Simulation for Robotics Test Automation: Developer Perspectives

Afsoon Afzal,[1] Deborah S. Katz,[1] Claire Le Goues, and Christopher S. Timperley
Carnegie Mellon University, Pittsburgh, PA
Email: afsoona@cs.cmu.edu, dskatz@gmail.com, clegoues@cs.cmu.edu, ctimperley@cmu.edu

***Abstract*—Robotics simulation plays an important role in the design, development, and verification and validation of robotics systems. Simulation represents a potentially cheaper, safer, and more reliable alternative to the widely used practice of manual field testing, and introduces valuable opportunities for extensive test automation. The goal of this paper is to develop a principled understanding of the ways robotics developers use simulation in their testing processes and the challenges they face in doing so. This understanding can guide the improvement of simulators and testing techniques for modern robotics development.**

To that end, we conduct a survey of 82 robotics developers from a diversity of backgrounds, addressing the current capabilities and limits of simulation in practice. We find that simulation is used by 84% of our participants for testing, and that many participants want to use simulation as part of their test automation. Using qualitative and quantitative research methods, we identify 10 high-level challenges that impede developers from using simulation for manual and automated testing and in general. These challenges include the gap between simulation and reality, a lack of reproducibility, and considerable resource costs associated with simulation. Finally, we outline ways in which simulators can be improved for use as a means of verification and validation and ways that the software engineering community can contribute to these improvements.

***Index Terms*—robotics simulation challenges; robotics testing; simulation testing; testing challenges; empirical study; practitioner survey**

I. Introduction

From autonomously driving trucks and cars to packaging and shipping goods, robotics and autonomous systems are set to play an important role in our lives for the indefinite future [1]–[3]. Such systems have the potential to automate or assist many dull, dirty, dangerous, and difficult jobs. In response to the COVID-19 crisis, for example, robots have been used to deliver food to quarantined patients, disinfect public places, and cope with increased supply chain demands [4]–[7]. However, these systems also present new and unexpected opportunities for catastrophic failure, resulting in tremendous economic and human damage [8]–[11].

It is therefore vital that we continue to develop effective quality assurance techniques for robotics and autonomous systems. Presently, the predominant means of quality assurance for robotics is *field testing* — testing robots in a physical environment resembling the intended deployment [12], [13]. Field testing allows both the robot hardware and software stack to be tested in a realistic and inherently complex environment. However, field testing is time consuming and expensive, not scalable, error-prone and potentially dangerous, and can only be performed at the later stages of development when both the software stack and hardware platform have reached maturity.

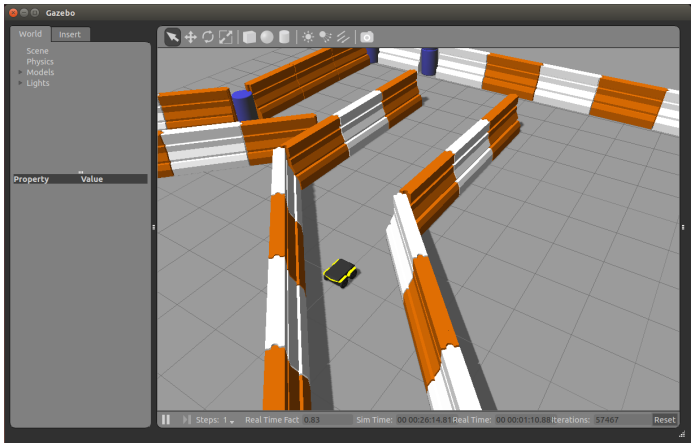

Fig. 1. A simulation of an unmanned ground vehicle, Jackal, in the Gazebo simulator [Source: http://www.clearpathrobotics.com].

Simulation-based testing, in which the robot controller is tested in a simulated environment, such as in Figure 1, is a promising method for testing robotics systems that is cheaper, safer, and more scalable than the manual process of field testing, with significant potential for test automation [14]–[17].

Researchers have used simulation to effectively and automatically discover bugs in a variety of robot application domains [18]–[21] and have shown that even cheaper, less-realistic simulations are capable of revealing many bugs [15]. Numerous companies in the autonomy sector, such as Uber [22], NVIDIA [23], and Waymo [24], use simulation on a large scale to develop, train, and test their algorithms.

Given the considerable potential of simulation-based testing, we set out to understand the extent to which simulation is used for testing in practice and identify the barriers that prevent wider use. Prior studies have examined technical features of particular robotics simulators [25]–[27] but paid little attention to their role in quality assurance or the challenges developers face when using them. Instead, prior work on the challenges of testing in robotics [13], [28] and cyber-physical systems (CPSs) in general [12] broadly identifies simulation as a key

[1]The first two authors contributed equally to this work.

element of testing that requires improvement.

We conduct a study of robotics developers to understand how they perceive simulation-based testing and what challenges they face when using simulators. Through a survey of 82 robotics developers, we find that *simulation is used extensively for manual testing,* especially during early stages of design and development, but that *simulation is rarely used for automated testing.* By analyzing participant responses using grounded theory and other qualitative and quantitative methods, we identified 10 challenges that make it difficult for developers to use simulation in general (i.e., for any purpose), for testing, and specifically for automated testing. The challenges include a lack of realism, a lack of reproducibility, and the absence of automation features. The full list of challenges is presented in Section IV (Results) in Figure 5.

Study results can inform the construction of a new generation of software-based simulators, designed to better accommodate developers' needs for robotics testing. In particular, we show how several key barriers that impede or prevent simulation-based testing are incidental software engineering challenges, such as the need for languages and tools to construct test scenarios and environments. The software engineering and testing communities are well positioned to study and address these challenges of testability.

Overall, we make the following contributions:

- We conduct a study of 82 robotics developers from a variety of organizations and with diverse levels of experience in robotics.
- We find that developers are using simulation extensively for testing their robots and that many developers want to incorporate simulation into their test automation.
- We identify and explore ten key challenges that impede or prevent developers from using simulation in general and for manual and automated testing.
- We suggest ways in which the software engineering community can help to lower or eliminate barriers to simulation-based testing.
- We provide our survey materials and additional results to allow the community to build on our research at https://doi.org/10.5281/zenodo.4444256.

II. Robotics Testing and Simulation

Robots are CPSs that sense, process, and physically react to information from the real world [29]. The robot controller gets information about the environment through a collection of sensors. As Figure 2 illustrates, the robot controller continuously processes its sensor inputs and interacts with its environment through a series of actuators, thereby forming a control loop. End-to-end testing of robot controllers takes place in either a physical, real-world environment as part of field testing, or in a simulated environment in which the robot's sensors and actuators are replaced by virtual equivalents during simulation-based testing, as shown in Figure 1.

Field testing does not adhere to the testing paradigms that those in the software engineering community might expect. For example, a field tester might run the robot through various scenarios, sometimes loosely specified, that fall within the robot's expected operating situations and informally evaluate whether the robot does anything unexpected, often by visually inspecting the robot or its logs. Field testing is constrained by the physical robot hardware and environments available, and as a result is not scalable. It can also be extremely expensive and dangerous [11], [13], and is vulnerable to human error.

