool used in this context is the MCMAS model checker [137], which is also evaluated on a small-scale benchmark against the general-purpose model checker NuSMV.

Many theoretical studies study the issue of abstraction for parameterised specifications, where parameters can be the number of autonomous agents [120–123, 135, 137] or the size and shape of the arena [8, 9]. Aminof et al. [9] investigate the decidability problem for parameterised grid sizes. They found that restricting the grid size, results the problem being solved in Pspace. On the same vein, Rubin et al. [8] establish a framework in which to model and automatically verify autonomous agents. The framework contains an algorithm tailored to solve a parameterised verification problem where they use the model graphs as parameter.

Coming up with temporal logic specifications is known to be difficult and require some level of formal training. To alleviate this, a few papers focus on writing LTL properties. Webster et al. [222, 223] model scenarios for a robot in the healthcare sector; they use Brahms as the language to describe human-robot interaction scenarios and the properties are written in LTL. Babiceanu et al. [17] combine LTL and Event-B to build models of trustworthiness for small unmanned aerial systems (sUAS).

Formalisation of and applying formal verification on different cognitive architectures have been the focus of many studies.

Bhattacharyya [31] formalise a rule-based representation of cognitive architecture using SOAR framework [125] UPPAAL and connect the verification agent to the simulation environment. They model an auto-pilot avionic system and analyse contingency situations during takeoff.

The Belief-Desire Intention framework is another natural cognitive architecture for specifying autonomous agents and it has been extensively used in the literature. Several studies make use of the MCAPL framework [61]. Dennis et al. [60, 62] verify ethical aspects of autonomous agents' interactions with people, by modelling their behaviour using the BDI models and capturing ethical priorities; these ethical models are subsequently model checked against LTL specifications using the MCAPL framework. Furthermore, Ferrandes et al. [71] model autonomous vehicle components and also use MCAPL to formally verify their BDI models. Lastly, Ferrando et al. [72] go further and provide an approach that combines formal verification and runtime monitoring by specifying trace behaviour in Prolog and connecting that with a JAVA implementation (using the JPL framework [3]) for runtime monitoring.

Sun et al. [206] study the effect of Neural networks components in the behaviour of autonomous systems; in particular, Rectified Nonlineary in Neural Networks and analyse it using Satisfiability Modulo Convex (SMC). In order to mitigate the verification effort, they perform a pre-processing and evaluate the effect of pre-processing on the verification time, measured in the number of neurons in the Neural Network.

With respect to the verification of safety on human-robot interaction, Vicentini et al. [217] provide safety assessments by formally verifying models written in TRIO temporal logic [84]. Their strategy aims to identify hazardous situations associated with non-negligible risks. Analogously, Farulla and Lampretc [69] focus on model checking security properties formulated in Computational Tree Logic (CTL).

Selvaraj et al. [198] evaluate the application of different formal techniques to verify control software for an autonomous vehicle. They investigate the application of Supervisory Control Theory, Model Checking and Deductive Verification and provide insights on how these different approaches can address different industrial challenges.

Quantitative Simulation Most interventions in this category focus strictly on introducing a simulation tool for a specific sub-domain, such as underwater robots [52, 149, 213], vehicles [54, 201, 233], robots [101, 127], and UAVs [140, 162].

On the other hand, a number of interventions combine a simulation approach with other testing aspects. Li et al. [132] employ a game-theoretical approach where vehicles have different levels of knowledge about other vehicles. For instance, a level 0 car has no knowledge about the other cars and a level k car has information about level k - 1 cars. Strangely, they show that, in some instances, lower level cars effect less constraint violations.

Szalay et al. [207] provide a scenario-in-the-loop simulation using SUMO and Unity engine. The simulate simplified platooning and valet parking scenarios and in both the simulation and in a real smart city (Salazone).

Verma et al. [216] present a Flight Software simulator that is used to simulate MARS Rover missions. The simulator assists in predicting the behaviour of semi-autonomous systems by providing the capability for human operators to check if their intent is correctly captured by the robot prior to execution in different scenarios and environments.

Two studies present supporting libraries: Koolen et al. [118] implement robotic simulation library in the Julia programming language. The library offers support for robot dynamics, visualisation, and control algorithms. Rohmer et al. [185] developed libraries to integrate VREP and other programming language (Lua, C++, Java, Python, Matlab and Ruby) with support for different types of 3D objects and modules for kinematics and dynamics.

Model Based Testing Multi-objective search is an increasingly popular technique for coping with complex robotics systems. Betts et al. [30] employ the monte-carlo search heuristic to verify the lateral distance between the outcome of surrogate-based models compared to a known ground truth in UAV applications. A similar approach is used by Nejati et al. [25] on pedestrian detection using vision-based system. They employ NSGA-II [59] using

[3] https://jpl7.org/

minimum distance and minimum time to collision as fitness functions and compare the performance of the heuristic with and without surrogates.

Sagliatelli and Meitner [191, 192], on the other hand, propose multiple notions of coverage to help generating test cases from Petri Nets models of autonomous, cooperative and reconfigurable robots. Furthermore, they employ statistical testing techniques, which intend to evaluate the degree of acceptance of the behaviour observed.

A framework for automated testing using metamorphic testing principles combined with model-based testing is employed by Lindvall et al. [134]. Test cases are generated from test models and multiple variations of scenarios that are programmatically generated based on metamorphic relations.

Runtime Monitoring In this category, verification checks at runtime have been typically coupled with model checking strategies, where properties are being checked during the system execution. For instance, Desai et al. [63] presents an STL-based framework where assumptions used during model checking hold at runtime, where the online monitor checks robustness on partial trajectories from low-level controllers.

In the context of obstacle avoidance, Luo et al. [141] employ JavaMOP to verify that a robot does not behaves against requirements written in FSM and PTLT languages. Temporal properties are also employed by Wang et al. [221], where the RoboticSpec specification language for robotic applications is translated into a framework for online monitoring that also uses PLTL properties.

Huang et al. [103] present ROSRV which is an online monitoring framework that runs on top of ROS. They make use of a public-subscribe communication architecture and intercept commands and messages passing through the communications channel. This way, they are able to verify safety and security requirements at runtime using a domain-specific language.

Open-source Simulation Manhaes and Rauschenbach present UUV simulator [149] which is an extension of Gazebo accommodating the domain-specific aspects of underwater vehicles. They assist with modelling of underwater hydrostatic and hydrodynamic effects, thrusters, sensors, and external disturbances and demonstrate their tool on a case using a modified model of the Sperre SF 30k ROV robot (RexROV).

As another tool for underwater robots, MARS [213] provides simulation environments for marine swarm robots that allows for hardware-in-the-loop simulation. The tool has a Java interface and has been applied to the MONSUN and HANSE autonomous underwater robots.

In the Matlab environment, the FROST tool [101] is an open-source Matlab toolkit for modeling, trajectory optimisation and simulation of robots, with a particular focus in dynamic locomotion. In the study, they model the ATLAS and DRC-HUBO as examples.

Munawar and Fischer [161] present the Asynchronous Framework, which incorporates real-time dynamic simulation and interfaces with learning agents to train and potentially allow for the execution of shared sub-tasks. Due to the asynchronous nature of the communication, they measure the number of packets against latency. Furthermore, they focus on surgical robots as part of their application domain and they employ the CHAI3D haptics framework. They connect their tools with ROS, which allows them to connect to learning libraries such as TensorFlow.

D’Urso, Santoro and Santoro [67] also present a simulator for multi-UAV applications, called GzUAVChannel. It works as a middleware that combines Gazebo, Autopilot, and NS-3 network simulator to provide a 3D visualisation engine, a physics simulator, a flight control stack and a network simulator to handle communications among unmanned aerial vehicles. They model a leader-follower example.

The MoVE tool [54] provides the possibility of modelling pedestrian behaviour. The framework focuses on testing autonomous system algorithms, vehicles, and their interactions with real and simulated vehicles and pedestrians. They conduct three case studies: traffic wave observation, medical evacuation and virtual vehicles avoiding real pedestrian.

Rohmer, Singh and Freese introduce VREP [185] a popular robotics physics simulator that is now known as CoppeliaSIM. The tool uses a kinematics engine and several physics libraries to provide rigid body simulations (including meshes, joints and multiple types of sensors).

Koolen et al. [118] implement robotic simulation library in the Julia programming language. The library offers support for robot dynamics, visualisation, and control algorithms.

Brambilla et al. have developed ARGOS [37], which is a multi-physics robot simulator that can simulate large-scale swarms and can be customised via plug-ins.

Cieslak et al. introduces Stonefish, a geometry-based simulator [52] that can be integrated with ROS. Lastly, the MARS [213] tool provides simulation environments for marine swarm robots.

Gambi, Mueller and Fraser present the AsFault prototype tool [78]. The tool combines procedural content generation and search-based testing in order to automatically create challenging virtual scenarios for testing self-driving car software.

Garzón and Spalanzani [80] present a tool that combines 3D simulation (for ego-vehicle control) with a traffic simulator (which controls the behaviour of other vehicles). The goal is to test the ego-vehicle in realistic high traffic situations.

Lugo-Cárdenas, Luzano and Flores [140] introduce a 3D simulation tool for UAVs whose focus is on assisting the development of flight controllers.

Formal Verification Parametric modelling of CAVs as a network of timed automata is used by Arcile et al. [16]. In this work, VerifCar tool is applied to asses the impact of communication delays on the decision algorithms of CAVs and to check the robustness and efficiency of such algorithms.

Gruber and Althoff [92] present a reachability analysis tool (Spot) that finds counter-example to property violations. It starts with a coarse model of the system dynamics but it can refine the abstraction levels for precision/scaling.

Desai et al. [63] present a runtime verification framework (DRONA) based on Signal Temporal Logic [], where an online monitor checks robustness on partial trajectories from low-level controllers.

RoVer [176], provides visual authoring of HRI, formalisation of properties in temporal logic, and verification (via model-checking with PRISM []) that the interactions abide by set of social norms and task expectations whose goal is to identify social norms violation.

Althoff introduces IMPROV [6], a tool that is used to formally verify human-robot interaction for modular robots.

Gainer et al. [77] provide a tool (CRutoN) for translation to formal models from control rules of robots in a DSL and input those models into NuSMV for model checking. Their main emphasis is the verification of human-robot interaction.

Bao et al. [18] present a prototype tool for parametric statistical model checking that can cope with complex parametric Markov chains where state-of-the-art tools (such as PRISM) have timed out. They provide evidence of their tool efficiency by conducting an industrial case study.

The FLPolyVF tool [4] connects functional verification, sensor verification, diagnostics, and industry/regulatory communication of autonomous vehicles whilst checking the effects of using different (matrix-based) scenario abstraction levels.

Lomuscio et al. have developed the MCMAS model checker [137]. They used a logic (ATL-k) alternating temporal logic (epistemic). One of the few tools that uses CTL. They demonstrate their strategy in a couple of small-scale examples, where they compare their strategy against alternatives (NuSMV and MCTK).

