Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

Master's Thesis

Property-Based Testing: Formalized Robotic Testing for Standard Compliance

Salman Omar Sohail

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Nico Hochgeschwender
Prof. Dr. Paul G. Plöger
M.Sc. Sven Schneider

December 2022

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Salman Omar Sohail

Abstract

Robotic standards are used to preserve and ensure the safety of a robot and its interactions with its environment. However, verifying and validating these standards on a robot remains a cumbersome process and is as yet non-automated. Thus, this study aims to accelerate and automate the validation process of existing robotic standards in a hierarchical and logical manner by means of simulated testing.

The approach of this project was to firstly identify the common test requirements in the current relevant robotic standards. Consequently, International Standard Organization (ISO) 23482-1 [1] was selected as the primary research subject because this robotic assessment standard encompasses tests for several other standards. An analysis was then carried out on ISO 23482-1, after which a methodology was devised to interpret the analyzed information by identifying simulatable tests, extracting key information, and mapping it into a Domain-Specific Language (DSL). In this methodology, the DSL employs the mapped information to generate and validate simulated test scenarios for mobile robots and robotic arms using the Property-Based Testing (PBT) framework [2]. A report is then generated.

This research has manifested that the proposed methodology is able to verify standard compliance from the mapped information of ISO 23482-1 in a variety of robots, including wheeled robots, tracked robots, quadrupeds, and robotic arms. This process unveiled several missing hardware and software functionalities in these robots. A further significant outcome was that two robot driver bugs were revealed, of which one was reproducible on a real robot. Additionally, the limitations of ISO 23482-1 were noted. This study not only enables quick multifaceted testing but also provides a considerable step towards bridging the gap between automation and standard conformity for different types of robots, utilizing simulation.

Acknowledgements

Firstly, I would like to start by offering my sincerest gratitude to my supervisors Prof. Nico Hochgeschwender, Prof. Paul G. Plöger, and M.Sc. Sven Schneider. Their guidance and support through discussions and lectures have enabled the completion of this Master's thesis. Secondly, I would also like to thank the staff and CEO of MYBOTSHOP for their cooperation as well as for providing resources to verify the faults detected in the robots detected by the property-based testing framework.

Thirdly, I would like to thank my mother, who is battling cancer, as well as my family, for their constant love and words of encouragement without which I would have not been as motivated throughout the entire duration of this thesis. Lastly, I would like to acknowledge my dearest and most closest friend Al-Rahman for his guidance and unyielding support.

Contents

List of Figures

List of Tables

1

Introduction

Robotic standards provide a guideline for existing and emerging robotic technologies in terms of technical requirements, terminologies, conventions, interchangeability of hardware components, safety, as well as security. It promotes streamlined robotic development for quality-assured production, testing, and integration.

(a) A mobile manipulation platform used for research in academia [14]

(b) A quadruped used for sewage tunnel inspection [14]

Industries nowadays frequently adopt and utilize robots to save costs and enhance their efficiency in terms of production. This trend is not only limited to industries but can be recognized in other areas such as academia, retail markets, agriculture, hospitals, and homes [18]. With the increased implementation of autonomous robots, safety is a great concern for which reason robotics standards are being extended and updated. Manual methods of testing the actions and behaviors of these autonomous machines have become infeasible due to the wide range of scenarios available [19, 20]. Hence, there has been a shift towards virtual test drives. Virtual test drives enable automation and ensure good coverage of safety testing in robots as well as the testing of robotic actions and behaviors. Several research papers can be found in which virtual test drives are used to test robotic actions and behaviors such as [21–23]. However, research papers focus on testing robotic actions and behaviors defined by their own metrics and not in accordance with robotic standard requirements.

Few research papers utilize virtual drives to test robotic systems for compliance with robotic standards. Some research projects such as [24] have tested their robotic system with several aspects of robotic

standards but to the author's knowledge, no research has been found that mainly focuses on complete standard compliance using virtual test drives for robots. Hence, in this thesis, the use of a property-based framework (a virtual test drive) is proposed to primarily validate different robotic actions and behaviors for standard compliance as well as secondarily validate different hardware functionalities where applicable. **Robotic standard compliance** means that the robot is compliant with the requirements stated by the standard. Robotic standard compliance can be categorized into two types. The first is the requirement of having certain hardware functionalities, such as the robot being weather resistant, enabling it to operate optimally in different types of weather conditions, such as hot and cold temperatures ISO 23482-1 §9 [1]. The second category is the requirement of having a particular software functionality, usually including the robot's actions and behaviors. An example of software functionality is that a robot must have the necessary software to successfully avoid obstacles when autonomously navigating to a given goal position ISO 23482-1 §16 [1].