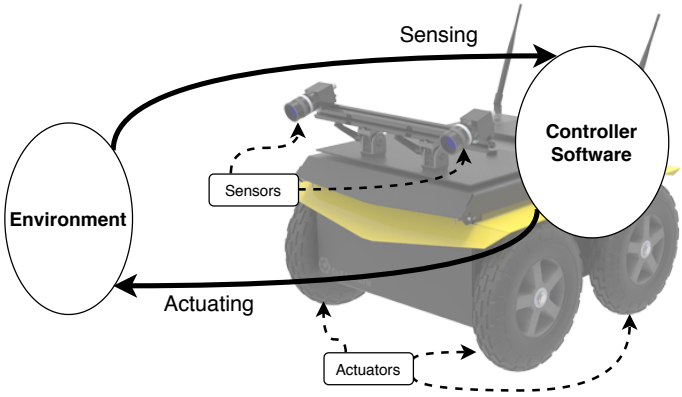

Fig. 2. The high-level architecture of a robotics system interacting with the environment, which may be physical or simulated. The controller software receives and processes input from its sensors (e.g., camera, GPS), and responds by actuating using its actuators (e.g., wheel motors).

In simulation, on the other hand, the robots' software stack and, optionally, the hardware platform, are replicated by connecting input-output channels (i.e., sensors and actuators) to a virtual robot embodied in a simulated world. Simulation-based testing encompasses a range of testing approaches conducted in the simulated environment, including everything from mimicking informal field testing in simulation, to creating a rigorous set of scenario inputs and using the simulator's logging outputs to determine whether the simulated robot adheres to specific expectations for its behavior in each scenario. Robotics testing approaches that have used simulation include: model-based testing [30], metamorphic testing [31], property-based testing [32], and robustness testing [33], [34]. Because robot simulation does not require robotics hardware, it has broader possibilities for use cases. For example, a simulated robot can be tested in an environment, such as the surface of Mars, that is unavailable to the tester in reality.

Simulation-based testing uses a virtual world produced by a simulator that models the physical aspects of the robot (e.g., kinematics), its operating environment (e.g., terrain, obstacles, lighting, weather), and the interaction between them. As illustrated in Figure 2, the simulator provides synthetic sensor readings to the robot controller at a fixed interval and listens to actuation signals from the robot controller (e.g., electronic speed control signals). The simulator performs a stepwise simulation of the virtual world that models certain physical interactions and phenomena at discrete time steps.

The two most popular forms of simulation-based testing are hardware-in-the-loop testing (HIL) and software-in-the-loop testing (SITL). During HIL, the robot controller software

runs on the robot's embedded hardware that is connected to a simulator that typically runs on a separate machine with greater resources (e.g., dedicated GPU). SITL, on the other hand, runs the robot controller software on the same machine as the simulator. HIL is typically slower and more expensive than SITL but can be used to test the integration between the robot controller's software and embedded hardware. For the purposes of this paper, any discussion of simulation-based testing includes both HIL and SITL, unless otherwise noted.

Numerous robotics simulators are available, each with different characteristics. General purpose simulators, such as Gazebo [35], CoppeliaSim (formerly known as V-REP) [36], Unity [37], and MuJoCo [38], can be used to model a wide range of systems. Others, such as CARLA [39], LGSVL [40], AirSim [41] and AADS [42], are specifically designed for a certain robotics domain. These simulators differ vastly in their capabilities, features, and performance [25]–[27], [43]. We do not focus on a particular simulator and instead study the challenges of using robotics simulators in general.

III. Methodology

We aim to better understand the ways in which robotics developers use simulation as part of testing processes, and the challenges they face in doing so. The following sections describe our methodology by presenting our research questions (Section III-A), survey design (Section III-B), participant recruitment (Section III-C), and analysis methods (Section III-D). Finally, we discuss some of the threats to the validity of our study and their mitigation (Section III-E).

A. Research Questions

To assess the ways in which robotics developers use simulation for testing, we first ask the following research question:

RQ1: To what extent do developers use simulation for testing and test automation?

Following the first research question, we focus on identifying the challenges robotics developers face when using simulation for testing. These challenges consist of both general limitations that impact all use-cases of simulation and challenges that specifically affect testing and test automation. Making the distinction between general limitations of simulators and testing-related challenges allows us to better understand and illustrate the issues that if resolved, can result in higher adoption of simulation for testing. As a result, we categorize the challenges of using simulation in three groups – general, testing-specific, and test-automation-specific challenges – in the following research questions:

RQ2: What challenges do developers face when using simulation in **general**?

RQ3: What challenges do developers face when using simulation for **testing**?

RQ4: What challenges do developers face when using simulation for **test automation**?

RQ1	• Have you ever used a software-based simulator? • For what purposes have you used software-based simulation?
RQ2	• Please tell us about how you used software-based simulation in [your latest] project? • Have you ever decided not to use software-based simulation for a project?
RQ3	• Which of these features are most useful when you use software-based simulation, specifically for testing?
RQ4	• How did you use software-based simulation as part of your test automation? • For what reasons, if any, have you not chosen to attempt to use software-based simulation for (partially) automated testing?

Fig. 3. Examples of survey questions separated by their corresponding research question (RQ). The full list of questions can be found at https://doi.org/10.5281/zenodo.4444256.

B. Survey Design

To answer our research questions, we conducted an online survey of robotics developers in November 2019. We followed best practices in survey design by explicitly breaking down the research questions into targeted survey questions, creating and pre-testing a pilot survey on a representative population of sample respondents, and making adjustments based on feedback until reaching saturation [44]–[46]. Examples of the survey questions are presented in Figure 3.

To ensure a meaningful interpretation of results, we provided our participants with a definition for the term "testing" as "any approach to finding errors in any part of a system (e.g., the software or hardware) by executing code used in the system in whole or in part, which can take place at any stage of system development and may be either automated or manual." Note that this definition is intentionally broader than that used in our previous study on the challenges of testing robotic systems, generally, which did not consider the manual use of simulation during the early stages of development to be a form of testing [13]. The full list of our survey questions, together with terminology and examples, are provided as part of the supplementary materials at https://doi.org/10.5281/zenodo.4444256.

C. Recruitment

To reach our intended audience (i.e., robotics developers), we distributed our survey via social media outlets, email, and several popular forums within the robotics community: the Robot Operating System (ROS) and Robotics subreddits on Reddit,[1] the ROS Discourse,[2] and the RoboCup forums.[3] We decided to advertise our survey to the ROS community as ROS is a popular and widely used robotics software framework [47], [48]. We also advertised our survey on Facebook and Twitter

[1]https://reddit.com
[2]https://discourse.ros.org
[3]http://lists.robocup.org/cgi-bin/mailman/listinfo

and posted a recruitment email to mailing lists for a university robotics department and a robotics research institution.

In total, 151 participants took the survey, out of which 82 completed it. For the purpose of analysis, we only consider the 82 completed responses. All 82 participants who completed the survey reported that they had used a robotics simulator. Figure 4 presents the demographics of these 82 participants. In terms of experience, more than two thirds of participants (71.95%) reported having worked with robotics software for more than three years. Most participants (79.27%) reported that they had worked with robotics in academia at some point during their life, and almost two thirds (65.85%) reported working with robotics in industry at some point. Participants reported that they currently work at organizations of varying sizes. Overall, our study sample is composed of a diverse array of candidates with differing levels of experience who have worked in a variety of organizations, thus ensuring that the results of the study are not limited to any one population.