Runtime Monitoring Huang et al. present ROSRV [103], which is a runtime verification framework that that can be used with ROS.

Desai et al. [63] presents a runtime verification framework based on Signal Temporal Logic, where an online monitor checks robustness on partial trajectories from low-level controllers. In the context of surgical robots.

Public

Simulation Cavalcanti et al. [46–48, 157] introduce RoboTool, supporting graphical modelling, validation, and model checking (via FDR [85]) of robotic models written in RoboChart [157] and RoboSim [48].

Shah et al. [199] introduce the AirSim simulator that generates training data for building machine learning models used in autonomous air crafts. It offers physical and visual simulation, including models of physics engine, vehicle, environment, and sensors. Further, it connects to an API for planning and control [REF].

Zhang et al. [234] introduce CyberEarth, a framework for program-driven simulation, visualisation and monitoring of robots. The tool integrates modules from several other open-source tool such as ROS [178] and OpenSceneGraph (OSG [43]).

Model Based Testing Mullins et al. [160] developed a tool (RAPT - Range Adversarial Planning Tool) for generating test scenarios to be employed on the System Under Test. The tools employs an adaptive search method that generates challenging scenarios based on the performance and results of the previous ones. A clustering algorithm ranks the scenarios based on the performance type and how close they are to the boundaries of each cluster. The boundaries are based on notions of efficiency (precision and convergence), diversity (how many performance boundaries are being covered) and scaling.

Formal Verification The only tool in this category is RoboTool [48] which has also been described above in the Simulation category. It provides formal verification via translated CSP models fed into the FDR model checker [85].

Private

Simulation Heitmeyer and Leonard [99] introduce two tools integrated into the FORMAL framework; the tools synthesise and validate formal models. The first tool syntheses a formal Software Cost Reduction (SCR) requirements model from scenarios and the second tool combines the existing SCR simulator [97] with eBotworks 3D simulator to allow for simulation of continuous components. They focus on the verification of human-machine interaction.

Verma et al. [216] present a Flight Software simulator (SSIM - part of Rover Sequencing and Visualisation program (RSVP) suite) that is used to simulate MARS Rover missions. The simulator assists in predicting the behaviour of semi-autonomous systems by providing the capability for human operators to check if their intent is correctly captured by the robot prior to execution in different scenarios and environments.

Zhang et al. present RoadView [233] a photo-realistic simulator that tests performance of autonomous vehicles and evaluate their self-driving tasks. They make use of driving scenarios where they compare autonomous vehicle to a human driven scenario in order to demonstrate their tool.

Schöner presents a simulation tool that is part of the (industrial) Pegasus framework [196]. It integrates sensors, traffic and road models (in open-drive Format) into the simulation where different scenarios and situations are executed.

Model Based Testing Collet et al. [53] introduce RobTest, a tool for generating collision-free trajectories for stress testing of single-arm robots. It employs constraint programming techniques to solve continuous domain constraints in its trajectory generation process. The efficiency of such a process is evaluated in a controlled experiment where the generation time of acceptable near-optimal trajectories.

Formal Verification O'kelly introduces APEX [166], which is a formal verification tool for verifying vehicle dynamics, trajectory planning and tracking stacks of ADAS in vehicles. Property specifications are written in metric interval temporal logic. The tool calls DReach [117] in the background to perform reachability analysis on the vehicle trajectories.

Foughali et al. [75] implement an automatic translation from GenoM [146], a robotics model-based software engineering framework, to the formal specification language Fiacre [28], which can be fed into TINA for model checking (on the Petri Net models). They apply their tool to an autonomous ground vehicle (RMP 400 Segway)

Small

Simulation Regarding small scale case studies for simulation environments, the vast majority conduct a simple demonstration to illustrate features of their simulation tools [11, 42, 48, 50, 56, 96, 140, 233, 234], such as Mars [213] (for underwater robots) and VREP [185] (for generic robots) case studies.

Differently, Li et al. [132] employ a game-theoretical approach where vehicles have different levels of knowledge about other vehicles. For instance, a level 0 car has no knowledge about the other cars and a level k car has information about level k - 1 cars. Strangely, in their case study, they show that, in some instances, lower level cars effect less constraint violations.

MBT Two of the case studies in this category consider generating test cases from formal models [133, 164] of autonomous agents. Furthermore, Andrews et al. [10] model autonomous systems and their environment using Petri nets to generate test cases and apply their technique to a case study in the human-robot interaction domain. In Hagerman's case study [94], finite state machines are used to extract security test suites. Sagglietti et al. [191, 192] conduct a case study in which the reconfiguration behaviour of autonomous agents is verified. Betts et al. [30] compare the effectiveness of two search based testing methods, with a case study involving a UAV flight control software.

Formal Verification Several of the included case studies in this category concern abstract representations of multi-agent autonomous systems and provided efficient algorithms for parametric (formal) verification or state-space reduction techniques [18, 22, 24, 120, 121, 123, 136, 137].

Several other case studies [19, 46–48, 138, 205] concern systems used in model checking tools such as Prism [124] and FDR [85]. Another use of these case studies is to demonstrate usage of introduced tools such as APEX [166] and MDE [148]. Differently, Dennis et el. [60, 62] focus on formalising and verifying ethical concerns in BDI agents and provide corresponding small case studies. Aminof et al. [9] investigate the decidability problem for parameterised grid sizes. In their case study, they found that restricting the grid size, results the problem being solved in Pspace.

Runtime monitoring In the only paper in this category, Desai, Tomasso, and Seshia [63] make use of an STL-based (signal temporal logic) online monitoring system to ensure that the assumptions about the low-level controllers (discrete models) used during model checking hold at runtime. They demonstrate the strategy in a surveillance application case study.

Industrial

Simulation Zhou et al. [238] introduces a framework for virtual testing of advanced driver assistant systems that uses real world measurements. Shah et al. [199] build a model of quadrotor with pixhawk controller in their newly developed simulator, AirSim, that includes a physics engine and and supports real-time hardware-in-the-loop. Schöner presents a simulation tool that is part of the (industrial) Pegasus framework [196]. It integrates sensors, traffic and road models (in open-drive Format) into the simulation where different scenarios and situations are executed. [186] demonstrate their validation framework for autonomous systems in a farming context with simulations and field testing.

Uriagereka et al. [215] conduct simulation-assisted fault injection to assess safety and reliability of robotic systems. The feasibility of their method is demonstrated by applying it to the design of a real-time cartesian impedance control system. Manhaes and Rauschenbach [149] model the Sperre SF 30k ROV underwater robot (RexROV) in the demonstration of the simulator for unmanned underwater vehicles. Verma et al. [216] present a Flight Software simulator that is used to simulate MARS Rover missions. They demonstrate their approach with a corresponding case study. AbdElSalam et al. [1] use Hardware Emulation-in-the-loop to verify Electronic Control Units (ECUs) for ADAS systems.

MBT In the only industrial case study in this category, Abdessalem et al. [25] generate test cases for a system that can visually detect pedestrians in the context of advanced driver assistance systems (ADAS).

Formal Verification Gainer et al. [77] conduct a case study in the context of verification of human-robot interaction using the Care-O-Bot robotic assistant. Bhattacharyya et al. [31] apply formal verification methods to an autonomous CoPilot agent.

Runtime monitoring Gladisch et al. [88] select case studies that use industrial automated driving (adaptive cruise control, lane keeping, and steering control scenarios) to evaluate their search-based testing strategy.

Benchmarks

Simulation Several benchmark scale case studies can be found in this category. In what follows, we briefly discuss some of them. Wigand et al. [225] have developed CoSiMA, which is an architecture for simulation, execution and analysis of robotics system. They conduct experiments on the humanoid robot COMAN. Tosum et al. [214] present a design framework that facilitates the rapid creation of configurations and behaviours for modular robots. They've demonstrate their framework on the SMORES robot. Pereira et al. [173] employ several small case studies in their attempt to couple two simulators, namely SUMO and USAR-Sim. Brambilla et al. [37] model a probabilistic swarm that is checked in PRISM to evaluate their property-driven design method. Bohlmann, Klinger, and Szczerbicka [36] automatically generate a model of a parallel delta robot on-the-fly. Their method for model generation is based on machine learning and symbiotic simulation techniques. Mutter et al. [162] also explore the simulation of UAV models in Simulink and discuss the results when combining the platform and environment models. Bi et al. [32] present a deep Learning-based framework for traffic simulation and execute several scenarios of intersections with and without pedestrians. D'Urso, Santoro, and Santoro [67] simulate leader follower UAV scenarios in their framework. Their goal is to combine four simulation environments: a 3D visualisation engine, a physics simulator, a flight control stack and a network simulator. Wang and Cheng [220] present a hardware-in-the-loop simulator for drones that can generate synthetic images from the scene as datasets, detect and verify objects with a trained neural network, and generate point cloud data for model validation. They simulate and conduct filed testing on a physical UAV. Heitmeyer et al. [99] synthesise software cost-reduction models of multiple autonomous systems to be used in a simulator integrated with the eBotworks simulation tool.

MBT Proetzsch et al. [177] use a designed DSL (graph-based models) to describe the system behaviour of the autonomous off-road robot RAVON. The model is used is used as test model for generating test cases. Mullins et al. [160] have developed a tool (RAPT - Range Adversarial Planning Tool) for generating test scenarios to be employed on the System Under Test. Their tool is applied to realistic underwater missions. Furthermore, in their case study, Araiza-Illan, Pipe and Eder [14] use BDI models and model checking of probabilistic timed automata (in UPPAAL) to generate test sequences for human-robot collaboration tasks.

Formal Verification Here, we briefly discuss some of the benchmarks that involve formal verification. Halder et al. [95] use the physical robot Kobuki as a case study, over which properties are automatically verified using the UPPAAL model checker. The focus of their approach is to model and verify ROS systems using real time properties. Brambilla et al. [37] model a probabilistic swarm that is checked in PRISM to evaluate their property-driven design method. Bicevskis, Gaujens, and Kalnins [33] develop models for the testing of UAV and UGV collaboration in the Simulink environment. Althoff et al.

[6] propose a framework (IMPROV) for self-programming and self-verification for robots, which is demonstrated on a physical robotic arm. Zhao et al. [235] model an unmanned aerial vehicle (UAV) inspection mission on a wind farm and, via probabilistic model checking in PRISM, show how the battery features may affect verification results. Gruber and Althoff [92] present a reachability analysis tool (Spot) that finds counter-example to property violations. Their tool is evaluated using the CommonRoad benchmark PM1:MW1:DEU_Muc-3_1_T-1.

Runtime monitoring Pasareanu, Gopinath, and Yu [171] present a compositional approach for the verification of autonomous systems and apply the technique on a neural network implementation of a controller for a collision avoidance system on the ACAS Xu unmanned aircraft. Temporal properties are employed in Wang's case study [221], where the RoboticSpec specification language for robotic applications is translated into a framework for online monitoring that also uses PLTL properties. Huang et al. conduct a case study using a model of the LandShark UGV to demonstrate their tool, ROSRV [103], which is a runtime verification framework that that can be used with ROS. In the context of obstacle avoidance, Luo et al. [141] employ JavaMOP in their case study to verify that the robot does not behaves against requirements written in FSM and PTLT languages.