Concluding, this thesis aims to enable automated robotic standard compliance (e.g. ISO/TR 23482-1) using a revised framework of property-based testing[1]. This framework has been refined and modularized in order to provide clarity and structure to the testing framework as well as to enable generalization so that different robots can be utilized within the framework. Moreover, a domain-specific language (DSL) has been integrated into the framework as a means to translate the required compatible standards into properties and scenarios for testing. The use cases include autonomous ground vehicles (e.g. Clearpath Husky, Clearpath Jackal, MBS ROVO2, B1) and a robotic arm (e.g. Ufactory xARM6) that are automatically tested using standard designated procedures as well as custom user-based tests. Both standard-based testing and user testing include autonomous navigation, terrain traversal, collision avoidance, dynamic stability, pick-and-place action, etc.

This project utilizes an assortment of software tools, the most prominent ones being ROS Noetic, Gazebo, Hypothesis, TextX, and Allure.

1.1 Motivation

As stated above, achieving safety and reliability for robotic actions and behaviors is one of the primary driving factors for this thesis. In order to achieve this, the robots are required to have fault-tolerant systems and to be able to function nominally in any given task [25]. Frequently, existing robotic systems do not have sufficient test coverage in terms of their actions and behaviors and therefore cannot identify failure points in different scenarios. Until now previous researchers have tested common failure points of software in order to address this problem. Identification of these common failure points can be carried out using standards as standards are developed based on market demand [26]. These standards have several benefits as listed below:

1. The standards are created by a technical specialist committee.

2. The standard tests are designed to rigorously test the robotic system based on common failure points.

[1] Property-based testing is a simulation-based framework for automated randomized testing of robot actions and behaviors.

3. They are independently verified and validated by various organizations around the world.

Standards, however, may have hundreds of testing requirements with varied scenarios, and even with these numerous scenarios, it may not be possible to cover all test cases, especially with unique scenarios. This thesis enables the automation of standard compliance testing of different robots in standard specified scenarios including unique scenarios using the property-based testing framework. Brief details on how the proposed framework achieves the safety and reliability goals are given as follows:

Performance

As far as performance is concerned, it is considerably faster to verify whether a robot is standard compliant as compared to the traditional method of setting up a scenario manually and verifying it. This is especially beneficial for manufacturers, researchers, and integrators who are developing and testing a system or a specification.

Automation

Performing a single standard test may take up to several hours and is also costly to set up. The reason for this is that several of the test specifications require not only special equipment but also require specific types of objects, terrains, and obstacles for the robot to perform standard designated tests. Using simulation, it is possible to not only significantly reduce the test times with minor modifications to the robot's configuration but also to automate these tests.

Natural Language Testing

A frequent issue when using a monolithic and complex software to enable technical users to test the robot is the time required to grasp the basic understanding of the software and to be able to start up a minimal example to test the robot. This issue becomes even more prominent for those who are inexperienced in this field. In order to tackle this issue a domain-specific language (DSL) was selected to set up and execute tests. The DSL allows users to express their test scenarios and requirements in natural language using a specific format making delving into the code no longer necessary. This approach allows both novice and advanced users alike to conduct tests both easily and quickly.

Standard Verifiability

Standards often are costly and time intensive to set up as well as including hundreds of tests. This thesis enables standards' tests to be quickly, efficiently, and cost-effectively verified.

1.2 Research Question

This master thesis addresses the challenge of how to automatically validate standard compliance[2] in different robots via Property-Based Testing using natural language. Three aspects relating to this challenge are queried:

[2]Robotic Standard Compliance Definition: 1

RQ1 Do different robotic standards share common test requirements? For this purpose, we survey a variety of robotic standards to identify common test elements between them (Section 2.6, Appendix: C).

RQ2 Can the robotic standards be expressed in natural language that can be interpreted by machines for validation in a simulation? For this query, we formalize a development process to translate existing robotic standards into verifiable use cases that are testable in simulation (Section 4.3, Appendix: D).

RQ3 Can Property-Based Testing be used for validation of standard compliance in various mobile robots and robotic arms? For this research question, we formalize of a systematic process of identifying and converting key features of existing robotic standards into **properties** that can be tested and validated via the property-based testing framework [2, 10] for mobile robots and robotic arms (Fig. 4.1).