D. Analysis

Our survey includes both quantitative (closed-ended) and qualitative (open-ended) questions. To analyze the open-ended responses, we used descriptive coding [49] to assign one or more short labels, known as *codes*, to each data segment (i.e., a participant's response to a given question), identifying the topic(s) of that segment. After developing an initial set of codes, we adjudicated to reach consistency and agreement, then used code mapping to organize the codes into larger categories [49]–[51]. Using the explicit mapping from survey questions to research questions, devised during survey design, we aggregated the set of relevant categories for each research question. Finally, we used axial coding to examine relationships among categories and identify a small number of overarching themes for each research question.

Throughout this paper, we present quotes directly taken from participant responses. We have fixed minor grammatical and spelling mistakes for better readability. We refer to each participant by a unique number (e.g., P12), for anonymity.

E. Threats to Validity

To mitigate the threat of asking the wrong questions and introducing bias in the wording of questions, we followed survey design best practices [44]–[46], such as the use of iterative pilots, and included a number of open-ended questions to allow participants to freely discuss topics of concern to them.

Our analysis of open-ended survey responses is a potential threat to internal validity. To mitigate this concern, we followed established guidelines on qualitative and quantitative studies [49]–[51]. As a post-study validation, we shared the results and conclusions on several public platforms, including those used for recruitment (Section III-C), and received positive feedback on our findings from the robotics community.

Although we received many responses from robotics developers, we cannot make broad claims on the generalizability of our findings. To mitigate this threat, we distributed the survey among robotics developers from different backgrounds and organizations by targeting popular robotics platforms.

To promote further research, we share our recruitment materials, questionnaire, codebook, and additional results at the following URL: https://doi.org/10.5281/zenodo.4444256.

IV. Results

In this section, we present the results of our study of robotics developers on their use of simulation for testing, and the challenges they face along the way. In Section IV-A, we discuss the extent to which developers use simulation for testing. We then present and analyze the challenges of using simulators, summarized in Figure 5. Section IV-B discusses challenges that apply broadly to many uses of simulation, including design and development, in addition to testing. Section IV-C narrows the focus to challenges that apply when simulation is particularly used for testing, and Section IV-D further narrows to the challenges specific to test automation.

A. RQ1: To what extent do developers use simulation for testing and test automation?

Our survey asked participants about their use of simulation: both broadly and specifically for testing robotics systems. We find that our participants are unanimously familiar with simulation, and they use it on a regular basis. 59 out of 82 (71.95%) participants reported that they used simulation within the last month at the time of completing the survey. When asked about their most recent project that involved simulation, 51 of 82 (62.20%) participants reported that they used a simulator daily, and 25 of 82 (30.49%) participants reported that they used a simulator on a weekly basis.

Figure 6 presents the variety and popularity of purposes for which our participants use simulation. Almost 84% of participants have used simulation for testing, and testing is the most popular use case for simulation. This suggests that developers generally see value in using simulation for testing.

Participants reported using simulation for various forms of testing, including: testing the underlying algorithms; variability testing (e.g., testing the robot with different components); sanity checking (e.g., checking software in simulation before deploying it to the hardware); and multi-robot testing (e.g., simulating how robots will interact with each other).

Participants also reported a variety of reasons for using simulation for testing. These include when it is unsuitable or impractical to test on real hardware or in a real environment. They also reported using simulation to better understand the design and behavior of existing robotic systems and their associated software, and to incorporate simulation into automated robotics testing, including continuous integration (CI).

Of the 84% of participants who have used simulation for testing, we find that 61% of them have also tried to use simulation as part of their test automation. These findings demonstrate that developers find simulation to be a valuable tool for testing, and there is a desire to incorporate simulation-based testing into their test automation processes.

Experience			**Organization**			**Size of organization**		
Years of experience	#	%	Type	#	%	Number of people	#	%
Less than one year	10	12.20%	Academia	65	79.27%	1–10 people	22	26.83%
Between one and three years	13	15.85%	Industry	54	65.85%	11–50 people	23	28.05%
Between three and ten years	40	48.78%	Individual	35	42.68%	51–100 people	9	10.98%
More than ten years	19	23.17%	Government	12	14.63%	More than 100 people	28	34.15%
			Other	9	10.98%			

Fig. 4. Demographics for the 82 survey participants that completed the survey in terms of their experience, the types of organization at which they had worked, and the size of the most recent organization to which they belonged.

These results motivate the rest of our study, which looks at the challenges robotics developers face using simulation. Given the ubiquity of simulation and its importance to robotics testing and development, there is great potential benefit from lowering the barriers to using simulation, especially for testing. We highlight these barriers to direct attention to areas in which improvements to simulators may have the most impact, thereby allowing developers to advance the state of software engineering and quality assurance in robotics.

Key Insight: Simulation is an essential tool for developers that is used extensively for building and testing robot software. Given its importance, it is vital that we better understand the challenges that prevent developers from realizing its full potential.

B. RQ2: What challenges do developers face when using simulation ***in general?***

Although we find that simulation is popular among developers, 28 of 82 (34.15%) participants reported that there was a project for which they decided to not use simulation. Their reported reasons are given in Figure 7. By analyzing these reasons along with the difficulties that participants experienced when they did use simulation, we identified three high-level challenges of using simulation in general, discussed below.

Reality gap*:* Simulation, by definition, creates an abstraction of the real world and the robotics hardware in it. As a result, it can never be 100% accurate in representing all aspects of the real environment. The sometimes inadequate representation of physical reality in simulation is known colloquially as the *reality gap*. Many participants cited the reality gap both as a challenge when trying to use simulation and a reason not to use it in the first place. P33 notes that simulation can produce unrealistic behaviors that would not occur in the real world. P16 highlighted that accounting for all relevant physical phenomena can also be challenging: "my simple simulation model did not include a tire model, so simulations at higher speeds did not account for realistic behaviors for cornering or higher accelerations or deceleration." In particular, realistically modeling stochastic processes, such as signal noise, and integrating those models into the simulation as a whole is a challenge: P15 shared, "A classic problem is integrating wireless network simulation with physical terrain simulation. This also applies to GPS signal simulation, as well."

For some, such as P29, the reality gap can be too large to make simulation valuable: "too big discrepancy between simulation results and reality (physical interaction)." For others, simulation can still serve as a valuable tool despite the existence of the reality gap. As P36 puts it, "Software behavior in simulation is different compared to [the] real [world], so not everything can be tested, but a lot can be." In talking about the reality gap, respondents faulted both the limitations of the modeling formats and the limitations of the simulators.

Complexity*:* Accurate simulation of the physical world is inherently challenging and involves composing various models. Alongside the essential complexity of simulation are sources of *accidental complexity* [52] that do not relate to the fundamental challenges of simulation itself, but rather the engineering difficulties faced when trying to use simulation. For example, a lack of user-friendly features is a source of accidental complexity. Sources of accidental complexity may ultimately lead users to abandon or not use simulation at all.

Inaccurate, inadequate, or missing documentation can make it difficult to learn and use a simulator. P22 highlights that a "lack of documents for different platform types and sometimes wrong documentation makes us lose a lot of time working on [stuff] that will never work, for example, the Gazebo simulator does not work well in Windows." A language barrier may cause documentation to be inaccessible, as reported by P74: "The language was Japanese, but we don't speak that language so we couldn't use well the simulator."