RQ1 By far quantitative testing techniques are the most widely researched strategies (this was also a common observation for the domain and connectivity aspects).

RQ2 Among the measures used for evaluating interventions efficiency is most often used, with effectinvess being a close second. Few interventions, however, were evaluated using a notion of coverage [14, 33, 94, 133, 144, 191, 192]. It is notable that, for runtime monitoring, only two publications [88, 103] employ an efficiency metric.

RQ3 There is a considerable lack of tools for model-based testing and runtime monitoring. For simulation and formal verification, there seems to be some considerable strength in terms of tool support.

RQ4 Approximately 54% of the interventions used small-scale case studies for their evaluations, whilst only 10% evaluated their strategy in an industrial context, indicating a clear gap.

6.2 For researchers

Throughout the various categories we have coded in this case study, the most prominent gap is in the use of agreed-upon rigorous measures to evaluate the efficiency and effectiveness of the interventions as well as real-world benchmarks that can be used to evaluate such measures. As observed in the earlier sections, much of the measures of efficiency and effectiveness measures are very generic and there is also a relative gap in domain-specific measures suitable for the RAS sub-domains. A lack of domain-specific modelling languages and the limited number of runtime verification approaches indicate that there is room for improvement in RAS testing strategies.

Another considerable gap is in the use of quantitative specification languages to specify the desired properties of the system; due to the inherent heterogeneity of RAS, we need to have property languages that cover aspects such as the combination of discrete and continuous dynamics as well as stochastic and epistemic aspects that may be used to model the aspects of behaviour concerning the environment and the users. Connected to this point is the relative gap in interventions that perform a quantitative analysis of the system and provide quantitative metrics of quality as the outcome of the test. Some starting points in this direction are the use of quantitative properties that incorporate probabilistic and stochastic- [37, 172], timed- [95, 166, 202], and continuous dynamical [63] aspects of RAS. We have also noted the use of a specification language that caters for a combination of stochastic and continuous aspect of RAS [138]. On the contrary there is a relative strength in using qualitative models, including property specification languages as predicate- [8] and temporal logics [19, 41, 75, 104, 108, 141, 171, 204], as well as epistemic extensions thereof [122, 135]. Also, there is a wealth of studies on the use of discrete relational [150], state-based [37, 47, 94, 141, 157, 229, 231] and belief-based [13, 14, 108, 164] abstract models in testing and verification of RAS. Also several studies used informal simulation models for simulation tools such as Gazebo and USARSim [13, 14, 42, 42, 51, 103, 128, 173]. A suitable middle-ground may be the semi-formal and domain-specific models such as those built in Matlab/Simulink [25, 30, 162].

Regarding techniques, most of the techniques used so far in the literature have been formal verification techniques applied (on relatively high-level) qualitative [8, 41, 71, 122, 135, 148, 188, 229, 231] or quantitative [12, 19, 33, 37, 47, 63, 74, 75, 95, 104, 108, 111, 138, 157, 166, 171, 172, 190, 202, 219, 226] models of RAS. There is also some strength in the use of informal simulation techniques [37, 42, 127, 131, 132, 140, 157, 162, 173, 193, 233]. We have seen relatively few model-based testing [10, 14, 25, 30, 94, 133, 164, 177, 192] and run-time verification [63, 103, 141, 171] techniques that have been applied to (models of) complex and detailed RAS. We hence see a gap, and a trend towards closing this gap, in dynamic and non-exhaustive testing of RAS techniques.

Finally, lack of public tooling is a major gap observed in the literature. They are very few techniques that are accompanied by a tool and there are very few public tools for testing RAS [47, 71, 75, 103, 122, 140, 157, 219, 233].

6.3 For practitioners

The most significant gap is lack of industrial evaluation of existing interventions. There have been very few interventions applied in an industrial context and to systems of industrial complexity [1, 2, 25, 31, 70, 77, 88, 106, 114, 149, 154, 186, 186, 195, 196, 198, 199, 209, 215, 216]

Unfortunately the number of interventions is too small to conclude any meaningful trend and indication of strong evidence for applicability in the industrial setting. Among the propose interventions, most either concerned simulation-based testing [149, 216] or connected the results of their verification to some simulation tool (mostly based on ROS-Gazebo integration) [1, 70, 154]. Search-based testing [2, 25, 88] and interaction testing [2, 209] are two notable techniques that have been used in industrial contexts. Among the models employed in the industrial context, variants of state machines [216] and fault trees [215] can be mentioned. A notable study in this regard [198] is a comparison of supervisory-control, deductive- and inductive (model-checking) verification techniques in the industrial context.

The human- and information-source is another aspect of testing interventions that is a severely understudied. We note a recent trend in combining user studies (in the sense of human-computer- and human-robot interactions) and traditional testing, validation, and verification techniques [6, 151, 189].

Also there is a gap in defining and evaluating testing processes, particularly in industrial contexts.

The lack of industrial- and domain-expert input into the models and techniques is evident and has led to generic and relatively simple modelling techniques and property languages being used for most intervention. Co-production with industrial partners can enrich these aspects and lead to models that can deal with the heterogeneity and complexity of industrial RAS.

7 CONCLUSION

We performed a systematic review of the interventions for testing robotics and autonomous systems in order to answer the following research questions:

(1) What are the *types of models* used for testing RAS?
(2) Which *efficiency and effectiveness measures* were introduced or used to evaluate RAS testing interventions?
(3) What are the interventions supported by *(publicly available) tools* in this domain?
(4) Which interventions have *evidence of applicability* to large-scale and industrial systems?

To this end, we started off by performing a pilot study on a seed of 26 papers. Using this pilot study, we designed and validated a search query, designed rigorous inclusion and exclusion criteria and developed an adaptation of the SERP-Test taxonomy. Subsequently, we went through two phases of search, validation and coding, in total going through a total of 10,534 papers. We finally coded the set of 192 included papers and analysed them to answer our research questions.

A summary of the findings of the review with regards to our research questions is provided below:

(1) There is a wealth of formal and informal models used for testing RAS. In particular, there is a sizeable literature on using generic property specification languages (such as linear temporal logic) and qualitative modelling languages, such as variants of state machines, UML diagrams, Petri nets and process algebras. There is a clear gap in quantitative modelling languages that can capture the complex and heterogeneous nature of RAS. There is also a lack of domain-specific languages that can capture domain knowledge for various sub-domains of RAS.
(2) We observed a gap in rigorous and widely accepted metrics to measure effectiveness and efficiency, and adequacy of testing interventions. Similar to the previous items, those measures used in the literature are very generic and do not pertain to the domain specific aspects of RAS. Hence, there is a gap and a research opportunity for defining and evaluating rigorous (domain-specific) measures for efficiency, effectiveness, and adequacy for RAS testing interventions.
(3) There is a considerable number of intervention that rely on public tools to implement or evaluate their interventions. However, there are very few which make their proposed / evaluated interventions available for public use in terms of publicly available tools. There is hence a considerable gap in providing data-sets and public tools for further development of the field.
(4) There are less than a handful of testing interventions that have been evaluated in an industrial context. There have been some other interventions that used some real robots or autonomous systems, but in an academic

context. This signifies the importance of future co-production between academia and industry in industrial evaluation of testing interventions for RAS.

ACKNOWLEDGMENTS

We would like to thank Jan Tretmans and Wojciech Mostowski for comments and discussions at the early stage of this research. Moreover, we would like to thank Thomas Arts, Michael Fisher, Mario Gleirscher, Robert Hierons, Fabio Palomba, and, Kristin Rozier for their comments at the validation stage of this study. Hugo Araujo and Mohammad Reza Mousavi have been partially supported by the UKRI Trustworthy Autonomous Systems Node in Verifiability, Grant Award Reference EP/V026801/2.

REFERENCES

[1] Mohamed AbdElSalam, Keroles Khalil, John Stickley, Ashraf Salem, and Bruno Loye. 2019. Verification of advanced driver assistance systems (ADAS) and autonomous vehicles with hardware emulation-in-the-loop. *International journal of automotive engineering* 10, 2 (2019), 197–204.

[2] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C Briand, and Thomas Stifter. 2018. Testing autonomous cars for feature interaction failures using many-objective search. In *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, Montpellier, France, 143–154.

[3] Nauman Bin Ali, Emelie Engström, Masoumeh Taromirad, Mohammad Reza Mousavi, Nasir Mehmood Minhas, Daniel Helgesson, Sebastian Kunze, and Mahsa Varshosaz. 2019. On the search for industry-relevant regression testing research. *Empir. Softw. Eng.* 24, 4 (2019), 2020–2055. https://doi.org/10.1007/s10664-018-9670-1

[4] Ala Jamil Alnaser, Mustafa Ilhan Akbas, Arman Sargolzaei, and Rahul Razdan. 2019. Autonomous vehicles scenario testing framework and model of computation. *SAE International Journal of Connected and Automated Vehicles* 2, 4 (2019), 60617 – 60628.

[5] Matthias Althoff and John M Dolan. 2011. Set-based computation of vehicle behaviors for the online verification of autonomous vehicles. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Washington, DC, USA, 1162–1167.

[6] Matthias Althoff, Andrea Giusti, Stefan B Liu, and Aaron Pereira. 2019. Effortless creation of safe robots from modules through self-programming and self-verification. *Science Robotics* 4, 31 (2019), 56–89.

[7] Saifullah Amin, Adnan Elahi, Kashif Saghar, and Faran Mehmood. 2017. Formal modelling and verification approach for improving probabilistic behaviour of robot swarms. In *2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*. IEEE, New York, NY, USA, 392–400.

[8] Benjamin Aminof, Aniello Murano, Sasha Rubin, and Florian Zuleger. 2015. Verification of asynchronous mobile-robots in partially-known environments. In *International Conference on Principles and Practice of Multi-Agent Systems*. Springer, Bertinoro, Italy, 185–200.

[9] Benjamin Aminof, Aniello Murano, Sasha Rubin, and Florian Zuleger. 2016. Automatic verification of multi-agent systems in parameterised grid-environments. In *Proceedings of the 2016 international conference on autonomous agents & multiagent systems*. ACM, Singapore, Singapore, 1190–1199.

[10] Anneliese Andrews, Mahmoud Abdelgawad, and Ahmed Gario. 2016. World model for testing urban search and rescue (USAR) robots using petri nets. In *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*. IEEE, Rome, Italy, 663–670.

[11] Vimal Rau Aparow, Apratim Choudary, Giridharan Kulandaivelu, Thomas Webster, Justin Dauwels, and Niels de Boer. 2019. A Comprehensive Simulation Platform for Testing Autonomous Vehicles in 3D Virtual Environment. In *2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*. IEEE, Singapore, 115–119.

