A Study on Challenges of Testing Robotic Systems

Afsoon Afzal, Claire Le Goues, Michael Hilton and Christopher Steven Timperley
Carnegie Mellon University, Pittsburgh, PA
Email: afsoona@cs.cmu.edu, clegoues@cs.cmu.edu, mhilton@cmu.edu, ctimperley@cmu.edu

***Abstract*—Robotic systems are increasingly a part of everyday life. Characteristics of robotic systems such as interaction with the physical world, and integration of hardware and software components, differentiate robotic systems from conventional software systems. Although numerous studies have investigated the challenges of software testing in practice, no such study has focused on testing of robotic systems. In this paper, we conduct a qualitative study to better understand the testing practices used by the robotics community, and identify the challenges faced by practitioners when testing their systems. We identify a total of 12 testing practices and 9 testing challenges from our participants' responses. We group these challenges into 3 major themes: *Real-world complexities*, *Community and standards*, and *Component integration*. We believe that further research on addressing challenges described with these three major themes can result in higher adoption of robotics testing practices, more testing automation, and higher-quality robotic systems.**

***Index Terms*—robotics testing; testing challenges; qualitative study;**

I. Introduction

Robots are systems that sense, process, and physically react to information from the real world.[1] In addition to being heavily used in manufacturing and industrial settings, robotic systems are now appearing in many important and safety-critical domains such as health care, education, and transportation. Increased interaction between these systems and the public raises the risk of catastrophic failure. For example, a fatal incident occurred in March 2018 in Tempe, Arizona when a self-driving car struck a pedestrian [1].

Because of the associated dangers and cost of failures in robotic systems, it is crucial that developers test these systems extensively before deployment. However, robotic systems differ from conventional software in several important dimensions [2]–[7]: (1) Robots are comprised of (unreliable and non-determinstic) hardware, software, and physical components [2], [3], [7]. (2) Robots interact with the physical world via inherently noisy sensors and actuators, and are sensitive to timing differences [7]. (3) Robots operate within the practically boundless state space of reality, making emergent behaviors (i.e., corner cases) difficult to predict [2]. (4) For robotic systems, the notion of correctness is often inexact and difficult to precisely specify [6]. These characteristics introduce unique challenges for testing, such as the need to either heavily abstract aspects of physical reality or conduct extensive real-world field testing.

Many studies have investigated testing practices in software development generally [8]–[12]. Several prior studies on testing on Cyber-Physical Systems (CPS) [4]–[6], of which robotic systems may be considered a subcategory [13], do include certain robotic systems in the larger CPS context (which includes non-robotics systems like networking systems or power grids). However, none of these studies focuses specifically on robotics, which are subject to system constraints that do not apply to CPS broadly (such as a need for autonomy, route planning, and mobility). Indeed, we are unaware of any prior published work that has examined testing practices and challenges in the field of robotics.

Overall, although testing is essential to software development [14], the challenges unique to the domain mean that testing for *robotics specifically* may pose particular and understudied challenges in both research and practice. Although numerous studies have proposed frameworks and algorithms for testing robotic systems [15]–[20], little attention has been paid to investigating the challenges of testing in robotics in practice. This has resulted in a gap in the research community's ability to engage with the challenges faced when testing robotics.

In this paper, we address this gap by studying testing practices and challenges in robotics. We conduct a series of qualitative interviews with 12 robotics practitioners from 11 robotics companies and institutions. Specifically, we investigate the testing practices that are being used in the field of robotics, and the challenges faced by roboticists when testing their systems. We answer the following research questions:

- **RQ1:** *What testing practices are currently being used by roboticists?*
- **RQ2:** *What are the costs and barriers to designing and writing tests for robotic systems?*
- **RQ3:** *What are the costs and barriers to running and automating tests in robotic systems?*

Having a better understanding of the current state of testing in robotics, as well as the problems and concerns of the robotics community regarding testing of robotic systems, will guide researchers and practitioners to provide and apply solutions that can ultimately result in higher-quality robotic systems. Overall, we make the following contributions:

- We conduct in-depth interviews with 12 robotics practitioners from 11 different robotics companies and institutions, in which we ask about their testing practices and challenges.
- We identify 12 testing practices used by robotics developers and test engineers, 4 challenges that they commonly face when designing testing platforms and writing tests, and 5 challenges that they face when running and au-

[1]Max Plank Institute: https://www.cis.mpg.de/robotics/

tomating tests.
- We identify and discuss three general themes of challenges in robotics testing that require attention from the research community, and provide our overview and suggestions on those challenges may be tackled.

II. Related work

Studies on testing in practice benefit both testing professionals, who can learn from the experiences of others, and researchers, who observe the strengths and weaknesses of these practices and can contribute to further progress. Qualitative and quantitative studies can identify gaps in existing research, and encourage the research community to address those gaps by directing research efforts towards the identified problems.

A number of studies have investigated software testing practices broadly, and the associated challenges [8]–[12]. Runeson [8] conducted a large-scale survey on unit testing with 19 software companies, and identified unit test definitions, strengths, and problems. Causevic et al. [9] qualitatively and quantitatively study practices and preferences on contemporary aspects of software testing. However, these studies did not focus on testing challenges in robotics.

Although the results of many studies on CPSs also apply to robotics, robotics is a subcategory of CPS [13]. For example, wireless networks, and smart buildings are cyberphysical systems that are not robotics systems. Robotics-specific challenges and constraints are thus not addressed by this type of work.

Zheng et al. [21] overview verification and validation in cyberphysical systems. They find significant research gaps in addressing verification and validation of CPS, and that these gaps potentially stand in the way of the construction of robust, reliable and resilient mission-critical CPS. They also find that developers lack trust in simulators; one of the main research challenges they identify is integrated simulation.

Seshia et al. [5]. introduce a combination of characteristics that define the challenges unique to the design automation of CPSs. Marijan et al. [6] speculate over a range of challenges involving testing of machine learning based systems. However, this work does not conduct qualitative or quantitative studies to confirm their hypotheses. Duan et al. [4] extract 27 challenges for verification of CPSs by performing a large-scale meta-analysis on papers published between 2006 to 2018. Even though they identify verification challenges that the research community is eager to solve, they do not provide information on which practices are used by practitioners, and to what extent the research has been deployed in practice.

Alami et al. [22] study the quality assurance practices of the Robot Operating System (ROS)[2] community by using qualitative methods such as interviews, virtual ethnography, and community reach-outs. They learn that implementation and execution of QA practices in the ROS community are influenced by social and cultural factors and are constrained by sustainability and complexity. However, their results only apply to a specific robotics framework and cannot be generalized to non-ROS systems.

Luckcuck et al. [23] systematically survey the state of the art in formal specification and verification for autonomous robotics, and identified the challenges of formally specifying and verifying (autonomous) robotic systems. Their study focuses on formal specification as a method of quality assurance and does not provide information regarding other testing practices within the wider field of robotics.