Difficult-to-use application programming interfaces (APIs) make it difficult to extend the simulator with new plugins. P4 points out that "Gazebo is the de-facto [simulator] right now and is poorly documented and difficult to customize to any degree." A lack of integration with popular computer aided design (CAD) software (e.g., AutoCAD, SolidWorks) and support for industry-standard 3D modeling formats (e.g., IFC), makes it difficult to import existing, high-quality models.

Together, these sources of complexity increase simulators' learning curve and may lead developers to abandon or never start to use them. P20 shared that there is a "steep learning curve in understanding the test environment software setup and libraries. Without a good software engineering skills the simulated environment will not replicate the real environment."

	Challenge	Description	Representative quote
■	*Reality gap*	The simulator does not sufficiently replicate the real-world behavior of the robot to a degree that is useful.	"[Simulation is] not realistic enough for accurately modeling task; preferred running on real robot" – P33
■	*Complexity*	The time and resources required to setup a sufficiently accurate, useful simulator could be better spent on other activities.	"It was easier and more accurate to setup and test on a physical system than simulate" – P4
■	*Lacking capabilities*	Simulators may not possess all of the capabilities that users desire, or those simulators that do may be prohibitively expensive.	"Most simulators are good at one thing, some are good at simulating the vehicles (drone,robot,car,etc) some are good at simulating the environment (good for generating synthetic data) some are good at senors, some are good at physics, some are good at pid control, etc. but not one has all these attributes." – P77
▲	*Reproducibility*	Simulations are non-deterministic, making it difficult to repeat simulations, recreate issues encountered in simulation or on real hardware, and track down problems.	"Deterministic execution: the same starting conditions must produce absolutely identical results." – P42
▲	*Scenario and environment construction*	It is difficult to create the scenarios and environments required for testing the system in simulation.	"Setting up a simulation environment is too much work, so I don't do it often." – P38
▲	*Resource costs*	The computational overhead of simulation requires special hardware and computing resources which adds to the financial cost of testing.	"Simulating multiple cameras (vision sensors) with full resolution at a high frame rate is usually very slow and therefore not practical." – P37
★	*Automation features*	The simulator is not designed to be used for automated testing and does not allow headless, scripted or parallel execution.	"Most simulations are NOT designed to run headless, nor are they easily scriptable for automatic invocation." – P34
★	*Continuous integration*	It is difficult to deploy the simulator in suitable environments for continuous integration (e.g., cloud computing servers).	"The simulation requires some computational resources which can be difficult to be part of CI, especially when our CI is running on the cloud" – P62
★	*Simulator reliability*	The simulation is not reliable enough to be used in test automation in terms of the stability of the simulator software, and the timing and synchronization issues introduced by the simulator.	"There were many challenges - 1. Getting difference in the real time and simulation time 2. Changing the entire physics engine source code for our application 3. Glitch during the process of trying to move the real hardware with the simulation model." – P80
★	*Interface stability*	The simulator's interface is not stable enough or sufficiently well-documented to work with existing code or testing pipelines.	"[We have automation difficulties with] integration into existing code, missing APIs, stability of libraries" – P28

Fig. 5. Summary of challenges participants encountered when using simulation in general (■), specifically for testing (▲), and for test automation (★).

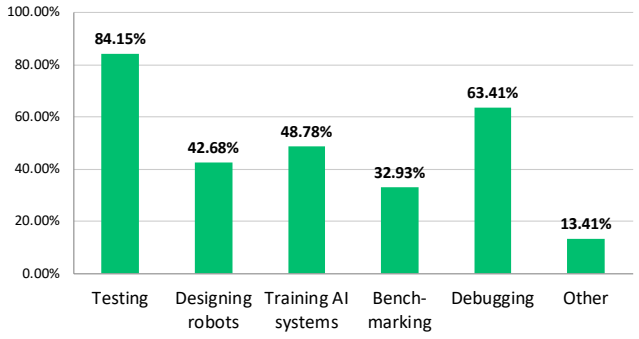

Fig. 6. An overview of the high-level reasons that participants gave for using simulation (82 responses).

Lacking capabilities: Finding a simulator that provides all of the characteristics a user desires can be challenging. P77 highlighted that, while it is possible to find a simulator that is good in one particular aspect, it is hard to find a simulator that is good in all desired aspects. As P4 pointed out, simulators that do possess all of the desired qualities tend to be expensive: "Adding plugins is usually very challenging, and the only good frameworks that do any of this stuff well are very expensive (V-Rep and Mujoco for example)."[4]

We asked which simulation features participants wanted most but are unable to use in their current setups. Among the most important features mentioned were the ability to simulate

[4]Coppelia Robotics informed the authors that V-REP has been re-branded as CoppeliaSim and is free for educational and non-commercial applications.

Reason for not using simulation	#	%
Lack of time or resources	15	53.57%
Not realistic/accurate enough	15	53.57%
Lack of expertise or knowledge on how to use software-based simulation	6	21.43%
There was no simulator for the robot	4	14.29%
Not applicable	4	14.29%
Too much time or compute resources	2	7.14%
Nobody suggested it	0	0.00%
Other	2	7.14%

Fig. 7. An overview of the reasons that participants gave for not using simulation for a particular project, based on 28 responses.

at faster-than-real-time speeds (i.e., where the simulation clock runs faster than the wall clock), native support for headless (i.e., without the graphical user interface (GUI)) execution (discussed in Section IV-D), and an easier means of constructing environments and scenarios (discussed in Section IV-C).

Numerous participants wanted the ability to run simulation at faster-than-real-time speeds but were unable to do so in their current simulation setups. For example, P52 said, "We needed to speed up simulation time, but that was difficult to achieve without breaking the stability of the physics engine." This feature is useful not only for reducing the wall-clock time taken to perform testing, but for other purposes, as P62 highlighted: "Faster than real time is really important to produce training data for deep learning."

Several participants also desired features that would increase simulation fidelity (i.e., how closely the simulation mimics reality). P46 wanted support for "advanced materials in environments (custom fluids, deformable containers, etc.)." Interestingly, P69 desired the ability to tune the fidelity of the simulation: "Ability for controllable physics fidelity. First order to prove concepts then higher fidelity for validation. Gazebo doesn't have that." Recent studies have shown that low-fidelity simulation can effectively and inexpensively discover many bugs in a resource-limited environment [14]–[16].

Participants also specified other capabilities such as native support for multi-robot simulation and large environments, and support for efficiently distributing simulation computations across multiple machines.

Ultimately, the complexities of setting up and using simulation, the reality gap, and the time and resources necessary to make the simulation useful led some participants to use physical hardware instead. As P4 said, "It was easier and more accurate to setup and test on a physical system than simulate."

Key Insight: Developers find considerable value in simulation, but difficulties of learning and using simulators, combined with a lack of realism and specific capabilities, constrain the way that developers use simulation. By alleviating these challenges, simulation can be used for a wider set of domains and applications.