[12] Ryota Arai and H Schlingloff. 2017. Model-based performance prediction by statistical model checking an industrial case study of autonomous transport robots. In *Proc. 25th CS&P 2017-Concurrency, Specification and Programming*. CEUR, Olsztyn, Poland.

[13] Dejanira Araiza-Illan, Anthony G Pipe, and Kerstin Eder. 2016. Intelligent agent-based stimulation for testing robotic software in human-robot interactions. In *Proceedings of the 3rd Workshop on Model-Driven Robot Software Engineering*. ACM, Leipzig, Germany, 9–16.

[14] Dejanira Araiza-Illan, Tony Pipe, and Kerstin Eder. 2016. Model-Based Testing, Using Belief-Desire-Intentions Agents, of Control Code for Robots in Collaborative Human-Robot Interactions. *arXiv preprint arXiv:1603.00656* 1 (2016), 1–16.

[15] Rafael Araújo, Alexandre Mota, and Sidney Nogueira. 2017. Analyzing Cleaning Robots Using Probabilistic Model Checking. In *International Conference on Information Reuse and Integration*. Springer, Los Angeles, CA, USA, 23–51.

[16] Johan Arcile, Raymond Devillers, and Hanna Klaudel. 2019. VerifCar: a framework for modeling and model checking communicating autonomous vehicles. *Autonomous agents and multi-agent systems* 33, 3 (2019), 353–381.

[17] Radu F Babiceanu and Remzi Seker. 2017. Formal verification of trustworthiness requirements for Small Unmanned Aerial Systems. In *2017 Integrated Communications, Navigation and Surveillance Conference (ICNS)*. IEEE, Herndon, VA, USA, 6A3–1.

[18] Ran Bao, Christian Attiogbe, Benoit Delahaye, Paulin Fournier, and Didier Lime. 2019. Parametric statistical model checking of UAV flight plan. In *International Conference on Formal Techniques for Distributed Objects, Components, and Systems*. Springer, Copenhagen, Denmark, 57–74.

[19] Benoît Barbot, Béatrice Bérard, Yann Duplouy, and Serge Haddad. 2017. Statistical model-checking for autonomous vehicle safety validation. In *Conference SIA Simulation Numérique*. HAL-Inria, Montigny-le-Bretonneux, France.

[20] Halil Beglerovic, Steffen Metzner, and Martin Horn. 2018. Challenges for the Validation and Testing of Automated Driving Functions. https://doi.org/10.1007/978-3-319-66972-4_15

[21] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. 2011. SUMO–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, Barcelona, Spain.

[22] Francesco Belardinelli, Panagiotis Kouvaros, and Alessio Lomuscio. 2017. Parameterised Verification of Data-aware Multi-Agent Systems.. In *IJCAI*. ACM, Melbourne, Australia, 98–104.

[23] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. 2017. Verification of Broadcasting Multi-Agent Systems against an Epistemic Strategy Logic.. In *IJCAI*, Vol. 17. ACM, Melbourne, Australia, 91–97.

[24] Francesco Belardinelli, Alessio Lomuscio, Aniello Murano, and Sasha Rubin. 2017. Verification of Multi-agent Systems with Imperfect Information and Public Actions.. In *AAMAS*, Vol. 17. ACM, São Paulo, Brasil, 1268–1276.

[25] Raja Ben Abdessalem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. 2016. Testing advanced driver assistance systems using multi-objective search and neural networks. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. IEEE/ACM, New York, NY, USA, 63–74.

[26] Luca Benvenuti, Davide Bresolin, Pieter Collins, Alberto Ferrari, Luca Geretti, and Tiziano Villa. 2014. Assume–guarantee verification of nonlinear hybrid systems with Ariadne. *International Journal of Robust and Nonlinear Control* 24, 4 (2014), 699–724.

[27] Christian Berger. 2015. Accelerating Regression Testing for Scaled Self-Driving Cars with Lightweight Virtualization–A Case Study. In *2015 IEEE/ACM 1st International Workshop on Software Engineering for Smart Cyber-Physical Systems*. IEEE, Florence, Italy, 2–7.

[28] Bernard Berthomieu, Jean-Paul Bodeveix, Patrick Farail, Mamoun Filali, Hubert Garavel, Pierre Gaufillet, Frederic Lang, and François Vernadat. 2008. Fiacre: an intermediate language for model verification in the topcased environment. In *4th European Congress ERTS Embedded Real Time Software (ERTS 2008)*. SEE, Toulouse, France, 8p.

[29] Bernard Berthomieu*, P-O Ribet, and François Vernadat. 2004. The tool TINA–construction of abstract state spaces for Petri nets and time Petri nets. *International journal of production research* 42, 14 (2004), 2741–2756.

[30] Kevin M Betts and Mikel D Petty. 2016. Automated search-based robustness testing for autonomous vehicle software. *Modelling and Simulation in Engineering* 2016 (2016).

[31] Siddhartha Bhattacharyya, Thomas C Eskridge, Natasha A Neogi, Marco Carvalho, and Milton Stafford. 2018. Formal Assurance for Cooperative Intelligent Autonomous Agents. In *NASA Formal Methods Symposium*. Springer, Houston, TX, USA, 20–36.

[32] Huikun Bi, Tianlu Mao, Zhaoqi Wang, and Zhigang Deng. 2019. A Deep Learning-based Framework for Intersectional Traffic Simulation and Editing. *IEEE Transactions on Visualization and Computer Graphics* 1 (2019).

[33] Janis BICEVSKIS, Artis GAUJENS, and Janis KALNINS. 2013. Testing of RUAV and UGV Robots' Collaboration in the Simulink Environment. *Baltic Journal of Modern Computing* 1 (2013).

[34] Andreas Bihlmaier and Heinz Wörn. 2014. Robot unit testing. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, Bergamo, Italy, 255–266.

[35] Eckard Böde, Matthias Büker, Ulrich Eberle, Martin Fränzle, Sebastian Gerwinn, and Birte Kramer. 2018. Efficient splitting of test and simulation cases for the verification of highly automated driving functions. In *International Conference on Computer Safety, Reliability, and Security*. Springer, Västerås, Sweden, 139–153.

[36] Sebastian Bohlmann, Volkhard Klinger, and Helena Szczerbicka. 2017. Integration of a physical system, machine learning, simulation, validation and control systems towards symbiotic model engineering. In *Proceedings of the Symposium on Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems*. ACM, Virginia, USA, 1–12.

[37] Manuele Brambilla, Arne Brutschy, Marco Dorigo, and Mauro Birattari. 2014. Property-driven design for robot swarms: A design method based on prescriptive modeling and model checking. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 9, 4 (2014), 1–28.

[38] Paul Bremner, Louise A Dennis, Michael Fisher, and Alan F Winfield. 2019. On proactive, transparent, and verifiable ethical reasoning for robots. *Proc. IEEE* 107, 3 (2019), 541–561.

[39] Davide Bresolin, Luca Geretti, Riccardo Muradore, Paolo Fiorini, and Tiziano Villa. 2015. Formal verification applied to robotic surgery. In *Coordination Control of Distributed Systems*. Springer, Italy, 347–355.

[40] Davide Bresolin, Luca Geretti, Riccardo Muradore, Paolo Fiorini, and Tiziano Villa. 2015. Formal verification of robotic surgery tasks by reachability analysis. *Microprocessors and Microsystems* 39, 8 (2015), 836–842.

[41] Julien Brunel and Jacques Cazin. 2012. Formal verification of a safety argumentation and application to a complex UAV system. In *International Conference on Computer Safety, Reliability, and Security*. Springer, Magdeburg, Germany, 307–318.

[42] Qing Bu, Fuhua Wan, Zhen Xie, Qinhu Ren, Jianhua Zhang, and Sheng Liu. 2015. General simulation platform for vision based UAV testing. In *2015 IEEE International Conference on Information and Automation*. IEEE, Lijiang, Yunnan, China, 2512–2516.

[43] Don Burns and Robert Osfield. 2004. Tutorial: open scene graph A: introduction tutorial: open scene graph B: examples and applications. In *IEEE Virtual Reality 2004*. IEEE, Chicago, IL, USA, 265–265.

[44] IPG CarMaker. 2014. Users guide version 4.5. 2. *IPG Automotive, Karlsruhe, Germany* 1 (2014).

[45] Stefano Carpin, Mike Lewis, Jijun Wang, Stephen Balakirsky, and Chris Scrapper. 2007. USARSim: a robot simulator for research and education. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, Roma, Italy, 1400–1405.

[46] Ana Cavalcanti, James Baxter, Robert M Hierons, and Raluca Lefticaru. 2019. Testing Robots Using CSP. In *International Conference on Tests and Proofs*. Springer, Porto, Portugal, 21–38.

[47] Ana Cavalcanti, Alvaro Miyazawa, Augusto Sampaio, Wei Li, Pedro Ribeiro, and Jon Timmis. 2018. Modelling and verification for swarm robotics. In *International Conference on Integrated Formal Methods*. Springer, Maynooth, Ireland, 1–19.

[48] Ana Cavalcanti, Augusto Sampaio, Alvaro Miyazawa, Pedro Ribeiro, Madiel Conserva Filho, André Didier, Wei Li, and Jon Timmis. 2019. Verified simulation for robotics. *Science of Computer Programming* 174 (2019), 1–37.

[49] Qianwen Chao, Xiaogang Jin, Hen-Wei Huang, Shaohui Foong, Lap-Fai Yu, and Sai-Kit Yeung. 2019. Force-based heterogeneous traffic simulation for autonomous vehicle testing. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, Montreal, Canada, 8298–8304.

[50] Shitao Chen, Yu Chen, Songyi Zhang, and Nanning Zheng. 2019. A novel integrated simulation and testing platform for self-driving cars with hardware in the loop. *IEEE Transactions on Intelligent Vehicles* 4, 3 (2019), 425–436.

[51] Yu Chen, Shitao Chen, Tangyike Zhang, Songyi Zhang, and Nanning Zheng. 2018. Autonomous vehicle testing and validation platform: Integrated simulation system with hardware in the loop. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Changshu, Suzhou, China, 949–956.

[52] Patryk Cieślak. 2019. Stonefish: An Advanced Open-Source Simulation Tool Designed for Marine Robotics, With a ROS Interface. In *OCEANS 2019*. IEEE, Marseille, France, 1–6.

[53] Mathieu Collet, Arnaud Gotlieb, Nadjib Lazaar, and Morten Mossige. 2019. Stress testing of single-arm robots through constraint-based generation of continuous trajectories. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, Newark, CA, USA, 121–128.

[54] Marc Compere, Garrett Holden, Otto Legon, and Roberto Martinez Cruz. 2019. MoVE: A Mobility Virtual Environment for Autonomous Vehicle Testing. In *ASME International Mechanical Engineering Congress and Exposition*, Vol. 59414. American Society of Mechanical Engineers, Salt Lake City, Utah, USA, V004T05A097.