III. Methodology

Our goal in this study is to gain an in-depth understanding of existing testing practices and challenges within the robotics industry. Inspired by established guidelines and previous work [24]–[28], we conducted a series of semi-structured interviews with robotics practitioners from a diverse set of companies. We chose to perform interviews because they are useful instruments for getting the story behind a participant's experiences, acquiring in-depth information on a topic, and soliciting unexpected types of information [29], [30]. We developed our interview script by performing a series of iterative pilots.

We recruited our participants through a variety of means. Our goal was to select participants from a broad range of positions and to sample across a diversity of industries, company size, and experience. We recruited our first three participants using convenience sampling. We recruited the rest of our participants using snowball sampling and targeted messages to developers that we found on LinkedIn and Twitter who had the phrase "robotics engineer" in their profile.

Overall, we interviewed 12 robotics practitioners with a variety of backgrounds and experiences. This practitioners represent 11 robotics companies and institutions ranging from small startups to large multi-national companies. A summary of the relevant details of the participants of our study is presented in Table I. After performing the interviews, we determined that while P5 and P7 work at a company that is heavily involved in robotics, both of the participants are focused on non-robotics-related software development, and so we removed them from our sample moving forward.

Interviews and coding: We conducted semi-structured interviews that lasted between 30 to 60 minutes over the phone, using video chat, or conducted face-to-face. We prepared an interview script with detailed questions to provide insight into our research questions, which we provide as a supplement to this paper.[3] A subset of questions on the interview script are presented in Table II. However, we only used the script to guide the interviews. We adjusted interview questions based on the experience of the participant, to gain a deeper understanding of their testing practices and challenges. We took notes on interviewee responses, and recorded the interviews with participant consent to validate our notes. We then used a grounded, iterative approach to code our notes. We first labeled responses based on their relevance to our

[2]https://ros.org

[3]https://doi.org/10.5281/zenodo.3625199

TABLE I
INTERVIEW PARTICIPANTS, THEIR EXPERIENCE WITH ROBOTICS, THEIR ROLE IN THE COMPANY OR INSTITUTION, TYPE AND SECTOR OF THEIR COMPANY OR INSTITUTION, AND WHETHER THEIR TESTING PROCESS INCLUDES A DEDICATED QUALITY ASSURANCE TEAM.

	Participant			Company/Institute		
ID	Background	Years in robotics	Role	Type	Sector	QA team?
P1	Software Engineering	6	Developer	Startup	Mobile Services	✘
P2	Electrical Engineering	> 10	Principal Engineer	Academia	Research & Development	✘
P3	Embedded Software Engineering	2	Developer	Multinational Company	Autonomous Vehicle	✔
P4	Mechanical & Robotics Engineering	5	Developer	Research Lab	Agriculture	✘
P5	Software Engineering	> 10	Test Engineer	Multinational Company	Industrial Automation	✔
P6	Math & Physics	> 10	Project Manager	Startup	Education	✘
P7	Experimental Physics	7	Test Engineer	Multinational Company	Industrial Automation	✔
P8	Mechanical Engineering & Math	5	Manager/Engineer	Startup	Cleaning	✘
P9	Computer Science	4	Engineer	Robotics contractor	Research & Development	✔
P10	Computer Science	> 10	Research Engineer	Academia	Industrial Automation	✘
P11	Computer Science & Math	< 1	Software Engineer	Multinational Company	Industrial Automation	✔
P12	Robotics Engineering	> 10	CTO	Startup	Mobile Services	✘

TABLE II
SAMPLE QUESTIONS ON THE INTERVIEW SCRIPT.

Practice	• What are all the different types of testing you do? • Can you describe your test running/writing process? • How much of your testing is done for certification? • Which types of tests find the most problems?
Testing Challenges	• What is difficult about writing tests? • Have these difficulties ever made you giving up on writing the tests at all? • Is there any part of writing tests that is not difficult? • What types of tests do you have the most difficulty running? • In your experience, is there anything that helps with making it easier to run tests? • For your tests that are not fully automated, why are they not? • What tools/frameworks/techniques do you use to simplify running tests? • Do you use simulation?
General	• What do you think is the most important bottleneck in the way of testing in robotics? • How do you think the difficulties of testing in robotics differ from your other experiences in other software development domains?

research questions. Then, we iteratively coded the notes based on common themes, discussed the codes and redefined them. We present these themes in Section IV.

Validation: To validate the results of our study and conclusions, we sent a full draft of Sections IV- V to our participants. We asked participants to inform us of their level of agreement with our conclusions and to provide their thoughts on our results. In total, six of the participants responded to our request. Four responded in total agreement with the results. The other two participants that responded provided specific feedback on our interpretation of their responses, and we incorporated their feedback into the final version of this paper.

IV. RESULTS

In this section, we discuss the results in response to our research questions. We identify 12 testing practices in use by roboticists, and 9 challenges for robotics testing.

A. RQ1: Testing practices in robotics

To determine the testing practices that are used in the robotics industry, we asked our participants to describe their own testing practices. In total, our participants reported 12 different testing practices, summarized in Table III. Given the explorative nature of our study, we do not make any claim about the popularity of the reported practices; rather, we aim to identify the variety of testing methods that are used in robotics. Below, we discuss a selection of identified practices, bolded in Table III, in more detail.

(T1) Field testing: A full system test that happens in an environment that is similar or identical to the intended deployment environment can reveal many problems, as a robot is exposed to real-world scenarios and input. According to our participants, field testing is a common practice in robotics. Several of our participants mentioned that they conduct field testing frequently during both development and testing of a robot. For example, P9 and P12 both mentioned one or two-week long field testing events that take place after each development cycle. P2 stated that they conduct field testing once or twice a week.

(T2) Logging and playback: Logs that are collected during the operation of a system contain important information about system execution for testing, debugging, and development of algorithms. Five of our participants reported that they collect detailed logs during the operation of their systems. Some use recorded logs for debugging and monitoring. For example, P9 provided an example where their robot logs whenever it resets, and they automatically process the logs to ensure that no unexpected, silent reset took place during the operation of the robot.

Logs can also be used to playback events and sensor data (a.k.a. record-and-replay) by feeding input collected from previous operations (either in the field or in simulation) to a robot. For example, ROSBAG is a widely-used command-line tool that records and replays messages for robots built using ROS.[4] P10 and P12, for example, use record-and-replay to

[4]http://wiki.ros.org/rosbag

TABLE III
A SUMMARY OF THE TESTING PRACTICES THAT WERE REPORTED BY PARTICIPANTS. PRACTICES IN BOLD ARE DISCUSSED IN MORE DETAIL IN THE TEXT.