*C. RQ3: What challenges do developers face when using simulation for **testing**?*

Participants reported a variety of challenges in attempts to use simulation for testing, summarized in Figure 5. We identified the following challenges that mainly affect the use of simulation for testing:

Reproducibility: From inevitable sensor and actuator inaccuracies to stochastic algorithms (e.g., vision and navigation), robotics systems have numerous, inherent sources of nondeterminism. Temporal effects (e.g., message timing and ordering) can also lead to different outcomes. A deterministic simulator should be able to simulate nondeterministic factors but also allow reproducibility, in which it repeats a given simulation with all of these factors behaving in the same manner. For example, a deterministic simulator should be able to execute two simulations of the same scenario, producing the exact same messages in the exact same order. Such a simulator would allow for consistent testing and fault diagnosis.

The lack of reproducibility and presence of nondeterminism in simulators lead to difficulties when testing, as reported by participants. P42 highlighted that a "Lack of deterministic execution of simulators leads to unrepeatable results." This points to a need to accurately reproduce system failures that are discovered in testing, in order to diagnose and debug those failures. If a tester cannot consistently reproduce the failures detected in simulation, it will be difficult to know whether changes made to the code have fixed the problems. P7 pointed to the particular difficulty with achieving reproducibility in Gazebo: "Resetting gazebo simulations was not repeatable enough to get good data." P48 and P81 also mentioned a desire for reproducibility.

Consistent and systematic testing often relies on deterministic test outcomes, particularly when incorporating test automation and continuous integration tests, which rely on automatically detecting when a test has failed.Flaky [53] and non-deterministic tests may lead to a false conclusion that a problematic software change does not have a problem (a false negative) or that a good change has problems (a false positive).

Scenario and environment construction: Testing in simulation requires a simulated environment and a test scenario, which can be understood as a set of instructions for the robot under test. Participants reported difficulty in constructing both environments and test scenarios. P38 said: "Setting up a simulation environment is too much work, so I don't do it often," and P3 contributed, "Scripting scenarios was not easy. Adding different robot dynamics was also not easy." They wanted to be able to construct these more easily or automatically. Participants pointed out that the scenarios or environments they require sometimes must be created "by hand," which requires a heavy time investment and is subject to inaccuracies. In recent years, high-level languages have been proposed to aid the construction of rich driving scenes in simulation [54]–[56]. P4 said, "Making URDF[5] files is a

[5]Unified Robot Description Format (URDF) is an XML file format used in robotics platforms to describe all elements of a robot.

tremendous pain as the only good way to do it right now is by hand which is faulty and error prone," while P67 wanted, "Automated generation of simulation environments under some [custom] defined standards," because "The automated simulation environment generation is not easy. Plenty of handy work must be done by human operators."

Resource costs: Simulation is computationally intensive. It often benefits from specialized hardware, such as GPUs. Participants report that hardware requirements contribute strongly to the expense of simulation. These costs are compounded when tests are run many times, such as in test automation. For example, P42 reported that difficulties including simulation in test automation include: "High hardware requirements (especially GPU-accelerated simulators) driving high cloud server costs." Participants reported problems with running simulations in parallel or using distributed computing across several machines. Participants also reported challenges in simulating large environments and simulations of long duration, as they became too resource-demanding to be practical. P67 requested, "High computational performance when the environment size grows large (Gazebo performance drops down rapidly when the number of models raises)." Participants also had issues with the cost of licenses for appropriate simulators. P66 reported that cost drove the choice not to use a given simulator: "Back then, Webots was not free," and P1 complained: "Not to mention the licensing price for small companies."

Key Insight: 84% of participants used simulation for testing, but a lack of reproducibility, the complexities of scenario and environment construction, and considerable resource costs limit the extent of such testing.

*D. RQ4: What challenges do developers face when using simulation for **test automation**?*

Research has shown that test automation can provide many benefits, including cost savings and higher software quality [57]. Despite the benefits of test automation, 27 of 69 (39.13%) participants who have used simulation for testing, reported never attempting to use simulation for automated testing. Responses indicated that the challenges with using simulation, both in general and for testing, prevented participants from attempting to incorporate it into test automation. Their reasons fell into three general categories:

1) Lack of value, where they did not find test automation valuable or necessary. As P24 mentioned "There were no obvious test harnesses available for the simulation environments I use and it did not seem obviously valuable enough to implement myself."
2) Distrust of simulation, in which the limitations of simulation itself drove the decision to not use it in automated testing. *Reality gap* and *lacking capabilities*, discussed in Section IV-B, contribute to this belief. P33 mentioned, "[Simulation is] not realistic enough for accurately modeling task; preferred running on real robot," and P20 believed that "Without a good software engineering skills the simulated environment will not replicate the real environment."
3) Time and resource limitations, where the complexity of the simulator (Section IV-B) and resource costs (Section IV-C) prevented them from attempting test automation. P18 explained "[We did not attempt to use software-based simulation for automated testing] due to the amount of time needed to setup the automated testing. Even if I think on the long term it cuts down development time, my company does not allocate time for that." and P17 simply reported that "[it] seemed very hard to do."

Among 42 people who attempted to use simulation as part of their test automation, 33 (78.57%) reported difficulties. Based on their descriptions of these difficulties, we identified the following four challenges specifically affecting test automation:

Automation features: Although a GUI is an important component of a simulator, participants reported a preference towards running the simulator headless (i.e., without the GUI) when used for test automation. Disabling the GUI should reduce the computational overhead of the simulator by avoiding the need to render computation-heavy graphical models. Not being able to run a simulator headless is one of the major difficulties our participants faced in automation.

> "Making the simulator run without GUI on our Jenkins server turned out to be more difficult than expected. We ended up having to connect a physical display to the server machine in order to run the simulation properly." – P37

Furthermore, the ability to set up, monitor, and interact with a simulation via scripting, without the need for manual intervention, is vital for automation. Our participants reported devising creative solutions in the absence of scripting support. P8 shared, "Ursim[6] needs click-automation to run without human interaction." In other words, they needed to use a tool that simulated mouse clicks to run the simulator automatically.

Continuous integration (CI): CI is emerging as one of the most successful techniques in automated software maintenance. CI systems can automate the building, testing, and deployment of software. Research has shown that CI practices have a positive effect on software quality and productivity [58].

CI, by definition, is an automated method, and in many cases involves the use of cloud services such as TravisCI. Our participants faced difficulties engineering the simulation to be used in CI and run on cloud servers. For example, P66 shared "I wasn't able to setup a CI pipeline which runs in GPU machines, for use with rendering sensors."

Many of these difficulties arise from missing automation features (e.g., headless execution) and high resource costs (e.g., requiring expensive, GPU-heavy hardware), discussed earlier. P77 reported "It is expensive to spin up cloud GPU VMs to run the simulator."

Simulator reliability: One of the challenges of using a simulator in a test automation pipeline is the reliability of the simulator itself. In other words, participants reported facing

[6]Universal Robots Simulator https://www.universal-robots.com

unexpected crashes, and timing and synchronization issues while using the simulator in automation. Robotics systems, as timing-sensitive CPSs, can have their behavior distorted when timing is distorted, such as when messages arrive out-of-order or are dropped or deadlines are missed. P29, P54, P73, and P80 all reported software stability and timing issues as difficulties they faced for automation. P29 further elaborated difficulty in ensuring a clean termination of the simulator. That is, when the simulator crashes, it should properly store the logs and results before termination of the simulation, and properly kill all processes to prevent resource leaks. Clean termination is particularly relevant to test automation. Resource leaks can compound when simulations are repeated, as they are in automated testing. Compounded resource leaks can reach the point where they interfere with the ability to run additional simulations and require manual intervention.