[55] A. Cortesi, P. Ferrara, and N. Chaki. 2013. Static analysis techniques for robotics software verification. In *IEEE ISR 2013*. IEEE, KINTEX, Seoul, Korea, 1–6.

[56] Piotr Cybulski. 2019. A Framework for Autonomous UAV Swarm Behavior Simulation. In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, Leipzig, Germany, 471–478.

[57] Werner Damm and Roland Galbas. 2018. Exploiting learning and scenario-based specification languages for the verification and validation of highly automated driving. In *2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)*. IEEE, Gothenburg, Sweden, 39–46.

[58] Vânia de Oliveira Neves, Márcio Eduardo Delamaro, and Paulo Cesar Masiero. 2019. Automated Structural Software Testing of Autonomous Vehicles. In *20th Conferencia Iberoamericana en Software Engineering*. CIbSE, Buenos Aires, Argentina.

[59] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[60] Louise Dennis, Michael Fisher, Marija Slavkovik, and Matt Webster. 2016. Formal verification of ethical choices in autonomous systems. *Robotics and Autonomous Systems* 77 (2016), 1–14.

[61] Louise A. Dennis. 2018. The MCAPL Framework including the Agent Infrastructure Layer and Agent Java Pathfinder. *Journal of Open Source Software* 3, 24 (2018), 617. https://doi.org/10.21105/joss.00617

[62] Louise A Dennis, Michael Fisher, and Alan FT Winfield. 2015. Towards verifiably ethical robot behaviour. In *WORKSHOPS AT THE TWENTY-NINTH AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE (AAAI-15)*. IEEE, Austin, Texas USA.

[63] Ankush Desai, Tommaso Dreossi, and Sanjit A. Seshia. 2017. Combining Model Checking and Runtime Verification for Safe Robotics. In *Runtime Verification*, Shuvendu Lahiri and Giles Reger (Eds.). Springer International Publishing, Cham, 172–189.

[64] Ha Thi Thu Doan, François Bonnet, and Kazuhiro Ogata. 2018. Model checking of robot gathering. In *21st International Conference on Principles of Distributed Systems (OPODIS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Lisbon, Portugal.

[65] Simulink Documentation. 2020. Simulation and Model-Based Design. https://www.mathworks.com/products/simulink.html

[66] Daniela Doroftei, Anibal Matos, Eduardo Silva, Victor Lobo, Rene Wagemans, and Geert De Cubber. 2015. Operational validation of robots for risky environments. In *8th IARP Workshop on Robotics for Risky Environments*. IARP/EURON, Benicàssim, SPAIN.

[67] Fabio D'Urso, Corrado Santoro, and Federico Fausto Santoro. 2019. An integrated framework for the realistic simulation of multi-UAV applications. *Computers & Electrical Engineering* 74 (2019), 196–209.

[68] Emelie Engström, Kai Petersen, Nauman bin Ali, and Elizabeth Bjarnason. 2017. SERP-test: a taxonomy for supporting industry–academia communication. *Software Quality Journal* 25 (2017), 1269–1305.

[69] Giuseppe Airò Farulla and Anna-Lena Lamprecht. 2017. Model checking of security properties: a case study on human-robot interaction processes. In *2017 12th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS)*. IEEE, Mallorca, Spain, 1–6.

[70] S Alireza Fayazi, Ardalan Vahidi, and Andre Luckow. 2019. A Vehicle-in-the-Loop (VIL) verification of an all-autonomous intersection control scheme. *Transportation Research Part C: Emerging Technologies* 107 (2019), 193–210.

[71] Lucas ER Fernandes, Vinicius Custodio, Gleifer V Alves, and Michael Fisher. 2017. A rational agent controlling an autonomous vehicle: Implementation and formal verification. *arXiv preprint arXiv:1709.02557* 1 (2017).

[72] Angelo Ferrando, Louise A Dennis, Davide Ancona, Michael Fisher, and Viviana Mascardi. 2018. Verifying and validating autonomous systems: Towards an integrated approach. In *International Conference on Runtime Verification*. Springer, Limassol, Cyprus, 263–281.

[73] Mohammed Foughali. 2019. On Reconciling Schedulability Analysis and Model Checking in Robotics. In *International Conference on Model and Data Engineering*. Springer, Tolouse, France, 32–48.

[74] Mohammed Foughali, Bernard Berthomieu, Silvano Dal Zilio, Pierre-Emmanuel Hladik, Félix Ingrand, and Anthony Mallet. 2018. Formal verification of complex robotic systems on resource-constrained platforms. In *2018 IEEE/ACM 6th International FME Workshop on Formal Methods in Software Engineering (FormaliSE)*. IEEE, Gothenburg, Sweden, 2–9.

[75] Mohammed Foughali, Bernard Berthomieu, Silvano Dal Zilio, Félix Ingrand, and Anthony Mallet. 2016. Model checking real-time properties on the functional layer of autonomous robots. In *International Conference on Formal Engineering Methods*. Springer, Tokyo, Japan, 383–399.

[76] National Science Foundation. 2018. Smart and Autonomous Systems (S&AS) Program Solicitation. Available at https://www.nsf.gov/pubs/2018/nsf18557/nsf18557.htm. Last accessed 04 March 2022.

[77] Paul Gainer, Clare Dixon, Kerstin Dautenhahn, Michael Fisher, Ullrich Hustadt, Joe Saunders, and Matt Webster. 2017. CRutoN: Automatic verification of a robotic assistant's behaviours. In *Critical Systems: Formal Methods and Automated Verification*. Springer, Turin, Italy, 119–133.

[78] Alessio Gambi, Marc Mueller, and Gordon Fraser. 2019. Automatically testing self-driving cars with search-based procedural content generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. ACM, Beijing, China, 318–328.

[79] Shenjian Gao and Yanwen Tan. 2017. Paving the Way for Self-driving Cars - Software Testing for Safety-critical Systems Based on Machine Learning : A Systematic Mapping Study and a Survey.

[80] Mario Garzón and Anne Spalanzani. 2018. An hybrid simulation tool for autonomous cars in very high traffic scenarios. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, Singapore, 803–808.

[81] Lydia Gauerhof, Peter Munk, and Simon Burton. 2018. Structuring validation targets of a machine learning function applied to automated driving. In *International Conference on Computer Safety, Reliability, and Security*. Springer, Västerås, Sweden, 45–58.

[82] Luca Geretti, Riccardo Muradore, Davide Bresolin, Paolo Fiorini, and Tiziano Villa. 2017. Parametric formal verification: the robotic paint spraying case study. *IFAC-PapersOnLine* 50, 1 (2017), 9248–9253.

[83] Achim Gerstenberg and Martin Steinert. 2019. Evaluating and Optimizing Chaotically Behaving Mobile Robots with a Deterministic Simulation. *Procedia CIRP* 84 (2019), 219–224.

[84] Carlo Ghezzi, Dino Mandrioli, and Angelo Morzenti. 1990. TRIO: A logic language for executable specifications of real-time systems. *Journal of Systems and Software* 12, 2 (1990), 107–123. https://doi.org/10.1016/0164-1212(90)90074-V The Role of Language in Programming.

[85] Thomas Gibson-Robinson, Philip Armstrong, Alexandre Boulgakov, and A.W. Roscoe. 2014. FDR3 — A Modern Refinement Checker for CSP. *Lecture Notes in Computer Science* 8413 (2014), 187–201.

[86] Thomas Gibson-Robinson, Philip Armstrong, Alexandre Boulgakov, and Andrew W Roscoe. 2014. FDR3—a modern refinement checker for CSP. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, Grenoble, France, 187–201.

[87] Edmond Gjondrekaj, Michele Loreti, Rosario Pugliese, Francesco Tiezzi, Carlo Pinciroli, Manuele Brambilla, Mauro Birattari, and Marco Dorigo. 2012. Towards a formal verification methodology for collective robotic systems. In *International Conference on Formal Engineering Methods*. Springer, Kyoto, Japan, 54–70.

[88] Christoph Gladisch, Thomas Heinz, Christian Heinzemann, Jens Oehlerking, Anne von Vietinghoff, and Tim Pfitzer. 2019. Experience paper: search-based testing in automated driving control applications. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, San Diego, California, 26–37.

[89] Mario Gleirscher, Simon Foster, and Yakoub Nemouchi. 2019. Evolution of formal model-based assurance cases for autonomous robots. In *International Conference on Software Engineering and Formal Methods*. Springer, Oslo, Norway, 87–104.

[90] Mario Gleirscher, Simon Foster, and Jim Woodcock. 2020. New Opportunities for Integrated Formal Methods. *ACM Comput. Surv.* 52, 6 (2020), 117:1–117:36. https://doi.org/10.1145/3357231

[91] João SV Gonçalves, João Jacob, Rosaldo JF Rossetti, António Coelho, and Rui Rodrigues. 2015. An integrated framework for mobile-based ADAS simulation. In *Modeling Mobility with Open Data*. Springer, Berlin, Germany, 171–186.

[92] Felix Gruber and Matthias Althoff. 2018. Anytime safety verification of autonomous vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Maui, Hawaii, USA, 1708–1714.

[93] KM Gupta and K Gillespie. 2015. eBotworks: A software platform for developing and evaluating communicative autonomous systems. *AUVSI Unmanned Systems, Atlanta, GA* 1 (2015).

[94] Seana Hagerman, Anneliese Andrews, and Stephen Oakes. 2016. Security testing of an unmanned aerial vehicle (UAV). In *2016 Cybersecurity Symposium (CYBERSEC)*. IEEE, Coeur d'Alene, ID, USA, 26–31.

[95] Raju Halder, José Proença, Nuno Macedo, and André Santos. 2017. Formal verification of ROS-based robotic applications using timed-automata. In *2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE)*. IEEE, Buenos Aires, Argentina, 44–50.

[96] Jani Erik Heikkinen, Salimzhan Gafurov, Sergey Kopylov, Tatiana Minav, Sergey Grebennikov, and Artur Kurbanov. 2019. Hardware-in-the-Loop Platform for Testing Autonomous Vehicle Control Algorithms. In *2019 12th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, Kazan, Russia, 906–911.

[97] Constance Heitmeyer, Myla Archer, Ramesh Bharadwaj, and Ralph Jeffords. 2005. *Tools for constructing requirements specification: the SCR toolset at the age of ten.* Technical Report. Naval Research Lab Washington DC Center for High Assurance Computing Systems ....

[98] Constance L Heitmeyer. 2002. Software cost reduction. *Encyclopedia of software engineering* 1 (2002).

[99] Constance L Heitmeyer and Elizabeth I Leonard. 2015. Obtaining trust in autonomous systems: tools for formal model synthesis and validation. In *2015 IEEE/ACM 3rd FME Workshop on Formal Methods in Software Engineering*. IEEE, Florence, Italy, 54–60.