ID	Title	Description	Representative quote
T1	**Field testing**	Full-system testing in a real-world environment that shares similarities with the deployment environment.	"*I'm really a proponent of test often, fail often. We do field testing once or twice a week.*" – P2
T2	**Logging and playback**	The use of logged data, collected in the field, for the purposes of testing, debugging, and development (a.k.a. record and replay).	"*There is an event logger, which will log if something weird happens. When you do playback, if the test engineer thinks things are wrong, they can manually say what went wrong.*" – P3
T3	**Simulation testing**	Tests that are executed in a simulated environment that can be used for both testing and development.	"*We do have tests running in simulation. We use Gazebo. We mostly run planning algorithm tests in there.*" – P12
T4	**Plan-based testing**	The practice of planning an adequate sequence of field tests for validating that the system meets its requirements given a fixed testing budget (e.g., time, hardware, cost).	"*We have oral or written test plan. Test plan is created in the design review.*" – P2
T5	**Compliance testing**	Testing for the purposes of determining whether a system complies with certain standards.	"*Government project prescribes testing. We have several projects that have gone through certifications.*" – P9
T6	Unit testing	Small, automated tests for validating individual code-level software components (i.e., functions).	"*For smaller software modules, we do unit testing and code coverage analysis.*" – P12
T7	Performance testing	Subjecting a system to various workloads to ensure that it meets its functional (e.g., localization accuracy) and non-functional requirements (e.g., timeliness, memory).	"*Whenever we release a new version of the system, we run a system test on it: check that performance on ground truth data isn't degraded.*" – P4
T8	Hardware testing	Testing for the purposes of assessing the quality and integrity of hardware components prior to software integration (e.g., testing sensors and cameras).	"*We manually check every single robot in the factory before it's shipped. We check that firmware is correct, IO ports are functional.*" – P6
T9	Robustness testing	Testing the system under extreme boundary conditions (e.g., a malfunctioning sensor) that are usually artificially injected to determine the safe operating limits of the system.	"*We do an automated test where the middleware spins the system up, then runs endurance tests. The middleware can specify certain commands from system. This makes there be very little risk in the full system.*" – P9
T10	Regression testing	Ensuring that changes to the system (e.g., the addition of a new feature) do not negatively affect existing functionality in an unintended way.	"*We have an automated system that runs the tests nightly. We get some false positives. It's used as regression testing: looking at changes between old and new versions.*" – P4
T11	Continuous integration	The practice of continually and automatically rebuilding the system and executing some portion of its tests (e.g., unit tests) as changes are made.	"*It will build three projects and run tests. Since the builds can take a long time, they push up AWS to build and test.*" – P1
T12	Test maintenance	The practice of refactoring and maintaining tests to eliminate false positives, flakiness and redundancy, and to reflect changes to the requirements of the system.	"*We maintain our tests fairly regularly. On a weekly level.*" – P11

collect test inputs and debug their robots. P2 mentioned using record-and-replay to develop algorithms for their robots:

> *We often do not know why robots are making the choices that they make. By playing back the data in testing we can see why the robot made the choice it did, and then tweak the algorithm to see how it changes the robots behavior.*

(T3) Simulation testing: Using simulated environments (rather than real-world, physical environments) can beneficially reduce the cost of testing and increase the opportunities for test automation [31], [32]. A recent study by Timperley et al. [31] suggests that many real-world robotics bugs could have be reproduced and fixed in simulated environments. However, few participants reported that they used simulation as part of their testing process, even though all participants were aware of the theoretical benefits of simulation. For instance, P12 specifically said:

> *Our best way to test [algorithmic modules that are very dependent on input data] is through simulation, but we don't. We test in the real-world.*

Participants report that simulation is sometimes used as a tool during development, especially for high-level algorithms such as planning. P2 mentioned that they use simulation to create artificial scenarios while developing an algorithm. P9 said that their software team uses simulation to facilitate software development before the hardware platform is available.

Although simulation testing provides additional opportunities for test automation, our participants rarely used simulation for this purpose. Only P3 mentioned using simulation to some extent for automated testing:

> *We have some simulation cluster, so we could setup a script to run a simulation test automatically.*

(T4) Plan-based testing: An outline and objectives for testing can be specified in advance in order to manage and guide testing. Test plans can be created based on different criteria such as formally specified system requirements, and recently added/modified features. P2, P8, and P9 all create a system requirements list, which is used to ensure all components of the system are covered by the tests. For example, P9 said:

> *We have a requirements list we edit with our sponsor. We measure quality of tests against requirements coverage, not code coverage.*

However, in P3's company, developers provide a test plan and failure criteria to test engineers for newly added or modified features.

(T5) Compliance testing: In many fields such as electronics, a certain level of testing is required for the purpose of certification.[5] In robotics, there are a number of standards and certifications in place for sub-fields such as autonomous vehicles [33]. However, in most sub-fields there are no standards or mandatory certifications in place. P12 mentioned that they have not done any testing for certification since they are not required to do so. P10 considers their work as too experimental to require certification. However, government agencies or sponsors may enforce standards to ensure the quality of products (example quote in Table III). Companies may also voluntarily adapt standards to better test their systems. For example, P11 mentioned manually verifying several standards for their robots.

B. RQ2: Challenges of designing and writing tests

We asked our participants to describe the challenges they face when designing and writing tests for their products. We identified four common themes of challenges from their responses, summarized in Table IV. We provide detailed descriptions for each theme below.

(C1) Unpredictable corner cases: Robots are typically expected to operate in many different environments and conditions. In most cases, the robotic state space is infinite, since it interacts with the real world; predicting the exact behavior of the physical environment is not viable [34], [35]. Attempting to account for all possible conditions and scenarios when designing tests is extremely difficult, if not impossible. For example, a plastic bag flying in front of a self-driving car's sensor is a case that may not immediately come to mind when designing tests. However, these unexpected corner cases are often the cause of failures [36].

Even though this challenge of a vast input space with unpredictable corner cases is not specific to robotic systems, it can be more manageable in non-robotics, software systems. In software systems (e.g., a web application), well-defined interfaces control and limit the range of inputs that can be received from external sources (e.g., users). P12 elaborated:

> *Software systems need to communicate only within themselves and you can strongly define the range of inputs that will come in. When a user is involved, the range of inputs grows, but it is limited by the range of inputs that can be produced by the user. When you have a physical system that needs to interact with the real world, you need to handle the vast state space and noise in the real world.*

They later shared an example of this problem where the hardware for their robot was affected by very low temperatures in the field. At -30°C, some of the hardware started misbehaving and produced unexpected sensor data, which could lead to poor algorithmic decisions and unexpected behaviors. This event was something that had not been anticipated before it was actually witnessed in the field.

[5]https://compliancetesting.com

(C2) Engineering complexity: The engineering effort required to prepare all pieces needed for testing a robot can be extremely high, as these systems can be very complex. All of our participants unanimously described their systems as extremely complex. P12 believes that robotics field is far away from deploying complex systems. They said:

> *As an industry, we haven't managed to deploy anything more complex than a Roomba, which basically operates using a one-dimensional input.*

One aspect of engineering complexity involves the amount of scaffolding that is required to put the system into a testable state [3]. For example, P2 and P6 both consider it a challenge to write tests for incomplete components such as cases where the hardware of the system has not yet been fully designed or manufactured. P10 said:

> *Whenever working with network protocols, I see whether anyone has already written a protocol. If not, I'll start by creating a Wireshark plugin to debug the protocol before I start working on it.*

Another engineering complexity affecting test design is the specification of test inputs. To design realistic inputs, roboticists sometimes need to collect data from the real world, which may be a challenge (e.g., a space rover). For instance, P4 mentioned the need to collect gigabytes of LIDAR data to reasonably test a small snippet of code.