***Interface stability**:* The stability of the simulator interface can significantly impact the automation process because inconsistent simulator APIs can lead to failures in client applications [59]. Participants reported unstable and fragile interfaces as a challenge for automation. For example, P39 mentioned "APIs are pretty fragile and a lot of engineering need to be done to get it working."

Five participants reported difficulties in integrating existing code or infrastructure with simulation APIs. P80 mentioned changing the entire physics engine source code for an application. Participants specifically desired better integration with the Robot Operating System (ROS). For example, P74 shared "I would like that [the simulator] can be used with ROS."

Key Insight: Developers want to include simulation as part of their test automation, but most developers who attempt to do so face face numerous difficulties. These difficulties include an absence of automation features and a lack of reliability and API stability. Ultimately, these challenges discourage developers from using simulation for test automation, limit the extent to which it is used, and prevent developers from leveraging the benefits of continuous integration.

V. Discussion

As robots and their associated codebases become larger and more complex, the need for, and cost of, continuous verification and validation will increase considerably. While field testing is popular it will be unable to handle the increased needs by itself because it is limited in practice by expense, hardware, human resources, and safety [13]. Simulation-based testing may serve as a cheaper, safer, and more reliable alternative by offering a means of scalable test automation [57]. Indeed, 61% of our survey participants reported that they had attempted to use simulation as part of test automation, indicating that practitioners have a strong interest in simulation-based testing.

Despite widespread interest, a multitude of challenges prevent developers from readily enjoying the benefits of simulation-based testing. A number of these challenges, such as the need for sufficient physical fidelity (i.e., the reality gap), are *inherent challenges* of building a simulator. Such challenges are well studied and widely recognized by the robotics and simulation community, and therefore outside the purview of our recommendations. While the *reality gap* can never fully be solved, the robotics and simulation community can study the trade offs inherent in usefully modeling the aspects that are important to simulating relevant systems.

However, there are ***incidental challenges of testability***, which are understudied but just as important as the inherent challenges and also impede effective simulation-based testing. These challenges include, but are not limited to, the ability to reliably run simulations without the expensive and unnecessary overhead of visualization (i.e., headless execution); the need for a powerful, expressive language for constructing realistic environments and scenarios; the ability to set up, monitor, and interact with the simulation via scripting; and stability in the simulator's client interface. Interestingly, when participants were asked to choose the simulator features that they found most useful for testing, features related to testability (e.g., "custom plugins", "exposed APIs", and "recording and logging") appeared as more popular than features related to the underlying simulation (e.g., "advanced graphics" and "high-performance physics engines").[7] Addressing these incidental challenges can enable robotics developers to better take advantage of automated simulation-based testing.

Addressing the incidental challenges requires varying levels of effort and expertise. Certain challenges can be largely fixed with engineering effort and application of best practices from the software engineering community. For example, *automation features* (e.g., headless operation), *continuous integration* support, and *simulator reliability* are all mainly engineering challenges. *Scenario construction* can be addressed through the development and application of domain-specific languages, discussed in Section VI.

In other cases, the challenges can be addressed but are subject to certain limitations. For example, while it may not be realistic to expect indefinite *interface stability* from software that must change to fit evolving needs, some problems may be ameliorated by following good API design and documentation best practices [60] and engineering for backwards compatibility. Challenges in this category also include *lacking capabilities*, *reproducibility*, *resource costs*, and *complexity*. For *lacking capabilities*, it is unlikely that one simulator would provide all capabilities needed for every possible use case, but it is more realistic to engineer simulators that are extensible or tailored for the capabilities needed for individual use cases. While it would be possible to control some factors to create *reproducibility*, there is always a trade off against realistically modeling relevant nondeterministic phenomena. For *resource costs*, there is a trade off between the desired property of high fidelity and the corresponding resource cost. Because of the inherent complexity of the tasks needed in simulation, there

[7]The full list of these features and their ranking according to the participants can be found as part of our supplementary material.

will always be a degree of *complexity* in learning the simulator. The complexity and learning curve can be lessened with good design and documentation but will never disappear entirely.

From participant responses and our own experience, we observe that most popular simulation platforms (e.g., Gazebo) are predominantly designed to support manual testing during the earlier stages of robot development; test automation, on the other hand, does not appear to have been considered as an explicit use case by popular simulation platforms. To support scalable automation of simulation-based testing as part of continuous process of verification and validation, simulators should address the incidental challenges discussed earlier. We believe that the software engineering and testing communities are well-equipped to address these incidental challenges as they have studied similar problems in other domains such as distributed systems [61]–[64]. We call upon these communities to work alongside robotics and simulator developers to study and address these incidental testability challenges.

VI. Related Work

Simulator Comparisons: As mentioned earlier, there are numerous robotics simulators which have different characteristics, such as: paid or free, general-purpose or domain-specific [35]–[42], [65]–[67]. Researchers and end users have long evaluated and compared these simulators based on their features, capabilities in representing the real world, and performance [25]–[27], [43], [68], [69]. In one such study, de Melo et al. compare the three most popular general-purpose simulators, by replicating the same standard 3D scene in each simulator [26]. Staranowicz and Mariottini present a detailed overview and comparison of eleven commercial and open-source robotics simulators [25]. Similarly, Erez et al. compare five robotics simulators and find that each performs best on the type of system it was designed and optimized for [43].

Our paper differs from the aforementioned studies. Rather than comparing a set of robotics simulators, our paper broadly investigates the challenges of using simulators, with a focus on using simulators for manual and automated testing. Unlike prior studies, this investigation is conducted empirically.

Simulator Development: There is continued development and improvement of simulators. While some of these new developments address issues this paper identifies, considerable room for improvement remains. Several recent simulators are specialized for particular domains: CARLA [39], LGSVL [40], and AADS [42] are for automated driving applications, while AirSim [41] simulates a wider variety of autonomous vehicles. Notably, these simulators are built on top of popular video game engines and support complex, dynamic urban environments. AADS [42] enhances visual fidelity, allowing for more realistic perception component testing. By contrast, Ignition Gazebo [65], the descendant of the Gazebo simulator, is agnostic to application and domain and supports various rendering and physics backends, allowing customization (e.g., optimizing for fidelity or performance). AWS RoboMaker [66] is a web-based IDE, simulator, and fleet management front-end designed for easier development, testing, and deployment. RoboMaker internally builds on top of Gazebo by adding infrastructure for parallel simulations and automatic hardware scaling, and providing prebuilt environments. Although each simulator addresses at least one of the identified challenges, it is as yet unclear whether they address enough of developers' needs in the right combinations to succeed.

Testing Robotics Systems: Testing robotics systems presents unique problems. Beschastnikh et al. outline some of the challenges and drawbacks to existing approaches to debugging distributed systems, such as robotics systems [70]. Zheng et al. point out the challenges of applying verification to CPSs [12], [71]. Several approaches have addressed aspects of the problems in testing these systems. Sotiropoulos et al. motivate testing robotics in simulation and demonstrate the approach's effectiveness in some domains [14]. Tuncali et al. define an approach that relies on well-defined system requirements, which are absent in many systems [20]. Timperley et al. attempt to categorize real bugs reported in the ArduPilot autonomous vehicle software as to whether they can be reproduced and/or detected in simulation [15].