[100] Philipp Helle, Wladimir Schamai, and Carsten Strobel. 2016. Testing of Autonomous Systems – Challenges and Current State-of-the-Art. *INCOSE International Symposium* 26, 1 (2016), 571–584. https://doi.org/10.1002/j.2334-5837.2016.00179.x

[101] Ayonga Hereid and Aaron D Ames. 2017. FROST*: Fast robot optimization and simulation toolkit. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Vancouver, BC, Canada, 719–726.

[102] Charles Antony Richard Hoare. 1978. Communicating sequential processes. *Commun. ACM* 21, 8 (1978), 666–677.

[103] Jeff Huang, Cansu Erdogan, Yi Zhang, Brandon Moore, Qingzhou Luo, Aravind Sundaresan, and Grigore Rosu. 2014. ROSRV: Runtime verification for robots. In *International Conference on Runtime Verification*. Springer, Toronto, ON, Canada, 247–254.

[104] Laura R Humphrey. 2013. Model checking for verification in UAV cooperative control applications. *Recent Advances in Research on Unmanned Aerial Vehicles* 444 (2013), 69–117.

[105] David Husch and John Albeck. 2004. Trafficware SYNCHRO 6 user guide. *TrafficWare, Albany, California* 11 (2004).

[106] Adam Jacoff, Hui-Min Huang, Elena Messina, Ann Virts, and Anthony Downs. 2010. Comprehensive standard test suites for the performance evaluation of mobile robots. In *Proceedings of the 10th Performance Metrics for Intelligent Systems Workshop*. ACM, Baltimore, Maryland, USA, 161–168.

[107] Andreas Junghanns, Jakob Mauss, Mugur Tatar, et al. 2008. Tatar: Testweaver-a tool for simulation-based test of mechatronic designs. In *6th International Modelica Conference*. Citeseer, Bielefeld, Germany.

[108] Maryam Kamali, Louise A Dennis, Owen McAree, Michael Fisher, and Sandor M Veres. 2017. Formal verification of autonomous vehicle platooning. *Science of computer programming* 148 (2017), 88–106.

[109] Y. Kang, H. Yin, and C. Berger. 2019. Test Your Self-Driving Algorithm: An Overview of Publicly Available Driving Datasets and Virtual Testing Environments. *IEEE Transactions on Intelligent Vehicles* 4, 2 (2019), 171–185.

[110] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. 2018. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, Porto, Portugal, 287–296.

[111] Hojat Khosrowjerdi and Karl Meinke. 2018. Learning-based testing for autonomous systems using spatial and temporal requirements. In *Proceedings of the 1st International Workshop on Machine Learning and Software Engineering in Symbiosis*. ACM, Montpellier, France, 6–15.

[112] Baekgyu Kim, Yusuke Kashiba, Siyuan Dai, and Shinichi Shiraishi. 2016. Testing autonomous vehicle software in the virtual prototyping environment. *IEEE Embedded Systems Letters* 9, 1 (2016), 5–8.

[113] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, Montreal, QC, Canada, 1039–1049.

[114] Florian Klück, Martin Zimmermann, Franz Wotawa, and Mihai Nica. 2019. Genetic algorithm-based test parameter optimization for ADAS system testing. In *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, Sofia, Bulgaria, 418–425.

[115] A. Knauss, J. Schroder, C. Berger, and H. Eriksson. 2017. Software-Related Challenges of Testing Automated Vehicles. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, Buenos Aires, Argentina, 328–330.

[116] Nathan Koenig and Andrew Howard. 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, Vol. 3. IEEE, Sendai, Japan, 2149–2154.

[117] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. 2015. dReach: δ-reachability analysis for hybrid systems. In *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*. Springer, London, United Kingdom, 200–205.

[118] Twan Koolen and Robin Deits. 2019. Julia for robotics: Simulation and real-time control in a high-level programming language. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, Montreal, Canada, 604–611.

[119] Philip Koopman and Michael Wagner. 2016. Challenges in Autonomous Vehicle Testing and Validation. *SAE International Journal of Transportation Safety* 4 (04 2016), 15–24. https://doi.org/10.4271/2016-01-0128

[120] Panagiotis Kouvaros and Alessio Lomuscio. 2015. A counter abstraction technique for the verification of robot swarms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29. AAAI PRESS, Austin, Texas, USA.

[121] Panagiotis Kouvaros and Alessio Lomuscio. 2015. Verifying emergent properties of swarms. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*. AAAI Press, Buenos Aires, Argentina.

[122] Panagiotis Kouvaros and Alessio Lomuscio. 2016. Formal verification of opinion formation in swarms. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. ACM, Singapore, 1200–1208.

[123] Panagiotis Kouvaros, Alessio Lomuscio, Edoardo Pirovano, and Hashan Punchihewa. 2019. Formal verification of open multi-agent systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. ACM, Montreal, QC, Canada, 179–187.

[124] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *International conference on computer aided verification*. Springer, Snowbird, UT, USA, 585–591.

[125] John E Laird. 2019. The Soar cognitive architecture.

[126] Kim G Larsen, Paul Pettersson, and Wang Yi. 1997. UPPAAL in a nutshell. *International journal on software tools for technology transfer* 1, 1-2 (1997), 134–152.

[127] Adrien Lasbouygues, Benoit Ropars, Robin Passama, David Andreu, and Lionel Lapierre. 2015. Atoms based control of mobile robots with Hardware-In-the-Loop validation. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Hamburg, Germany, 1083–1090.

[128] Jannik Laval, Luc Fabresse, and Noury Bouraqadi. 2013. A methodology for testing mobile autonomous robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Tokyo, Japan, 1842–1847.

[129] Philippe Ledent, Anshul Paigwar, Alessandro Renzaglia, Radu Mateescu, and Christian Laugier. 2019. Formal Validation of Probabilistic Collision Risk Estimation for Autonomous Driving. In *2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. IEEE, Bangkok, Thailand, 433–438.

[130] Li Li, Wu-Ling Huang, Yuehu Liu, Nan-Ning Zheng, and Fei-Yue Wang. 2016. Intelligence testing for autonomous vehicles: A new approach. *IEEE Transactions on Intelligent Vehicles* 1, 2 (2016), 158–166.

[131] Nan Li, Dave Oyler, Mengxuan Zhang, Yildiray Yildiz, Anouck Girard, and Ilya Kolmanovsky. 2016. Hierarchical reasoning game theory based approach for evaluation and testing of autonomous vehicle control systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, Las Vegas, NV, USA, 727–733.

[132] Nan Li, Dave W Oyler, Mengxuan Zhang, Yildiray Yildiz, Ilya Kolmanovsky, and Anouck R Girard. 2017. Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems. *IEEE Transactions on control systems technology* 26, 5 (2017), 1782–1797.

[133] Raimar Lill and Francesca Saglietti. 2014. Testing the cooperation of autonomous robotic agents. In *2014 9th International Conference on Software Engineering and Applications (ICSOFT-EA)*. IEEE, Vienna, Austria, 287–296.

[134] Mikael Lindvall, Adam Porter, Gudjon Magnusson, and Christoph Schulze. 2017. Metamorphic model-based testing of autonomous systems. In *2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)*. IEEE, Buenos Aires, Argentina, 35–41.

[135] Alessio Lomuscio and Jakub Michaliszyn. 2015. Verifying Multi-Agent Systems by Model Checking Three-valued Abstractions.. In *AAMAS*, Vol. 15. ACM, Istanbul, 189–198.

[136] Alessio Lomuscio and Edoardo Pirovano. 2019. A Counter Abstraction Technique for the Verification of Probabilistic Swarm Systems.. In *AAMAS*. ACM, Montreal, Canada, 161–169.

[137] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. 2017. MCMAS: an open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer* 19, 1 (2017), 9–30.

[138] Yu Lu, Hanlin Niu, Al Savvaris, and Antonios Tsourdos. 2016. Verifying collision avoidance behaviours for unmanned surface vehicles using probabilistic model checking. *IFAC-PapersOnLine* 49, 23 (2016), 127–132.

[139] Matt Luckcuck, Marie Farrell, Louise A. Dennis, Clare Dixon, and Michael Fisher. 2019. Formal Specification and Verification of Autonomous Robotic Systems: A Survey. *ACM Comput. Surv.* 52, 5 (2019), 100:1–100:41. https://doi.org/10.1145/3342355

[140] Israel Lugo-Cárdenas, Gerardo Flores, and Rogelio Lozano. 2014. The MAV3DSim: A simulation platform for research, education and validation of UAV controllers. *IFAC Proceedings Volumes* 47, 3 (2014), 713–717.

[141] Chenxia Luo, Rui Wang, Yu Jiang, Kang Yang, Yong Guan, Xiaojuan Li, and Zhiping Shi. 2018. Runtime verification of robots collision avoidance case study. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 1. IEEE, Tokyo, Japan, 204–212.

[142] Damian M Lyons, Ronald C Arkin, Shu Jiang, Dagan Harrington, Feng Tang, and Peng Tang. 2015. Probabilistic verification of multi-robot missions in uncertain environments. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, Vietri sul Mare, Italy, 56–63.

[143] Damian M Lyons, Ronald C Arkin, Shu Jiang, Matt O'Brien, Feng Tang, and Peng Tang. 2017. Performance verification for robot missions in uncertain environments. *Robotics and Autonomous Systems* 98 (2017), 89–104.

[144] István Majzik, Oszkár Semeráth, Csaba Hajdu, Kristóf Marussy, Zoltán Szatmári, Zoltán Micskei, András Vörös, Aren A Babikian, and Dániel Varró. 2019. Towards system-level testing with coverage guarantees for autonomous vehicles. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, Munich, Germany, 89–94.

[145] Oded Maler and Dejan Nickovic. 2004. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 152–166.

[146] Anthony Mallet, Cédric Pasteur, Matthieu Herrb, Séverin Lemaignan, and Félix Ingrand. 2010. GenoM3: Building middleware-independent robotic components. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, Anchorage, Alaska, 4627–4632.

[147] Michel Mamrot, Stefan Marchlewitz, Jan-Peter Nicklas, Petra Winzer, Thomas Tetzlaff, Philipp Kemper, and Ulf Witkowski. 2015. Model-based Test and Validation Support for Autonomous Mechatronic Systems. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, Hong Kong, 701–706.

[148] Md Abdullah Al Mamun, Christian Berger, and Jorgen Hansson. 2013. MDE-based sensor management and verification for a self-driving miniature vehicle. In *Proceedings of the 2013 ACM workshop on Domain-specific modeling*. ACM, Indianapolis, Indiana, USA, 1–6.

[149] Musa Morena Marcusso Manhães, Sebastian A Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. 2016. UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE*. IEEE, Monterey, California, USA, 1–8.

[150] Niloofar Mansoor, Jonathan A Saddler, Bruno Silva, Hamid Bagheri, Myra B Cohen, and Shane Farritor. 2018. Modeling and testing a family of surgical robots: an experience report. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, Lake Buena Vista, FL, USA, 785–790.