Finally, the complexity of the system itself creates a challenge for roboticists to design and write tests that, as P9 puts it, "effectively validate all requirements for the system". P11 finds it difficult to understand what needs to be tested, and design tests that clearly signal failures and help the developers to identify the source of failures. They later mentioned that, based on their experience, writing tests can consume more time than the actual implementation. P12 believes that writing tests sometimes requires knowledge about many fields such as computer science, mathematics, and engineering.

(C3) Culture of testing: Our participants referred to a culture of not believing in the value of testing in their company or institute among not only the roboticists, but also their sponsors and customers. For example, P4 mentioned that many developers do not see much practical value in unit tests, even though they theoretically understand the value of having them. The prevalence of such culture within a community may result in developers getting discouraged from writing effective tests. Both P4 and P9 mentioned being under pressure by their sponsors and clients to deliver the product as quickly as possible, and being discouraged from spending time on writing tests.

One of the characteristics of the robotics community is that it brings together people from many different disciplines (e.g., electrical and mechanical engineering). While the diversity of the community is a driving factor for many great advances in robotics, it can also introduce challenges. As P11 said:

> *The world of robotics unites folks from different backgrounds. Folks from a software background might observe testing differently from those who aren't.*

TABLE IV
A SUMMARY OF THE CHALLENGES OF WRITING AND DESIGNING TESTS FOR ROBOTICS THAT WE IDENTIFIED BASED ON PARTICIPANT RESPONSES. CHALLENGES IN BOLD ARE DISCUSSED IN MORE DETAIL IN THE TEXT.

ID	Title	Description	Representative quote
C1	**Unpredictable corner cases**	The challenge of attempting to anticipate and cover for all possible edge cases within a large (and possibly unbounded) state space when designing tests.	"*We know that the customer is going to do something really stupid with the robot. It's hard to imagine what stupid things some [users] are going to do.*" – P8
C2	**Engineering complexity**	A disproportionate level of engineering effort is required to build and maintain end-to-end test harnesses for robotic systems with respect to the benefit of those tests.	"*Writing tests for a [ROS] node can be difficult because you have to have all the workspace built to run your unit tests.*" – P1
C3	**Culture of testing**	The challenge of operating within a culture that places little value on testing and provides developers with few incentives to write tests.	"*The biggest bottleneck in testing robotics is developers not wanting to do it.*" – P1
C4	**Coordination, collaboration, and documentation**	A lack of proper channels for coordination and collaboration among multiple teams (especially software and hardware teams), and a lack of documentation.	"*There are so many people working on the same software stack. Sometimes there will be inconsistency.*" – P3

Another cultural aspect of the robotics community that impacts testing practices pertains to the age of the industry and its associated startup culture that often values rapid prototyping and development over testing and quality assurance. P10 said:

> *The robotics community are more focused on making cool things than software quality and making things better.*

The desire to be first to the market and having the robot with greatest number of features is often valued more than the quality and robustness of the robot.

Finally, we observed from the responses of our participants that there is often a high degree of reliance upon intuition during testing and development. P2, P4, P6, and P12 all specifically mentioned their intuition as an important tool for testing and debugging. For example, P2 said:

> *Personally, I have an intuition. I think I know what when something goes wrong.*

(C4) Coordination, collaboration, and documentation: In many robotics companies, significant coordination and collaboration is required to design meaningful tests for the system (especially for full system tests). This coordination can take place between separate development and testing teams, or between software and hardware teams. For example, both P2 and P3 found it challenging to integrate components developed by different teams and to coordinate final full-system testing after integration of software and hardware. A lack of documentation for third-party components adds further complexity to writing tests. Many robotic systems consist of third-party components for which full access to the source code is not granted. Furthermore, developers are often less familiar with such components since they did not develop those components themselves, and as such, they need to refer to documentation when designing tests involving third-party components. P10 faced this challenge when writing tests involving a third-party component without any form of documentation.

An additional challenge is that there are very few standards and guidelines for practitioners to guide their testing. P8 said:

> *A standard for robotic system testing would be neat. A process to follow. Like "here's how you assess a robotic system".*

The more popular and advanced subfields of robotics (e.g., self-driving vehicles) are already beginning to provide standards and certification [33]. As P10 describes, "some areas of robotics have very crystalized safety regulations".

C. RQ3: Challenges of running and automating tests

We asked our participants to describe the challenges they face when running tests on their systems, and the challenges of automating their tests. Below, we describe the five challenges, summarized in Table V, that we extracted from participant responses.

(C5) Cost and resources: There are several costs involved in running tests for robotics systems. First, conducting manual field testing can be dangerous and expensive. P10 described an accident where the robotic arm behaved unexpectedly, and crashed into the tester on their knee. They further said:

> *Once you experience a few accidents [during field testing], you realize that testing is really dangerous. If a typical software system like Excel crashes, no-one dies. For robotics, that certainly isn't the case.*

Second, developers and test teams need to spend many hours running test scenarios on the robot. The test team of P3's company receives tens to hundreds of test requests every day, but their time and resources (e.g., physical robots) are limited. Given limited time and resources, developers and testers are forced to select, prioritize, and minimize tests in order to test as many changes to the software or requirements as is possible. Third, it may take a long time to run the tests. P1, P6, P8, and P9 all find the long running time of tests as one of the challenges of testing.

Finally, the cost of the equipment and setup required for running tests may be prohibitively expensive for smaller companies. P10 said that their robots are so expensive that they need to be extremely careful when interacting with them. To be able to design large-scale automated tests for their robot,

TABLE V
SUMMARY OF CHALLENGES IDENTIFIED WHEN ASKING ABOUT RUNNING AND AUTOMATING TESTS FOR ROBOTICS. THE ONES IN BOLD ARE DISCUSSED IN MORE DETAILS.

ID	Title	Description	Representative quote
C5	**Cost and resources**	The cost of running and automating the tests in terms of human hours, resources and setup, and running time.	"*Full testing is expensive because you have to pay for the labor for someone to test. If there is downtime for robots, then there is cost to pay people to wait for the fix.*" – P9
C6	**Environmental complexity**	The inherent difficulties of attempting to account for the complexities of the real world when simulating, testing, and reproducing full-system behavior.	"*Simulation just doesn't reflect the real world. It is hard to find a test environment [for our robot]. Logistics are also difficult, like how the sunlight affects sensors.*" – P2
C7	**Lack of oracle**	The challenge of specifying an oracle that can automatically distinguish between correct and incorrect behavior.	"*Manual tests are much more open-ended and can be used to test strange, unexpected behaviors.*" – P6
C8	**Software and hardware integration**	Difficulties that arise when different software and hardware components of the system are integrated and tested.	"*The specification says something, you implemented it correctly, the test succeeds, but then when you deploy it to the actual robot, it fails and you're confused; turns out that different things happen when you run on the real hardware.*" – P10
C9	**Distrust of simulation**	A lack of confidence in the accuracy and validity of results obtained by testing in simulation and synthetic environments, and a sole reliance on field testing.	"*It's unlikely that bugs found in the field would be found in simulation testing.*" – P4

P8 needs to build a framework that can automatically capture the state of the robot. P8 believed that it is less expensive for their company to pay an intern to manually test the robot, rather than designing a computer vision platform that will allow them to write automated tests. P1 uses simulation for running tests, but finds simulation a bottleneck of their testing practices because of its low speed and the amount of resources required for running it.