Recent work on testing in simulation for robotics systems looks at a range of issues, including: simulating the human experience of being in a self-driving car [72]; accurately simulating vehicle hardware and its interaction with the environment [73], [74]; and automatically generating test scenarios for the vehicles in simulation [18], [75], [76].

As part of a study to determine the general challenges of testing robotic systems, Afzal et al. also find that few developers use simulation for test automation [13]. In this study, we investigate the challenges that deter developers from using simulation for test automation.

VII. Conclusion

In this paper, we conducted a study of 82 robotics developers to explore how robotics simulators are used and the challenges that developers commonly face when using simulation for general purposes, testing, and test automation. Our results indicate that simulation is a popular tool among robotics developers and is commonly used for testing with 84% of participants reporting having used simulation for testing, 61% of whom have also used simulation as part of their test automation. We identified 10 high-level challenges associated with using simulation, and discussed these challenges in detail. We further outlined ideas on how the software engineering and testing communities can tackle these challenges to unlock the full potential of simulation-based testing.

Acknowledgment

We would like to thank both the ROS and Reddit communities, and in particular, Chris Volkoff and Olly Smith, for their invaluable support in distributing our survey. We also thank the ICST reviewers for their particularly insightful comments. This research was partially funded by AFRL (#OSR-4066), 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.

REFERENCES

[1] R. Cellan-Jones, "Robots 'to replace up to 20 million factory jobs' by 2030," *BBC News*. [Online]. Available: https://www.bbc.com/news/business-48760799

[2] R. Heilweil, "Networks of self-driving trucks are becoming a reality in the US," *Vox*. [Online]. Available: https://www.vox.com/recode/2020/7/1/21308539/self-driving-autonomous-trucks-ups-freight-network

[3] Hyperdrive, "The state of the self-driving car race 2020," *Bloomberg*. [Online]. Available: https://www.bloomberg.com/features/2020-self-driving-car-race

[4] D. Berreby, "The pandemic has been good for one kind of worker: robots," *National Geographic*. [Online]. Available: https://www.nationalgeographic.com/science/2020/09/how-pandemic-is-good-for-robots

[5] Z. Thomas, "Coronavirus: Will covid-19 speed up the use of robots to replace human workers?" *BBC News*. [Online]. Available: https://www.bbc.com/news/technology-52340651

[6] N. Statt, "Boston Dynamics' Spot robot is helping hospitals remotely treat coronavirus patients," *The Verge*. [Online]. Available: https://www.theverge.com/2020/4/23/21231855/boston-dynamics-spot-robot-covid-19-coronavirus-telemedicine

[7] M. Belko, "Airport using robots, UV light to combat COVID-19," *Pittsburgh Post-Gazette*. [Online]. Available: https://www.post-gazette.com/business/development/2020/05/05/Pittsburgh-International-Airport-COVID-19-Carnegie-Robotics/stories/202005050120

[8] S. O'Kane, "Boeing finds another software problem on the 737 Max," *The Verge*. [Online]. Available: https://www.theverge.com/2020/2/6/21126364/boeing-737-max-software-glitch-flaw-problem

[9] M. Wall, "European Mars lander crashed due to data glitch, ESA concludes," *Space*. [Online]. Available: https://www.space.com/37015-schiaparelli-mars-lander-crash-investigation-complete.html

[10] R. N. Charette, "Nissan recalls nearly 1 million cars for air bag software fix," *IEEE Spectrum*. [Online]. Available: https://spectrum.ieee.org/riskfactor/transportation/safety/nissan-recalls-nearly-1-million-cars-for-airbag-software-fix

[11] P. McCausland, "Self-driving Uber car that hit and killed woman did not recognize that pedestrians jaywalk," *NBC News*. [Online]. Available: https://www.nbcnews.com/tech/tech-news/self-driving-uber-car-hit-killed-woman-did-not-recognize-n1079281

[12] X. Zheng, C. Julien, M. Kim, and S. Khurshid, "Perceptions on the state of the art in verification and validation in cyber-physical systems," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2614–2627, Dec 2017.

[13] A. Afzal, C. Le Goues, M. Hilton, and C. S. Timperley, "A study on challenges of testing robotic systems," ser. ICST '20, 2020, pp. 96–107.

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

[15] 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, Validation, and Verification*, ser. ICST '18, 2018, pp. 331–342.

[16] C. Robert, T. Sotiropoulos, J. Guiochet, H. Waeselynck, and S. Vernhes, "The virtual lands of Oz: testing an agribot in simulation," *Empirical Software Engineering*, 2020.

[17] C. Gladisch, T. Heinz, C. Heinzemann, J. Oehlerking, A. von Vietinghoff, and T. Pfitzer, "Experience paper: Search-based testing in automated driving control applications," in *Automated Software Engineering*, ser. ASE '19, 2019, pp. 26–37.

[18] 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.

[19] G. E. Mullins, P. G. Stankiewicz, and S. K. Gupta, "Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles," in *International Conference on Robotics and Automation*, ser. ICRA '17, 2017, pp. 1443–1450.

[20] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles," in *Intelligent Transportation Systems*, ser. ITSC '16, 2016, pp. 1470–1475.

[21] E. Rocklage, H. Kraft, A. Karatas, and J. Seewig, "Automated scenario generation for regression testing of autonomous vehicles," in *International Conference on Intelligent Transportation Systems*, ser. ITSC '17, 2017, pp. 476–483.

[22] Uber, "Self-Driving Simulation." [Online]. Available: https://www.uber.com/us/en/atg/research-and-development/simulation

[23] NVIDIA, "NVIDIA DRIVE Constellation: Virtual reality autonomous vehicle simulator." [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/drive-constellation

[24] Waymo, "Waymo safety report: On the road to fully self-driving," 2018. [Online]. Available: https://waymo.com/safety

[25] A. Staranowicz and G. L. Mariottini, "A survey and comparison of commercial and open-source robotic simulator software," in *Pervasive Technologies Related to Assitive Environments*, ser. PETRA '11, 2011, pp. 56:1–56:8.

[26] M. S. P. de Melo, J. G. da Silva Neto, P. J. L. da Silva, J. M. X. N. Teixeira, and V. Teichrieb, "Analysis and comparison of robotics 3D simulators," in *Symposium on Virtual and Augmented Reality*, ser. SVR '19, 2019, pp. 242–251.

[27] L. Pitonakova, M. Giuliani, A. Pipe, and A. Winfield, "Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators," in *Annual Conference Towards Autonomous Robotic Systems*, 2018, pp. 357–368.

[28] 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, pp. 1–41, 2019.

[29] C. S. R. at Max Planck Institutes, "Robotics and Cyber-Physical Systems." [Online]. Available: https://www.cis.mpg.de/robotics

[30] G. Kanter and J. Vain, "TestIt: an open-source scalable long-term autonomy testing toolkit for ROS," in *Dependable Systems, Services and Technologies*, ser. DESSERT '19, 2019, pp. 45–50.

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

[32] A. Santos, A. Cunha, and N. Macedo, "Property-based testing for the Robot Operating System," in *Automating TEST Case Design, Selection, and Evaluation*, ser. A-TEST '18, 2018, pp. 56–62.

[33] 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. ICSE-SEIP '18, 2018, pp. 276–285.