1.3 Research Contribution

Informed Automation

The first expected contribution of this thesis is an informed automation of robotic standards using the property-based testing framework, enabling both scenarios that can be customized as well as rapid testing. Additionally, the testing will provide a theoretical certification of the robotic standards that are simulatable and feasible provided that the scenarios are set up correctly according to the standard specifications, the simulated robots are accurately configured according to the real robots, and the physics simulator has the correct parameter configuration.

Safety & Reliability

Using the property-based testing framework, safety is ensured as not only can standard requirements be validated and verified for a robot, but also automated randomized tests can be run for custom scenarios in which the robot is to be deployed, so ensuring critical or anomalous behavior is identified and reported. In terms of reliability, repeated randomized testing ensures that the robot consistently makes autonomous safe decisions in a range of diversified scenarios.

Heterogeneous Robot Testing

Another expected contribution is that the property-based testing framework will enable the testing of a multitude of test cases involving minimal effort for various types of mobile robots as well as robotic arms.

2

Related Work

In this chapter, we perform a literature review on similar existing frameworks that use virtual test drives for verifying standard compliance for robotic actions and behaviors after which we look into standards pertaining to safety in robotics for selecting a standard to test the safety of robots using the Property-Based Testing framework (PBT).

2.1 Evaluation of Robots in Simulation

First, we begin by investigating the current testing methodologies employed by academics for testing and evaluating robots in Simulation. The reason for investigating testing in simulation is because simulation-based testing provides a cost-effective and efficient manner of testing robots. Furthermore, testing is required to increase the safety, dependability, quality, security, and performance of a robotic system. As robots have hardware and software components, both require testing and the standards provide testing procedures for both these components. The focus of this study is primarily on testing the software of robots. When it comes to software testing of robots, we can categorize them into types. The first type is testing the code to ensure that there are no bugs or faults. The second type is to test the robot's autonomous action and behavior response, which is more challenging as compared to code testing. The reason for this is that the unintended autonomous robot action or behavior cannot always be determined by the source code. To identify such behaviors, the robot is required to play out a scenario in which an observer evaluates the actions of the robot and rectifies the error. Normally, the observer is a human, however, different techniques exist such as PBT that enable the simulated environment to act as the observer and evaluator.

Reviewing some prominent testing techniques for robots in simulation, Timperley et al. [27] answers the research question of whether simulation-based tests are adequate enough to detect faults in the autonomous behavior of real-world robots. The aim of their study was to efficiently catch the bugs in the robotic software. The methodology they employed to accomplish their aim was by means of utilizing simulation in conjunction with automated testing. The use case that was investigated by them was the ARDUPILOT project which is an autonomous navigation software for mobile robots. To identify the faults, the authors analyzed 29,000+ commit history of the ARDUPILOT project and filtered out 228 bugs that could be verifiable in simulation. The result of their examination was that numerous bugs simulated and tested by their framework for the ARDUPILOT project were independent of the simulation and

hardware. Furthermore, the faults were primarily due to the incorrect configuration of the robotic software system. Nonetheless, there were a few bugs due to hardware and the environment. The contribution of their study provides a strong argument that simulation-based testing techniques are a viable approach for testing robotic actions and behaviors.

Looking more into basic robot testing techniques, Bihlmaier et al. [28] present Robot Unit Tests (RUT) as a means to test robots in virtual environments automatically. The objective of this study was to improve robotic software systems. It follows the principles of Test Driven Development (TDD) and Continuous Integration (CI). The idea behind RUT is to create small test segments in simulation to test a specific functionality of a robot using simulation. The use case examined by them for their research was a robotic arm that was tested in different cartesian poses to verify whether it was in a collision or not. The collision was verified by visual means i.e. via a camera mounted on the robotic arm. The result of their work was an easy process of developing tests (RUTs) for testing robotics functionalities.

Recapping, several different types of simulation-based testing are available that vary mainly in the formulation of the scenario or in the verification of said scenario such as [27, 28]. The challenge is that the verification methodology of simulation-based testing is often not effortlessly transferable to other robots and may require specific simulated hardware or software components. Hence, there is a great benefit to taking advantage of the simulator to provide verification for testing the robotic actions and behaviors, due to which we have opted to use the PBT framework as the primary scenario generation and verification mechanism for this work.