Similarly, automating tests requires significant efforts and investment that may be considered too expensive in terms of developer hours and resources. P1 describes automating tests of specific hardware and network interactions as "too much work" as they need to use mocks and patches to imitate other components. P2 and P11 both claim that establishing an infrastructure for automated testing (via simulation) is very difficult and expensive. P8 and P9 believe that it is always possible to hit deeper levels of test automation as long as the cost is justifiable. For example, P8 said:

> *There's a trade-off between cost of automating tests and number of times that we have to run them. We don't need to run some tests very often, so we don't really need to automate them.*

(C6) Environmental complexity: The intended operating environment for a robot can be very complex: robots are embodied within the unpredictable and practically boundless state of space of reality, and their behavior may be dependent upon certain physical features (e.g., terrain) and phenomena (e.g., lighting, weather). Finding a suitable environment for testing the robot under expected operating conditions (e.g., on Mars or in the deep ocean) can be challenging: P8 mentioned that a challenge they face is finding as many qualitatively different physical locations as they can to test their robot, since every environment may have characteristics that reveals problems in their system. However, these environments sometimes constrain the number and quality of tests that can be run.

The complexity of attempting to model physical reality also complicates the development of high-fidelity simulators, which are extremely important for both running and automating tests [5], [21]. P12 believes that "no simulators currently exist where the information is even close to the reality, they are nowhere close to the noise and variability of real-world data". P2 and P10 also believe that simulation cannot provide sufficient fidelity for testing robots in realistic scenarios.

Finally, complexities of the real world can hinder the reproduction of bugs and certain tests. P6, P11, and P12 have all faced the challenge of reproducing bugs they discovered in the field. P12 used the term *Heisenbugs* [37] to describe these bugs that will only manifest when you are not looking for them. P11 believes that, even though record and replay has many benefits, it is not ultimate solution to reproducibility since you need to make sure that the replayed state is true to the world (e.g., with respect to timestamps and orderings).

Record and replay was reported as a popular approach for dealing with the challenges of testing systems with complex environments (discussed in Section IV-A). Sensor data is recorded in the field and then replayed for testing purposes. This approach has advantages, in that it uses real data and it is often easier to collect data than to synthetically create large volumes of data. However, there are also significant limitations to this approach. Without enough varied data, developers can run the risk of overfitting their approach to the recorded data, which might not represent the true variety of environments the robot will operate in. Additionally, because of the non-interactive nature of record-and-replay, it cannot be used for testing scenarios where there must be a feedback loop between the robot and the environment. P11 said:

> *Simulators are expensive, especially if you have to write your own. The more you are trying to test interactions with the physical world, the more value you will see in simulation. If there is less interaction, then record and replay is preferable.*

(C7) Lack of oracle: The well-known oracle problem concerns how to distinguish whether a given system behavior is correct or incorrect [38]. Fully automated tests require an oracle that can automatically determine the correctness of system behavior. Because of the noisy and non-deterministic nature of robotic systems, it is difficult to discretely specify the exact behavior that is intended [39]. For example, consider that a robot is instructed to move to a given position, but that the robot stops 5 centimeters away from the exact coordinates of its destination. Should such an outcome be deemed faulty or acceptable? In any case, due to inherently noisy sensing and actuation, the robot is highly unlikely to reach the exact intended position or to determine whether that position has been reached. Both P4 and P6 find specifying automated oracles challenging.

Furthermore, as explained by P4, in some cases, collecting data for the ground truth is either impossible or extremely expensive. P4 provided an example where a camera responsible for measuring the relative motion between two vehicles was under test. To validate the correctness of data provided by this camera, they needed a second method of measuring relative motion between the vehicles to act as the ground truth. The equipment and setup required to reach this ground truth turned out to be extremely expensive.

Following challenges described in C1, the vast space of inputs and corner cases makes it difficult to cleanly discriminate between correct-but-unusual and incorrect behaviors. P12 described this difficulty of defining suitable oracles for automated testing as "difficult to differentiate between bad behavior and correct, but strange behavior that is produced by unexpected inputs".

(C8) Software and hardware integration: Robotic systems consist of software and hardware components [2], [7]. When asked about the most important feature of robotic systems that complicates testing, P1 responded with "robotic hardware". P2 said:

> *Robotics as a field is all about integration. Robotics is where hardware, software, and the world come together.*

To better understand the differences of a system with and without hardware components, let us present a quote from P9:

> *At the full hardware level, we see hardware that is flaky, like not assembling the cooling properly. In parts of robotics, you are writing multiple pieces of software, and you are running on specialized hardware, which might be optimized for performance, so there are extra concerns beyond traditional testing practices.*

The integration of components into a system can create unique testing challenges. P8 shared that even when software and hardware parts work properly in isolation, they frequently break once the software is ran on physical hardware. In P9's opinion, developing a robotic system resembles developing many software and hardware systems all together (e.g., sensing, planning, and manipulation), and the simplest robot is at least three subsystems. Even though these subsystems work in isolation, unexpected failure modes are observed when they are combined. P3, P6, and P10 all faced confusion and challenges while running tests after integration of software and hardware components. P12 provided an interesting example of being limited by the battery on the robot after integrating software and hardware but not needing to worry about such problems when solely testing the software.

(C9) Distrust of simulation: Simulation-based testing appears to be a promising approach to the challenge of test automation within the field of robotics [5], [21]. In the absence of simulation, full-system tests need to be executed on the real-world hardware in a real, physical environment, which significantly constrains the possibilities for test automation (e.g., regulations applied to testing autonomous vehicles on public roads).

Despite being aware of the theoretical value in using simulation to automate parts of the system testing process, many of our participants reported that they distrust the accuracy and validity of simulated operations. P2, P4, P8, P10, and P12 all believe the fidelity of simulation is not sufficiently high for testing and that running tests on the actual robot is the only way to test the system. For example, P2 said:

> *We mostly do field testing. That's what really affects what happens. The robot gets lots of impact from the environment. Simulation just doesn't reflect the real world.*

The lack of trust in synthetic results discourages developers from using software-based simulation as part of their test automation. In part, this could stem from the perceptions that our participants shared with us that many simulation tools are difficult to use and are more hassle than they are worth. For example, P10 mentioned that they could extend the simulator to be more faithful to the real world, but it is not worth the amount of time and effort to do that when they can just test on the real robot. P8 said:

> *I have more bodies that can test the hardware. I don't have time to build a Gazebo [plugin]. Getting the cameras to work properly in simulation is difficult.*

However, in some areas of robotics, notably self-driving cars, significant investments have been made in improving simulation [40]–[42]. P2 shared their opinion on this matter:

> *Robotics operates in such a variety of domains that developing high fidelity simulators is very difficult and for the most part do not exist today. However, if they did exist (and people trusted their fidelity) I think people would use them.*

V. Interpretation and Discussion

In Section IV, we identified 12 testing practices used by robotics companies, and 9 challenges that roboticists face when designing, running, and automating tests. In this section, we identify 3 major themes among the identified challenges. We support these themes by showing quotes from our participants, and later speculate on the implications of each theme and provide suggestions for tackling their associated challenges. Figure 1 describes the association between each of

ID	Title	Real-world complexities	Community & standards	Component integration
C1	Unpredictable corner cases	•		
C2	Engineering complexity	•		•
C3	Culture of testing		•	
C4	Coordination, collaboration, and documentation		•	
C5	Cost and resources	•	•	•
C6	Environmental complexity	•		
C7	Lack of oracle	•		•
C8	Software and hardware integration			•
C9	Distrust of simulation	•		

Fig. 1. For each theme, we indicate the challenges that support that theme.

the three themes and the challenges of testing robotic systems that participants reported.