[34] D. S. Katz, M. Zizyte, C. Hutchison, D. Guttendorf, P. E. Lanigan, E. Sample, P. Koopman, M. Wagner, and C. Le Goues, "Robustness inside out testing," in *Dependable Systems and Networks – Industry Track*, ser. DSN-I, 2020, pp. 1–4.

[35] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *International Conference on Intelligent Robots and Systems*, ser. IROS '04, vol. 3, 2004, pp. 2149–2154.

[36] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Intelligent Robots and Systems*, ser. IROS '13, 2013, pp. 1321–1326.

[37] A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, 2018.

[38] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[39] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, "CARLA: An open urban driving simulator," in *Conference on Robot Learning*, ser. CoRL, 2017, pp. 1–16.

[40] LG, "LGSVL Simulator." [Online]. Available: https://www.lgsvlsimulator.com

[41] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, M. Hutter and R. Siegwart, Eds., 2018, pp. 621–635.

[42] W. Li, C. W. Pan, R. Zhang, J. P. Ren, Y. X. Ma, J. Fang, F. L. Yan, Q. C. Geng, X. Y. Huang, H. J. Gong, W. W. Xu, G. P. Wang, D. Manocha, and R. G. Yang, "AADS: Augmented autonomous driving simulation using data-driven algorithms," *Science Robotics*, vol. 4, no. 28, 2019.

[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] B. A. Kitchenham and S. L. Pfleeger, "Principles of survey research: Parts 1 – 6," *Software Engineering Notes*, 1995 and 1996.

[45] M. Ciolkowski, O. Laitenberger, S. Vegas, and S. Biffl, *Practical Experiences in the Design and Conduct of Surveys in Empirical Software Engineering*. Springer Berlin Heidelberg, 2003, pp. 104–128.

[46] S. B. Robinson and K. F. Leonard, *Designing Quality Survey Questions*, 1st ed. SAGE Publications.

[47] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009, p. 5.

[48] K. Wyrobek, "The origin story of ROS, the Linux of robotics." [Online]. Available: https://spectrum.ieee.org/automaton/robotics/robotics-software/the-origin-story-of-ros-the-linux-of-robotics

[49] J. Saldaña, *The coding manual for qualitative researchers*. Sage, 2015.

[50] K. M. MacQueen, E. McLellan-Lemal, K. Bartholow, and B. Milstein, *Team-based Codebook Development: Structure, Process, and Agreement*. Rowman Altamira, 2008, pp. 119–135.

[51] K. Charmaz, *Constructing Grounded Theory*. Sage, 2014.

[52] F. P. Brooks Jr., "No silver bullet: Essence and accidents of software engineering," *Computer*, vol. 20, no. 4, pp. 10–19, April 1987.

[53] J. Micco, "Flaky tests at Google and how we mitigate them," May 2016. [Online]. Available: https://testing.googleblog.com/2016/05/flaky-tests-at-google-and-how-we.html

[54] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, "Scenic: a language for scenario specification and scene generation," in *Conference on Programming Language Design and Implementation*, ser. PLDI '19, 2019, pp. 63–78.

[55] R. Majumdar, A. Mathur, M. Pirron, L. Stegner, and D. Zufferey, "Paracosm: A language and tool for testing autonomous driving systems," *arXiv preprint arXiv:1902.01084*, 2019.

[56] F. Klück, Y. Li, M. Nica, J. Tao, and F. Wotawa, "Using ontologies for test suites generation for automated and autonomous driving functions," in *International Symposium on Software Reliability Engineering Workshops*, ser. ISSREW '18. IEEE, 2018, pp. 118–123.

[57] V. Garousi and M. V. Mäntylä, "When and what to automate in software testing? A multi-vocal literature review," *Information and Software Technology*, vol. 76, pp. 92–117, 2016.

[58] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," in *International Conference on Automated Software Engineering*, ser. ASE '16, 2016, pp. 426–437.

[59] L. Xavier, A. Brito, A. Hora, and M. T. Valente, "Historical and impact analysis of API breaking changes: A large-scale study," in *Software Analysis, Evolution and Reengineering*, ser. SANER '17, 2017, pp. 138–147.

[60] J. Bloch, "How to design a good API and why it matters," in *Companion to Object-Oriented Programming Systems, Languages, and Applications*, ser. OOPSLA '06, 2006, pp. 506–507.

[61] T. Leesatapornwongsa, J. F. Lukman, S. Lu, and H. S. Gunawi, "TaxDC: A taxonomy of non-deterministic concurrency bugs in datacenter distributed systems," in *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2016, pp. 517–530.

[62] C. Boelmann, L. Schwittmann, M. Waltereit, M. Wander, and T. Weis, "Application-level determinism in distributed systems," in *International Conference on Parallel and Distributed Systems*, ser. ICPADS '16. IEEE, 2016, pp. 989–998.

[63] H. Liu, G. Li, J. F. Lukman, J. Li, S. Lu, H. S. Gunawi, and C. Tian, "Dcatch: Automatically detecting distributed concurrency bugs in cloud systems," *Computer Architecture News*, vol. 45, no. 1, pp. 677–691, 2017.

[64] I. Dhiah el Diehn, S. Alouneh, R. Obermaisser *et al.*, "Incremental, distributed, and concurrent service coordination for reliable and deterministic systems-of-systems," *IEEE Systems Journal*, 2020.

[65] Ignition Robotics, "Ignition Gazebo: A Robotic Simulator." [Online]. Available: https://ignitionrobotics.org/libs/gazebo

[66] Amazon Web Services, "AWS RoboMaker." [Online]. Available: https://aws.amazon.com/robomaker

[67] O. Michel, "Cyberbotics Ltd. Webots™: Professional mobile robot simulation," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[68] A. M. Hertz, E. I. George, C. M. Vaccaro, and T. C. Brand, "Head-to-head comparison of three virtual-reality robotic surgery simulators," *Journal of the Society of Laparoendoscopic Surgeons*, vol. 22, no. 1, 2018.

[69] R. R Shamshiri, I. A. Hameed, L. Pitonakova, C. Weltzien, S. K. Balasundram, I. J Yule, T. E. Grift, and G. Chowdhary, "Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison," 2018.

[70] I. Beschastnikh, P. Wang, Y. Brun, and M. D. Ernst, "Debugging distributed systems," *Queue*, vol. 14, no. 2, pp. 91–110, 2016.

[71] X. Zheng and C. Julien, "Verification and validation in cyber physical systems: Research challenges and a way forward," in *Software Engineering for Smart Cyber-Physical Systems*, 2015, pp. 15–18.

[72] D. Yeo, G. Kim, and S. Kim, "Toward immersive self-driving simulations: Reports from a user study across six platforms," in *Conference on Human Factors in Computing Systems*, ser. CHI '20, 2020, pp. 1–12.

[73] S. Chen, Y. Chen, S. Zhang, and N. Zheng, "A novel integrated simulation and testing platform for self-driving cars with hardware in the loop," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 425–436, 2019.

[74] T. Duy Son, A. Bhave, and H. Van der Auweraer, "Simulation-based testing framework for autonomous driving development," in *International Conference on Mechatronics (ICM)*, ser. ICM '19, 2019, pp. 576–583.

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

[76] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in *International Conference on Robotics and Automation*, ser. ICRA '19, 2019, pp. 8271–8277.