2.2 Property-Based Testing

Property-Based Testing (PBT) is an essential part of this work. In this section, we take a look into how different researchers have employed PBT for verifying and validating robot actions.

Santos and co-authors [29] put forward a methodology for automatically testing robotic software. The objective of their work was to identify and detect bugs before the deployment of a robot to ensure no critical bugs occur during the run-time of the robot. The proposed methodology tests the Robot Operating System (ROS) packages of a robot. It does so by providing shuffled inputs to the different modules of ROS in the robot and then verifies whether the output is as expected or not. The result of this work was that it was able to successfully and consistently identify bugs in the robotic software such as triggering an event that caused the robot to believe it was in a collision.

Akin to Santos, Xie et al. [30] utilize PBT in conjunction with Fuzz testing to verify the integrity of robotic software. The objective of their research is to identify bugs in robotic software and verify the correct behavior of the robot when given a large variety of valid inputs. For this purpose, they generate diverse inputs by mutating input data to create syntactically correct new input data. They then pass this data to a robot in a simulator that performs the given task. Verification of the tests is left up to the user and can be performed using different third-party software such as AddressSanitizer. The outcome of their work was that it was able to determine 43 software bugs in different robot software.

Both Santos and Xie [29, 30] provide exceptional testing techniques for testing the source code of the robot. However, their testing method focuses on the type-I tests (as mentioned in Section 2.1) in which

they focus on creating valid tests for source code verification. However, their approach does not directly verify the robot's action or behavior and is not capable of identifying faulty behaviors in a robot. For example, we take a test case in which the robot performs navigation towards a goal but collides with an object on the way but the robot still reaches the goal successfully. Their method which is verifying the output of the ROS message will return success (at least for the method proposed by Santos). However, due to no information from the environment, the unwanted behavior of collision will not be detected. For this, the proposed framework by Sohail et al. [2, 10] is more adequate as it uses the environment to validate several factors of the robot's behavior such as collision with the environment.

In short, the work presented by Santos and Xie may serve as complementary precursor tests that ensure that the basic robot functions work as intended and for more complex robot action and behavior tests, the framework by Sohail is more suitable as it provides a more in-depth verification method that makes use of the simulated environment information. More information on the framework by Sohail is in available in Chapter 3.

2.3 Standard Compliant Robotic Frameworks

A work that focuses on standard compliance for robotic testing has been presented by Cosmo et al. [31] in which they proposed a verification methodology for ensuring compliance with the Protective Separation Distance (PSD) requirement issued by the International Standard Organization (ISO) standards [8, 17]. The verification methodology utilizes velocity-dependent safety zones (dynamic safety zones) which are achieved via bounding volumes (BV). The BV calculates whether an assigned imaginary volume that encompasses a robot and a personnel intersects. Dependent on the intersection, the robot either lowers its speed or completely halts. The motivation for this work stems from increasing safety in human-robot collaboration (HRC) tasks. Experiments have been conducted with the Franka-Emika robot in which comparisons were made with the recommended static safety zones by ISO [5] and the proposed dynamic safety zones. The result of the comparison demonstrated a 10% efficiency increase.

A different approach for the verification of standard compliance was performed by Guiochet et al. [32] in which they investigated the steps needed to create a safe and reliable Multimodal Interactive Robot for Assistance in Strolling (MIRAS)[1]. The result of their investigation led them to take steps for ensuring the safety of the human user and environment which included designing risk assessment, safety cases, and hazard operability. The motivation for their work was to create a human-rehabilitation robot that encompasses all necessary safety standards provided by various regional as well as international regulatory organizations. In their research, non-medical robotic standards had been phased out due to non-applicability in MIRAS such as ISO10218 and ISO13482. The standards that they utilized were the ISO guide 51, ISO guide 63, ISO31000, ISO12100, ISO14971, ISO7176-5, ISO11199-2, and Machinery Directive 2006/42/EC. Additionally, they used medical experts' feedback for the development of MIRAS. They achieved safety validation for MIRAS via a risk-acceptability matrix as well as through safety cases.