Real-world complexities: By definition, robotic systems interact with the real world. This feature results in one of the major differences between robotic systems and traditional software systems. Interaction with complex, real-world environments is one of the most prominent challenges of testing robots that we observed in our interviews, and as P8 said:

> *Very little of the work on testing takes into account the physical aspects of the problem.*

The complexities of the real world contribute to C1 and C2 as the large input space results in unpredictable corner cases and engineering complexity of specifying test inputs, and adds more complications to defining oracles discussed in C7. C6 and C9 are both impacted by real-world complexities as it is too difficult to make an abstraction of the environment, and testing in physical environments requires more resources (C5).

One way to attempt to simplify the complexities of the real world for the purposes of testing is simulation. However, developers still encounter many barriers when they attempt to use simulation [43]. As pointed out by our participants, simulators sometimes abstract away too many nuances of the real-world, and so developers do not feel comfortable relying on them. In other cases, our participants responded that they feel that simulators are excessively complex to deploy, and since simulators often do not provide tools to manage that complexity, developers choose not to use them.

Other techniques that may be brought to bear on the challenges that arise from the interaction between robotic systems and the real world include *record and replay*, *model checking*, and *formal specification* [3], [44]–[49]. As mentioned in C6, record and replay is a popular approach for dealing with the challenges of testing systems with complex environments. However, it is only a partial solution, because of its non-interactive nature. Model checking and formal specification are other solutions proposed to decrease reliance on simulators for automated testing by abstracting away the complexities of the real-world [23], [50]–[55]. However, these systems are limited to specific types of systems, and, based on our study, have not yet been generally adopted in practice.

Furthermore, devising suitable oracles for full-system testing can quickly become an overwhelming task, as described in C7. To test their system, a developer may need to provide an oracle for several interrelated subsystems, all of which provide complex data. This can quickly become an overwhelming challenge. We believe that addressing this challenge of defining oracles for robotic systems requires the development of novel methods and techniques by the software engineering and robotics communities. In recent years, a number of studies have taken important steps towards tackling this problem [39], [56]–[58].

As a way of approaching the oracle problem for CPSs, studies have used metamorphic testing to observe the relations between the inputs and outputs of multiple executions of a CPS [55], [59], [60]. Even though metamorphic testing is a promising approach towards the oracle problem for cyberphysical and robotic systems, it requires identifying and proving metamorphic relations in the system [61].

We believe that the design and development of higher-fidelity simulators with better user interfaces and APIs may lead to a wider adoption of automated simulation testing. However, in absence of such simulators, the research community should develop novel tools and techniques for achieving test automation.

Community and standards: Not all barriers to testing robotic systems are a result of technical issues. Another important theme of challenges we encountered are challenges that stem from community and standards. From our study, we learned that the robotics community is diverse and that people from different backgrounds may value testing and validation differently (C3). Many robotics practitioners are not familiar with methods of software testing (e.g., robustness testing and performance testing), and need guidelines to assist them in deploying testing practices (C4). With notable exceptions (e.g., industrial automation), the robotics industry is relatively young and immature, and the value of being the first to the market often outvalues the safety and quality of the system (C5).

Standards create an advantage for robotics companies by approving the product quality to the customers, and increasing the business value of testing. In our study, we found that robotics companies sometimes have standards from other

industries that they can apply to certain domain-specific parts of their system, such as IEC standards from the vacuum industry for how many particles a vacuum should pick up [62]. However, for the robotics part of the system, there are often no standards or guidelines. A number of standards have already been introduced for sub-domains of robotics such as self-driving cars [33], [53], taking steps towards the right direction. However, we believe that more general-purpose guidelines and standards should be implemented to guide robotics developers, similar to those provided to other industries by UL or ISO [63], [64].

Component integration: The third factor that complicates testing in robotic systems is the challenge of testing integrated hardware and software systems. In addition to C8, which is directly associated with this theme, integration of components contributes to C2, C5, and C7 as it increases the complexity of the system, the cost associated with testing, and introduces complications when defining oracles.

Some of the hardware and software challenges are similar to those found in embedded systems: timing, power consumption, memory allocation, and architecture [65]. However, in comparison to embedded systems, robotics hardware is often much more expensive and complex.

In many cases, the considerable expense of manufacturing robotic systems can limit the availability of hardware for testing. While embedded systems are often small, low-power devices with a fixed form and function, robots are often more of an extendible platform upon which physical components (i.e., sensors and actuators) can be added and removed over time. We believe that the development of tools and practices for testing robots in a more controlled fashion (e.g., hardware-in-the-loop testing [66]) may reduce the costs and risks associated with field testing on expensive hardware.

We believe that these three themes best describe the major challenges of robotics testing. Although partial solutions exist for some of these challenges [15], [16], [20], [39], [56], [67], in theory, the applicability and effectiveness of those solutions in practice remains unstudied and unclear. We observed that testing practices that are not represented by these themes, such as unit testing, continuous integration, and plan-based testing, were adopted by most of our participants. We also observe that the level of associated tooling and support for a given practice influences the uptake of that practice among robotics developers. For example, continuous integration is a practice that is well supported by tools and has been adopted by many of our participants. Logging and playback is also extremely popular among our participants. One reason behind this popularity could be the well-established tools and support around them, even though logging and playback still face challenges such as C1. In contrast, simulation testing and robustness testing are rarely adopted by our participants, as supporting tools and infrastructure have not been properly established yet.

VI. Threats to Validity

Replicability *Can others replicate our results?* In general, qualitative studies can be difficult to replicate. We address this threat by making our interview script available on our companion site.[6] We cannot publish the interview transcripts because we promised our subjects that we would preserve their anonymity.

Construct *Are we asking the right question?* We used semi-structured interviews [30] to explore themes while also letting participants bring up new ideas throughout the interview. By allowing participants the freedom to bring up topics, we avoid biasing the interviews with our preconceived understanding of testing in robotics.

Internal *Did we skew the accuracy of our results with how we collected and analyzed the information?* Interviews can be affected by intentional or unintentional bias. To mitigate this concern, we followed established guidelines from literature [68], both designing and performing our interviews. Additionally, we ran a series of iterative pilots with robotics engineers, which we did not consider as data for the purposes of this work, but helped us shape a productive interview.