[151] Casper Sloth Mariager, Daniel Kjaer Bonde Fischer, Jakob Kristiansen, and Matthias Rehm. 2019. Co-Designing and Field-Testing Adaptable Robots for Triggering Positive Social Interactions for Adolescents with Cerebral Palsy. In *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, New Delhi, India, 1–6.

[152] MATLAB. 2010. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts.

[153] John Alexander McDermid, Yan Jia, and Ibrahim Habli. 2019. Towards a framework for safety assurance of autonomous systems. In *Artificial Intelligence Safety 2019*. CEUR Workshop Proceedings, Honolulu, Hawaii, 1–7.

[154] Steve McGuire, P Michael Furlong, Terry Fong, Christoffer Heckman, Daniel Szafir, Simon J Julier, and Nisar Ahmed. 2019. Everybody Needs Somebody Sometimes: Validation of Adaptive Recovery in Robotic Space Operations. *IEEE Robotics and Automation Letters* 4, 2 (2019), 1216–1223.

[155] David R. McLver. 2021. Hypothesis. https://github.com/HypothesisWorks/hypothesis. [Online].

[156] Christopher Medrano-Berumen and Mustafaff Ilhan Akbaş. 2019. Abstract simulation scenario generation for autonomous vehicle verification. In *2019 SoutheastCon*. IEEE, Huntsville, Alabama, 1–6.

[157] Alvaro Miyazawa, Pedro Ribeiro, Wei Li, ALC Cavalcanti, Jon Timmis, and JCP Woodcock. 2016. RoboChart: a state-machine notation for modelling and verification of mobile and autonomous robots.

[158] Maurizio Mongelli, Marco Muselli, Andrea Scorzoni, and Enrico Ferrari. 2019. Accellerating prism validation of vehicle platooning through machine learning. In *2019 4th International Conference on System Reliability and Safety (ICSRS)*. IEEE, Rome, Italy, 452–456.

[159] Shahabuddin Muhammad, Nazeeruddin Mohammad, Abul Bashar, and Majid Ali Khan. 2019. Designing human assisted wireless sensor and robot networks using probabilistic model checking. *Journal of Intelligent & Robotic Systems* 94, 3-4 (2019), 687–709.

[160] Galen E Mullins, Paul G Stankiewicz, R Chad Hawthorne, and Satyandra K Gupta. 2018. Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles. *Journal of Systems and Software* 137 (2018), 197–215.

[161] Adnan Munawar and Gregory S Fischer. 2019. An asynchronous multi-body simulation framework for real-time dynamics, haptics and learning with application to surgical robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Venetian Macao, Macau.

[162] Florian Mutter, Stefanie Gareis, Bernhard Schätz, Andreas Bayha, Franziska Grüneis, Michael Kanis, and Dagmar Koss. 2011. Model-driven in-the-loop validation: Simulation-based testing of UAV software using virtual environments. In *2011 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems*. IEEE, Tangier, Morocco, 269–275.

[163] Frederik Naujoks, Sebastian Hergeth, Katharina Wiedemann, Nadja Schömig, and Andreas Keinath. 2018. Use cases for assessing, testing, and validating the human machine interface of automated driving systems. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 62. SAGE Publications Sage CA: Los Angeles, CA, Philadelphia, PA, USA, 1873–1877.

[164] Cu D Nguyen, Simon Miles, Anna Perini, Paolo Tonella, Mark Harman, and Michael Luck. 2012. Evolutionary testing of autonomous software agents. *Autonomous Agents and Multi-Agent Systems* 25, 2 (2012), 260–283.

[165] Royal Academy of Engineering. 2015. Innovation in autonomous systems: Summary of an event held on Monday 22 June 2015 at the Royal Academy of Engineering.

[166] Matthew O'Kelly, Houssam Abbas, Sicun Gao, Shin'ichi Shiraishi, Shinpei Kato, and Rahul Mangharam. 2016. APEX: Autonomous vehicle plan verification and execution. In *SAE 2016 World Congress and Exhibition*. SAE International, Detroit, MI, USA.

[167] Matthew O'Kelly, Aman Sinha, Hongseok Namkoong, Russ Tedrake, and John C Duchi. 2018. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *Advances in Neural Information Processing Systems*. NeurIPS, Montreal, Canada, 9827–9838.

[168] Stephan Opfer, Stefan Niemczyk, and Kurt Geihs. 2016. Multi-agent plan verification with answer set programming. In *Proceedings of the 3rd Workshop on Model-Driven Robot Software Engineering*. ACM, Leipzig, Germany, 32–39.

[169] Matthew O'Brien, Ronald C Arkin, Dagan Harrington, Damian Lyons, and Shu Jiang. 2014. Automatic verification of autonomous robot missions. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, Bergamo, Italy, 462–473.

[170] Jisun Park, Mingyun Wen, Yunsick Sung, and Kyungeun Cho. 2019. Multiple event-based simulation scenario generation approach for autonomous vehicle smart sensors and devices. *Sensors* 19, 20 (2019), 4456.

[171] Corina S Pasareanu, Divya Gopinath, and Huafeng Yu. 2018. Compositional Verification for Autonomous Systems with Deep Learning Components. *arXiv preprint arXiv:1810.08303* 1 (2018).

[172] Shashank Pathak, Giorgio Metta, and Armando Tacchella. 2014. Is verification a requisite for safe adaptive robots?. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, San Diego, USA, 3399–3402.

[173] José LF Pereira and Rosaldo JF Rossetti. 2012. An integrated architecture for autonomous vehicles simulation. In *Proceedings of the 27th annual ACM symposium on applied computing*. ACM, Trento, Italy, 286–292.

[174] Mauro Pezze and Michal Young. 2007.

[175] Javier Poncela and MC Aguayo-Torres. 2013. A Framework for Testing of Wireless Underwater Robots. *Wireless personal communications* 70, 3 (2013), 1171–1181.

[176] David Porfirio, Allison Sauppé, Aws Albarghouthi, and Bilge Mutlu. 2018. Authoring and verifying human-robot interactions. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, Berlin, Germany, 75–86.

[177] Martin Proetzsch, Fabian Zimmermann, Robert Eschbach, Johannes Kloos, and Karsten Berns. 2010. A systematic testing approach for autonomous mobile robots using domain-specific languages. In *Annual Conference on Artificial Intelligence*. Springer, Atlanta, Georgia, USA, 317–324.

[178] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, Vol. 3. Kobe, Japan, Kobe, Japan, 5.

[179] Nijat Rajabli, Francesco Flammini, Roberto Nardone, and Valeria Vittorini. 2021. Software Verification and Validation of Safe Autonomous Cars: A Systematic Literature Review. *IEEE Access* 9 (2021), 4797–4819. https://doi.org/10.1109/ACCESS.2020.3048047

[180] Arvind Ramanathan, Laura L Pullum, Faraz Hussain, Dwaipayan Chakrabarty, and Sumit Kumar Jha. 2016. Integrating symbolic and statistical methods for testing intelligent systems: Applications to machine learning and computer vision. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Dresden, Germany, 786–791.

[181] Q. Rao and J. Frtunikj. 2018. Deep Learning for Self-Driving Cars: Chances and Challenges. In *2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)*. IEEE, Gothenburg, Sweden, 35–38.

[182] Signe A. Redfield and Mae L. Seto. 2017. Verification Challenges for Autonomous Systems. In *Autonomy and Artificial Intelligence: A Threat or Savior?*, William F. Lawless, Ranjeev Mittu, Donald Sofge, and Stephen Russell (Eds.). Springer International Publishing, USA, 103–127. https://doi.org/10.1007/978-3-319-59719-5_5

[183] Pedro Ribeiro, Alvaro Miyazawa, Wei Li, Ana Cavalcanti, and Jon Timmis. 2017. Modelling and verification of timed robotic controllers. In *International Conference on Integrated Formal Methods*. Springer, Turin, Italy, 18–33.

[184] Sergio Rico, Emelie Engström, and Martin Höst. 2019. A Taxonomy for Improving Industry-Academia Communication in IoT Vulnerability Management. In *Proceedings of the 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, Kallithea-Chalkidiki, Greece, 38–45.

[185] Eric Rohmer, Surya PN Singh, and Marc Freese. 2013. V-REP: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Tokyo, Japan, 1321–1326.

[186] Martijn Rooker, Pablo Horstrand, Aythami Salvador Rodriguez, Sebastian Lopez, Roberto Sarmiento, Jose Lopez, Ray Alejandro Lattarulo, Joshue Manuel Perez Rastelli, Zora Slavik, David Pereira, et al. 2018. Towards improved validation of autonomous systems for smart farming. In *Smart Farming Workshop*. ISEP, Porto, Portugal.

[187] Gregg Rothermel, Roland H. Untch, Chengyun Chu, and Mary Jean Harrold. 1999. Test Case Prioritization: An Empirical Study. In *Proc. of the 1999 International Conference on Software Maintenance, ICSM 1999*. IEEE Computer Society, Oxford, England, 179–188. https://doi.org/10.1109/ICSM.1999.792604

[188] Sasha Rubin. 2015. Parameterised verification of autonomous mobile-agents in static but unknown environments. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. ACM, Istanbul, Turkey, 199–208.

[189] Peter AM Ruijten, Antal Haans, Jaap Ham, and Cees JH Midden. 2019. Perceived human-likeness of social robots: testing the Rasch model as a method for measuring anthropomorphism. *International Journal of Social Robotics* 11, 3 (2019), 477–494.

[190] Rim Saddem, Olivier Naud, Karen Godary Dejean, and Didier Crestani. 2017. Decomposing the model-checking of mobile robotics actions on a grid. *IFAC-PapersOnLine* 50, 1 (2017), 11156–11162.

[191] Francesca Saglietti and Matthias Meitner. 2016. Model-driven structural and statistical testing of robot cooperation and reconfiguration. In *Proceedings of the 3rd Workshop on Model-Driven Robot Software Engineering*. ACM, Leipzig, Germany, 17–23.

[192] Francesca Saglietti, Stefan Winzinger, and Raimar Lill. 2014. Reconfiguration testing for cooperating autonomous agents. In *International Conference on Computer Safety, Reliability, and Security*. Springer, Delft, The Netherlands, 144–155.

[193] André Santos, Alcino Cunha, and Nuno Macedo. 2018. Property-based testing for the robot operating system. In *Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation*. ACM, Lake Buena Vista, FL, USA, 56–62.

[194] Ichiro Satoh. 2018. An approach for testing software on networked transport robots. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, Imperia, Italy, 1–4.

[195] Ichiro Satoh. 2019. Developing and Testing Networked Software for Moving Robots.. In *ENASE*. Springer, Heraklion, Crete, Greece, 315–321.

[196] Hans-Peter Schöner. 2018. Simulation in development and testing of autonomous vehicles. In *18 Internationales Stuttgarter Symposium*. Springer, Germany, 1083–1095.

[197] David Seiferth and Matthias Heller. 2017. Testing and performance enhancement of a model-based designed ground controller for a diamond-shaped unmanned air vehicle (UAV). In *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, Hawaii, USA, 1988–1994.