In terms of practical analysis of the standards, Rosenstrauch et al. [33] present an overview of the ISO/TS 15066 standard as well as an implemented demonstrative use case. In their work, they explain

[1] A robotic rollator.

how standard ISO/TS 15066 extends and updates the ISO 10218 standard in the context of collaborative robots fleshing out its details. An example that is presented is that the ISO 10218 requires that robotic movement "should not create an injury" and the ISO /TS 15066 provides parameters and limits for force and speed to act in accordance with ISO 10218. Similar to this example, several other safety-based skills are given in the ISO/TS 15066. The use-case includes a 6 DOF manipulator arm consisting of *Schunk Amtec PowerCube* with the safety skill of power and speed limiting extracted from ISO/TS 15066. The method of validation of this experiment was by measuring the deformation of a subject's flesh when the manipulator's end-of-effector came in contact with it. The result of this use case was that the parameters issued by the ISO/TS 15066 proved to be insufficient as the flesh was penetrated beyond reasonable limits stated by the standard. It seems to the authors that the equations and parameters provided by ISO/TS 15066 are suited more to "modern lightweight" robots and not to heavy industrial robots. This research strengthens our own proposed framework of property-based tests which validates safety-requirement use cases and has the ability to vary parameters in accordance with the robot being used and verify that no constraints or limitations have been violated.

Another work that focuses on the analysis and implementation of the standard is provided by Michalos et al. [3] in which they present the result of their investigation of safety in Human Robotic Interaction (HRI) in accordance with the European directive as well as international standards. They narrow down the relevant safety features from several standards some of which are listed in table C.1 and provide a brief and descriptive overview of these safety features. In addition to the identification of relevant safety mechanisms from the standards, they present three use cases that utilize the safety mechanisms. Three of the use cases are robotic manipulator arms that are working in an industrial setting with the task of assembling of a given component; the first being an automotive dashboard assembly, the second being an automotive rear suspension assembly, and finally the third being a refrigerators assembly. An evaluation was performed on the three use cases which had an implementation of various safety mechanisms. The result of the evaluation can be viewed in Fig. 2.1. One of the conclusions that can be taken from their work is that the safety mechanisms and their final goals cannot be set in stone and have to be adapted to the work requirement. For example in use-case:1 and use-case:2, separation of the robot to the human was acceptable but for use-case:3 separation was unavoidable. Flexibility in safety verification and validation methodology plays a huge role in the deployment of robotics commercially.

Collaborative methods	Case 1	Case 2	Case 3
Speed & Separation Monitoring	X	X	
Power & Force Limiting			X
Hand Guiding		X	X
Safety functions	**Case 1**	**Case 2**	**Case3**
Safety monitored stop	X	X	X
Enabling Device		X	X
Safety-Rated Monitored Speed	X	X	X
Safety-Rated Reduced Speed	X	X	X
Safety-Rated soft axis	X	X	X
Space to Stop Monitoring	X	X	X
Deceleration Monitoring	X	X	X
IR: Cartesian Regions		X	X
IR: Cartesian Safe Limited Position		X	X
Safe Tool Orientation		X	X
Force and Impedance Control		X	X
Collision Detection			X
Collision Avoidance			X

Figure 2.1: Figure by Michalos et al. [3]. The figure illustrates the result of the application of several safety mechanisms in the three use cases that were investigated.

2.4 Robotic Standards

Concerning the general applicability of standard testing to robots, an extensive survey has been conducted by Herrmann et al. [34] in which they identify and describe key research papers that focus on the development of safety-related frameworks for Human Robotic Interaction (HRI) scenarios including standard compliance. In their work, they discuss testing methodologies pertaining to safety planners, auxiliary human-monitoring sensors, and risk-acceptability matrix. A result can be drawn from the research that many of the current testing methodologies inclusive of the standard that is employed for HRI are not generalizable outside their designated scenarios. Hence, in terms of the general applicability of standards to robots, some of the standard tests are applicable in almost every situation, particularly those tests that do not test the behavior of the robot and instead test a hardware functionality or a specific action, however, the standard tests which test HRI are commonly not all-encompassing and cater to certain situations and their applicability is dependent on the concerned party (i.e. manufacturers, testers, and integrators).

Another research similar to Herrmann et al. [34] but more recent is presented by Valori et al. [4] in which they provide a comprehensive survey article that covers the standard safety tests and its applications in the context of human-robot interaction (HRI).

In the survey article, they summarize thirteen standards applicable to autonomous systems and list out the essential information on the safety-test protocols, recovery methods, and required safety skills for the autonomous system in question and provide existing use cases for them as well. Moreover, they point out the conflicts between standards; one of particular interest is the one in which a manipulator platform has a person's speed taken into account according to ISO 15066 [5], whereas ISO 3691 [6] does not. They mention that although these standards serve as soft laws and are not legally binding, it is imperative for the standards to be consistent and coherent. Another interesting part of their work was the complied safety skills that a robot must have extracted from the standards. This is depicted in Fig. 2.2.