External *Do our results generalize?* Because there is a lot that is not known about testing in robotics, in this work we decided to prioritize depth over breadth. While our interviews did generate very rich data for us to analyze, we cannot make broad claims about how prevalent these practices are across the industry. To mitigate this threat, we constructed a sample with a specific eye for breadth, interviewing participants across a wide range of companies, sizes, and sectors.

VII. Conclusion

In this paper, we studied robotics testing practices and challenges by conducting a qualitative study with 12 robotics practitioners from 11 different robotics companies. We identified 12 testing practices mentioned by roboticists in practice, and 9 challenges they face while testing their systems. We identified three main themes of challenges that impact testing practices in the field of robotics: *real-world complexities*, *community and standards*, and *component integration*. We believe that there is a gap in robotics testing and that the research community should focus on these three major themes of challenges to address the challenges of robotics testing and to develop the next generation of quality assurance techniques for robotic systems.

Acknowledgment

This research was partially supported by AFRL (#FA8750-15-2-0075), DARPA (#FA8750-16-2-0042), and the NSF (#CCF-1563797); the authors are grateful for their support. Any opinions, findings, or recommendations expressed are those of the authors and do not necessarily reflect those of the US Government.

[6]https://doi.org/10.5281/zenodo.3625199

REFERENCES

[1] "Self-Driving Uber Car Kills Pedestrian in Arizona, Where Robots Roam," https://www.nytimes.com/2018/03/19/technology/uber-driverless-fatality.html, accessed: 2020-01-03.

[2] L. Esterle and R. Grosu, "Cyber-physical systems: challenge of the 21st century," *e & i Elektrotechnik und Informationstechnik*, vol. 133, no. 7, pp. 299–303, 2016.

[3] C. Hutchison, M. Zizyte, P. E. Lanigan, D. Guttendorf, M. Wagner, C. Le Goues, and P. Koopman, "Robustness testing of autonomy software," in *International Conference on Software Engineering: Software Engineering in Practice*, ser. SEIP'18. ACM, 2018, pp. 276–285.

[4] P. Duan, Y. Zhou, X. Gong, and B. Li, "A systematic mapping study on the verification of cyber-physical systems," *IEEE Access*, vol. 6, pp. 59 043–59 064, 2018.

[5] S. A. Seshia, S. Hu, W. Li, and Q. Zhu, "Design automation of cyber-physical systems: Challenges, advances, and opportunities," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, pp. 1421–1434, 2016.

[6] D. Marijan, A. Gotlieb, and M. K. Ahuja, "Challenges of testing machine learning based systems," in *International Conference On Artificial Intelligence Testing*, ser. AITest'19. IEEE, 2019, pp. 101–102.

[7] H. Li, *Communications for control in cyber physical systems: theory, design and applications in smart grids*. Morgan Kaufmann, 2016, ch. 1-Introduction to cyber physical systems.

[8] P. Runeson, "A survey of unit testing practices," *IEEE software*, vol. 23, no. 4, pp. 22–29, 2006.

[9] A. Causevic, D. Sundmark, and S. Punnekkat, "An industrial survey on contemporary aspects of software testing," in *International Conference on Software Testing, Verification and Validation*, ser. ICST'10. IEEE, 2010, pp. 393–401.

[10] V. Garousi and J. Zhi, "A survey of software testing practices in canada," *Journal of Systems and Software*, vol. 86, no. 5, pp. 1354–1376, 2013.

[11] S. Ng, T. Murnane, K. Reed, D. Grant, and T. Chen, "A preliminary survey on software testing practices in australia," in *Australian Software Engineering Conference*. IEEE, 2004, pp. 116–125.

[12] E. Engström and P. Runeson, "A qualitative survey of regression testing practices," in *International Conference on Product Focused Software Process Improvement*. Springer, 2010, pp. 3–16.

[13] S. K. Khaitan and J. D. McCalley, "Design techniques and applications of cyberphysical systems: A survey," *IEEE Systems Journal*, vol. 9, no. 2, pp. 350–365, 2014.

[14] P. Ammann and J. Offutt, *Introduction to Software Testing*, 1st ed. Cambridge University Press, 2008.

[15] A. Paikan, S. Traversaro, F. Nori, and L. Natale, "A generic testing framework for test driven development of robotic systems," in *International Workshop on Modelling and Simulation for Autonomous Systems*. Springer, 2015, pp. 216–225.

[16] S. Sheng and N. Becker, "Challenges in standardizing ram testing for small unmanned robotic systems," in *Reliability and Maintainability Symposium*, ser. RAMS'13. IEEE, 2013, pp. 1–6.

[17] Y. K. Chung and S.-M. Hwang, "Software testing for intelligent robots," in *International Conference on Control, Automation and Systems*. IEEE, 2007, pp. 2344–2349.

[18] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 794–807, 2008.

[19] I. Díaz, J. J. Gil, and E. Sánchez, "Lower-limb robotic rehabilitation: literature review and challenges," *Journal of Robotics*, vol. 2011, 2011.

[20] M. Mossige, A. Gotlieb, and H. Meling, "Testing robot controllers using constraint programming and continuous integration," *Information and Software Technology*, vol. 57, pp. 169–185, 2015.

[21] X. Zheng, C. Julien, M. Kim, and S. Khurshid, "On the state of the art in verification and validation in cyber physical systems," *Technical Report at The University of Texas at Austin, The Center for Advanced Research in Software Engineering*, vol. 1485, 2014.

[22] A. Alami, Y. Dittrich, and A. Wasowski, "Influencers of quality assurance in an open source community," in *International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE'18. IEEE, 2018, pp. 61–68.

[23] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *ACM Computing Surveys*, vol. 52, no. 5, p. 100, 2019.

[24] C. Bogart, C. Kästner, J. Herbsleb, and F. Thung, "How to break an api: cost negotiation and community values in three software ecosystems," in *International Symposium on Foundations of Software Engineering*, ser. FSE'16. ACM, 2016, pp. 109–120.

[25] S. Phillips, T. Zimmermann, and C. Bird, "Understanding and improving software build teams," in *International Conference on Software Engineering*, ser. ICSE'14, 2014.

[26] T. D. LaToza, G. Venolia, and R. DeLine, "Maintaining mental models: A study of developer work habits," in *International Conference on Software Engineering*, ser. ICSE'06, 2006.

[27] K. Muşlu, C. Bird, N. Nagappan, and J. Czerwonka, "Transition from Centralized to Decentralized Version Control Systems: A Case Study on Reasons, Barriers, and Outcomes," in *International Conference on Software Engineering*, ser. ICSE'14, 2014.

[28] Y. Tao, Y. Dang, T. Xie, D. Zhang, and S. Kim, "How Do Software Engineers Understand Code Changes? – An Exploratory Study in Industry," in *International Symposium on the Foundations of Software Engineering*, ser. FSE'12, 2012.

[29] C. McNamara, "General guidelines for conducting interviews," 1999.