[198] Yuvaraj Selvaraj, Wolfgang Ahrendt, and Martin Fabian. 2019. Verification of Decision Making Software in an Autonomous Vehicle: An Industrial Case Study. In *International Workshop on Formal Methods for Industrial Critical Systems*. Springer, Amsterdam, The Netherlands, 143–159.

[199] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*. Springer, Zurich, Switzerland, 621–635.

[200] Weijing Shi, Mohamed Baker Alawieh, Xin Li, Huafeng Yu, Nikos Arechiga, and Nobuyuki Tomatsu. 2016. Efficient statistical validation of machine learning systems for autonomous driving. In *Proceedings of the 35th International Conference on Computer-Aided Design*. ACM, Austin, Texas, USA, 1–8.

[201] Christoph Sippl, Florian Bock, David Wittmann, Harald Altinger, and Reinhard German. 2016. From simulation data to test cases for fully automated driving and ADAS. In *IFIP International Conference on Testing Software and Systems*. Springer, Graz, Austria, 191–206.

[202] Gopinadh Sirigineedi, Antonios Tsourdos, Brian A White, and Rafał Żbikowski. 2011. Kripke modelling and verification of temporal specifications of a multiple UAV system. *Annals of Mathematics and Artificial Intelligence* 63, 1 (2011), 31–52.

[203] Michał Siwek, Leszek Baranowski, Jarosław Panasiuk, and Wojciech Kaczmarek. 2019. Modeling and simulation of movement of dispersed group of mobile robots using Simscape multibody software. In *AIP Conference Proceedings*, Vol. 2078. AIP Publishing LLC, Heraklion, Crete, Greece, 020045.

[204] Marc Spislaender and Francesca Saglietti. 2018. Evidence-Based Verification of Safety Properties Concerning the Cooperation of Autonomous Agents. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, Prague, Czech Republic, 81–88.

[205] Tomoo Sumida, Hiroyuki Suzuki, Sho Sei Shun, Kazuhito Omaki, Takaaki Goto, and Kensei Tsuchida. 2017. FDR verification of a system involving a robot climbing stairs. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*. IEEE, Wuhan, China, 875–878.

[206] Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. 2019. Formal verification of neural network controlled autonomous systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. ACM, Montreal, Quebec, Canada, 147–156.

[207] Zsolt Szalay, Mátyás Szalai, Bálint Tóth, Tamás Tettamanti, and Viktor Tihanyi. 2019. Proof of concept for Scenario-in-the-Loop (SciL) testing for autonomous vehicle technology. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, Graz, Austria, 1–5.

[208] Zaid Tahir and Rob Alexander. 2020. Coverage based testing for V&V and Safety Assurance of Self-driving Autonomous Vehicles: A Systematic Literature Review. In *IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, Oxford, UK, 23–30. https://doi.org/10.1109/AITEST49225.2020.00011

[209] Jianbo Tao, Yihao Li, Franz Wotawa, Hermann Felbinger, and Mihai Nica. 2019. On the industrial application of combinatorial testing for autonomous driving functions. In *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, Xi'an, Chin, 234–240.

[210] Mugur Tatar. 2015. Enhancing ADAS test and validation with automated search for critical situations. In *Driving Simulation Conference (DSC)*. DSC Europe, Paris, France.

[211] Unity Technologies. 2021. Unity. https://unity.com. [Online].

[212] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*. ACM, Gothenburg, Sweden, 303–314.

[213] Thomas Tosik, Jasper Schwinghammer, Mandy Jane Feldvoß, John Paul Jonte, Arne Brech, and Erik Maehle. 2016. MARS: A simulation environment for marine swarm robotics and environmental monitoring. In *OCEANS 2016*. IEEE, Shanghai, China, 1–6.

[214] Tarik Tosun, Gangyuan Jing, Hadas Kress-Gazit, and Mark Yim. 2018. Computer-aided compositional design and verification for modular robots. *Robotics Research* 1 (2018), 237–252.

[215] Garazi Juez Uriagereka, Estibaliz Amparan, Cristina Martinez Martinez, Jabier Martinez, Aurelien Ibanez, Matteo Morelli, Ansgar Radermacher, and Huascar Espinoza. 2019. Design-Time Safety Assessment of Robotic Systems Using Fault Injection Simulation in a Model-Driven Approach. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, Munich, Germany, 577–586.

[216] Vandi Verma and Chris Leger. 2019. SSim: NASA Mars Rover Robotics Flight Software Simulation. In *2019 IEEE Aerospace Conference*. IEEE, Big Sky, MT, USA, 1–11.

[217] Federico Vicentini, Mehrnoosh Askarpour, Matteo G Rossi, and Dino Mandrioli. 2019. Safety assessment of collaborative robotics through automated formal verification. *IEEE Transactions on Robotics* 36, 1 (2019), 42–61.

[218] Harsha Jakkanahalli Vishnukumar, Björn Butting, Christian Müller, and Eric Sax. 2017. Machine learning and deep neural network—Artificial intelligence core for lab and real-world test and validation for ADAS and autonomous vehicles: AI for efficient and quality test and validation. In *2017 Intelligent Systems Conference (IntelliSys)*. IEEE, London, UK, 714–721.

[219] Dennis Walter, Holger Täubig, and Christoph Lüth. 2010. Experiences in applying formal verification in robotics. In *International Conference on Computer Safety, Reliability, and Security*. Springer, Vienna, Austria, 347–360.

[220] Kai Wang and JC Cheng. 2019. Integrating Hardware-In-the-Loop Simulation and BIM for Planning UAV-Based As-Built MEP Inspection with Deep Learning Techniques. In *Proceedings of the 36th International Symposium on Automation and Robotics in Construction*. IAARC, Alberta, Canada, 310–316.

[221] Rui Wang, Yingxia Wei, Houbing Song, Yu Jiang, Yong Guan, Xiaoyu Song, and Xiaojuan Li. 2018. From offline towards real-time verification for robot systems. *IEEE Transactions on Industrial Informatics* 14, 4 (2018), 1712–1721.

[222] Matt Webster, Clare Dixon, Michael Fisher, Maha Salem, Joe Saunders, Kheng Lee Koay, Kerstin Dautenhahn, and Joan Saez-Pons. 2015. Toward reliable autonomous robotic assistants through formal verification: A case study. *IEEE Transactions on Human-Machine Systems* 46, 2 (2015), 186–196.

[223] Matt Webster, Maha Salem, Clare Dixon, Michael Fisher, and Kerstin Dautenhahn. 2014. Formal verification of an autonomous personal robotic assistant. *AIAA* 1 (2014).

[224] Matt Webster, David Western, Dejanira Araiza-Illan, Clare Dixon, Kerstin Eder, Michael Fisher, and Anthony G Pipe. 2019. A corroborative approach to verification and validation of human–robot teams. *The International Journal of Robotics Research* 39, 1 (2019), 73–99.

[225] Dennis Leroy Wigand, Pouya Mohammadi, Enrico Mingo Hoffman, Nikos G Tsagarakis, Jochen J Steil, and Sebastian Wrede. 2018. An open-source architecture for simulation, execution and analysis of real-time robotics systems. In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, Brisbane, Australia, 93–100.

[226] Tichakorn Wongpiromsarn, Sayan Mitra, Andrew Lamperski, and Richard M Murray. 2012. Verification of periodically controlled hybrid systems: Application to an autonomous vehicle. *ACM Transactions on Embedded Computing Systems (TECS)* 11, S2 (2012), 1–24.

[227] Bingqing Xu, Qin Li, Tong Guo, Yi Ao, and Dehui Du. 2019. A quantitative safety verification approach for the decision-making process of autonomous driving. In *2019 International Symposium on Theoretical Aspects of Software Engineering (TASE)*. IEEE, Guilin, China, 128–135.

[228] Bingqing Xu, Qin Li, Tong Guo, and Dehui Du. 2019. A Scenario-Based Approach for Formal Modelling and Verification of Safety Properties in Automated Driving. *IEEE Access* 7 (2019), 140566–140587.

[229] Wing Lok Yeung. 2011. Behavioral modeling and verification of multi-agent systems for manufacturing control. *Expert Systems with applications* 38, 11 (2011), 13555–13562.

[230] Levent Yilmaz. 2017. Verification and validation of ethical decision-making in autonomous systems. In *Proceedings of the Symposium on Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems*. Springer, Virginia Beach, Virginia, USA, 1–12.

[231] Fu Yujian and Drabo Mebougna. 2014. formal Modeling and verification of dynamic reconfiguration of autonomous robotics systems. In *Proceedings of the International Conference on Embedded Systems and Applications (ESA)*, Vol. 2. Csrea, Las Vegas, Nevada, USA, 14.

[232] Sunkil Yun, Takaaki Teshima, and Hidekazu Nishimura. 2019. Human–Machine Interface Design and Verification for an Automated Driving System Using System Model and Driving Simulator. *IEEE Consumer Electronics Magazine* 8, 5 (2019), 92–98.

[233] Chi Zhang, Yuehu Liu, Danchen Zhao, and Yuanqi Su. 2014. RoadView: A traffic scene simulator for autonomous vehicle simulation testing. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, Qingdao, China, 1160–1165.

[234] Xiaoyang Zhang, Hongpeng Wang, Jingtai Liu, and Haifeng Li. 2019. CyberEarth: a Virtual Simulation Platform for Robotics and Cyber-Physical Systems. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, Dali, Yunnan Province, China, 858–863.

[235] Xingyu Zhao, Matt Osborne, Jenny Lantair, Valentin Robu, David Flynn, Xiaowei Huang, Michael Fisher, Fabio Papacchini, and Angelo Ferrando. 2019. Towards Integrating Formal Verification of Autonomous Robots with Battery Prognostics and Health Management. In *International Conference on Software Engineering and Formal Methods*. Springer, Oslo, Norway, 105–124.

[236] Xingyu Zhao, Valentin Robu, David Flynn, Fateme Dinmohammadi, Michael Fisher, and Matt Webster. 2019. Probabilistic model checking of robots deployed in extreme environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. AAAI, Honolulu, Hawaii, USA, 8066–8074.

[237] Xingyu Zhao, Valentin Robu, David Flynn, Kizito Salako, and Lorenzo Strigini. 2019. Assessing the safety and reliability of autonomous vehicles from road testing. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, Berlin, Germany, 13–23.

[238] Jinwei Zhou, Roman Schmied, Alexander Sandalek, Helmut Kokal, and Luigi del Re. 2016. A framework for virtual testing of adas. *SAE International Journal of Passenger Cars-Electronic and Electrical Systems* 9, 2016-01-0049 (2016), 66–73.

[239] Marc René Zofka, Marc Essinger, Tobias Fleck, Ralf Kohlhaas, and J Marius Zöllner. 2018. The sleepwalker framework: Verification and validation of autonomous vehicles by mixed reality lidar stimulation. In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, Brisbane, Australia, 151–157.