[30] F. Shull, J. Singer, and D. I. K. Sjøberg, Eds., *Guide to Advanced Empirical Software Engineering*, 2008.

[31] C. S. Timperley, A. Afzal, D. S. Katz, J. M. Hernandez, and C. Le Goues, "Crashing simulated planes is cheap: Can simulation detect robotics bugs early?" in *International Conference on Software Testing, Verification and Validation*, ser. ICST'18. IEEE, 2018, pp. 331–342.

[32] T. Sotiropoulos, H. Waeselynck, J. Guiochet, and F. Ingrand, "Can robot navigation bugs be found in simulation? an exploratory study," in *International Conference on Software Quality, Reliability and Security*, ser. QRS'17. IEEE, 2017, pp. 150–159.

[33] EdgeCase, "Ul 4600: The first comprehensive safety standard for autonomous products," https://edge-case-research.com/ul4600/, 2019, [Online; accessed 4-October-2019].

[34] D. B. Rawat, J. J. Rodrigues, and I. Stojmenovic, *Cyber-physical systems: from theory to practice*. CRC Press, 2015.

[35] S. Ali, H. Lu, S. Wang, T. Yue, and M. Zhang, "Uncertainty-wise testing of cyber-physical systems," in *Advances in Computers*. Elsevier, 2017, vol. 107, pp. 23–94.

[36] R. Banabic, "Techniques for identifying elusive corner-case bugs in systems software," EPFL, Tech. Rep., 2015.

[37] J. Gray, "Why do computers stop and what can be done about it?" in *Symposium on reliability in distributed software and database systems*, 1986, pp. 3–12.

[38] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2014.

[39] Z. He, Y. Chen, E. Huang, Q. Wang, Y. Pei, and H. Yuan, "A system identification based oracle for control-cps software fault localization," in *International Conference on Software Engineering*, ser. ICSE'19. IEEE Press, 2019, pp. 116–127.

[40] "A Waymo engineer told us why a virtual-world simulation is crucial to the future of self-driving cars," https://www.businessinsider.com/waymo-engineer-explains-why-testing-self-driving-cars-virtually-is-critical-2018-8, accessed: 2020-01-03.

[41] "Simulation Becomes Increasingly Important For Self-Driving Cars," https://www.forbes.com/sites/davidsilver/2018/11/01/simulation-becomes-increasingly-important-for-self-driving-cars/#56b1fa045583, accessed: 2019-10-13.

[42] "NVIDIA Driver Constellation," https://www.nvidia.com/en-us/self-driving-cars/drive-constellation/, accessed: 2019-10-13.

[43] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX," in *International Conference on Robotics and Automation*, ser. ICRA'15. IEEE, 2015, pp. 4397–4404.

[44] A. Gambi, T. Huynh, and G. Fraser, "Generating effective test cases for self-driving cars from police reports," in *Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE '19, 2019, pp. 257–267.

[45] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *International Symposium on Software Testing and Analysis*, ser. ISSTA '19, 2019, pp. 318–328.

[46] C. A. González, M. Varmazyar, S. Nejati, L. C. Briand, and Y. Isasi, "Enabling model testing of cyber-physical systems," in *International*

Conference on Model Driven Engineering Languages and Systems, ser. MODELS'18. ACM, 2018, pp. 176–186.

[47] A. Dokhanchi, B. Hoxha, and G. Fainekos, "Formal requirement debugging for testing and verification of cyber-physical systems," *ACM Transactions on Embedded Computing Systems*, vol. 17, no. 2, p. 34, 2018.

[48] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[49] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *International Conference on Automated Software Engineering*, ser. ASE'18. ACM, 2018, pp. 132–142.

[50] A. Desai, T. Dreossi, and S. A. Seshia, "Combining model checking and runtime verification for safe robotics," in *International Conference on Runtime Verification*. Springer, 2017, pp. 172–189.

[51] M. Farrell, M. Luckcuck, and M. Fisher, "Robotics and integrated formal methods: necessity meets opportunity," in *International Conference on Integrated Formal Methods*. Springer, 2018, pp. 161–171.

[52] A. Desai, S. Qadeer, and S. A. Seshia, "Programming safe robotics systems: Challenges and advances," in *International Symposium on Leveraging Applications of Formal Methods*. Springer, 2018, pp. 103–119.

[53] F. Ingrand, "Recent trends in formal validation and verification of autonomous robots software," in *International Conference on Robotic Computing*, ser. IRC'19. IEEE, 2019, pp. 321–328.

[54] E. Zibaei, S. Banescu, and A. Pretschner, "Diagnosis of safety incidents for cyber-physical systems: A uav example," in *International Conference on System Reliability and Safety*, ser. ICSRS'18. IEEE, 2018, pp. 120–129.

[55] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: automated testing of deep-neural-network-driven autonomous cars," in *International Conference on Software Engineering*, ser. ICSE '18, 2018, pp. 303–314.

[56] Y. Chen, C. M. Poskitt, and J. Sun, "Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system," in *Symposium on Security and Privacy*, ser. SP'18. IEEE, 2018, pp. 648–660.

[57] N. Ye, S. M. Emran, Q. Chen, and S. Vilbert, "Multivariate statistical analysis of audit trails for host-based intrusion detection," *Transactions on computers*, vol. 51, no. 7, pp. 810–820, 2002.

[58] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," in *International Conference on Data Mining Workshops*, ser. ICDMW '17, 2017, pp. 1058–1065.

[59] Z. Q. Zhou and L. Sun, "Metamorphic testing of driverless cars," *Communications of the ACM*, vol. 62, no. 3, pp. 61–67, Feb. 2019.

[60] M. Lindvall, A. Porter, G. Magnusson, and C. Schulze, "Metamorphic model-based testing of autonomous systems," in *International Workshop on Metamorphic Testing*, ser. MET '17, 2017, pp. 35–41.

[61] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Computing Surveys*, vol. 51, no. 1, p. 4, 2018.

[62] "IEC standards," https://webstore.iec.ch/publication/30410, accessed: 2019-10-13.

[63] "Ul standards," https://ulstandards.ul.com, accessed: 2019-10-13.

[64] "ISO standards," https://www.iso.org/standards.html, accessed: 2019-10-13.

[65] E. A. Lee and S. A. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*. MIT Press, 2016.

[66] H. K. Fathy, Z. S. Filipi, J. Hagena, and J. L. Stein, "Review of hardware-in-the-loop simulation and its prospects in the automotive area," in *Modeling and Simulation for Military Applications*, K. Schum and A. F. Sisti, Eds., vol. 6228, International Society for Optics and Photonics. SPIE, 2006, pp. 117–136.

[67] M. Zhang, S. Ali, T. Yue, R. Norgren, and O. Okariz, "Uncertainty-wise cyber-physical system test modeling," *Software & Systems Modeling*, vol. 18, no. 2, pp. 1379–1418, 2019.

[68] I. Seidman, *Interviewing as Qualitative Research: A Guide for Researchers in Education and the Social Sciences*. Teachers College Press, 2006. [Online]. Available: https://books.google.com/books?id=pk1Rmq-Y15QC