In terms of standard safety framework enhancement, Chemweno et al. [9] proposed a new framework ISO 31000 for designing safeguards for collaborative robots to cover up the gaps of hazards and risks left by ISO 15066 (Normative standard for robots and robotic devices - collaborative robots) which is an extension of ISO 10218 (Standard for robots and robotic devices - safety requirements for industrial robots). The drive for their work was to increase safety in collaborative robot applications specifically when the robot shares a space with a human or there is a shared task between the human and the robot. Their research analysis shows that ISO 15066 and ISO 10218 introduce and emphasize hazard and safety analysis but does not provide concrete guidelines for its realization. In order to facilitate its realization, Chemweno et al. presented a systematic framework for designing and implementing the safety standards as illustrated in Fig.2.3.

The result of their work was a framework that covers the gaps left in standards ISO 15066 as well as ISO 10218 and provides an insight on what should be considered in the development of this thesis.

A research article akin to Chemweno et al. [9] is an article published by Harper et al. [25]. This research provides an overview of the enhancements performed in the ISO 10218 and the development of ISO 13482 standards for robots. The key points discussed for ISO 10218 were the addition of the human-robot

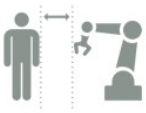
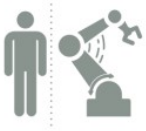
Icon	Safety Skill	Corresponding Operating Modes and/or Testing Procedures with Standard Reference
	Maintain safe distance (MSD)	• ISO/TS 15066 [30]: SRMS (§5.5.2), SSM (§5.5.4). • ISO/TR 23482-1 [37]: response to safety-related obstacles on the ground (§15). • ISO 3691-4 [35]: tests for detection of persons (§5.2).
	Maintain dynamic stability (DYS)	• ISO 3691-4 [35]: stability tests (§5.3). • ISO/TR 23482-1 [37]: static stability characteristics (§11), dynamic stability characteristics with respect to moving parts (§12), dynamic stability characteristics with respect to travel (§13), response to safety-related obstacles on the ground (§15).
	Limit physical interaction energy (LIE)	• ISO/TS 15066 [30]: PFL (§5.5.5), (HG, §5.5.3). • IEC-80601-2-78 [36]: overtravel end stops (§201.9.2.3.2), movement beyond pre-set limits for individual. Patient movement (§201.9.2.3.101), movement or force/torque of RACA robot (§201.12.101), mechanical hazards associated with misalignment (§201.9.101). • ISO/TR 23482-1 [37]: physical hazard characteristics (for mobile robots), (§7)
	Limit range of movement (LRM)	• ISO 10218-1 [28]: speed control (§5.6), axis and space limiting (§5.12). • IEC-80601-2-78 [36]: movement beyond pre-set limits for individual patient movement (§201.9.2.3.101), overtravel end stops (§201.9.2.3.2) • (ISO/TS 15066 [30]: HG, §5.5.3).
	Maintain proper alignment (MPA)	• IEC-80601-2-78 [36]: mechanical hazards associated with misalignment (§201.9.101)
	Limit restraining energy (LRE)	• ISO/TR 23482-1 [37]: test of physical hazard characteristics (for restraint-type physical assistance robots), (§8). • IEC 80601-2-78 [36]: accuracy of controls and instruments and protection against hazardous outputs (§201.12), mechanical hazards associated with misalignment (§201.9.101), movement beyond pre-set limits for individual patient movement (§201.9.2.3.101).

Figure 2.2: Figure by Valori et al. [4]. The figure illustrates identified standard-compliant safety skills for robots. Standards utilized are: ISO/TS 15066- [5], ISO/TR 23482-1 [1], ISO 3691 [6], IEC-80601-2-78 [7], ISO 10218-1 [8]

collaboration mode which included the stopping function, speed and position control, power and force control, and finally operation workspaces. As for ISO 13482, the key features were designating safety requirements for the different types of robots such as task definition, environment designation, hazard identification, and validation tests as well as the design of safety cases. The work provides keen insight into the development and decision-making by the ISO committee for the formulation and consideration required for standard definitions.

Following up on ISO 13482, Weng et al. [35] discuss the safety features of ISO 13482 [36]. In their work, they discuss the safety results required by three types of robots: the mobile servant robot, the person carrier robot, and the physical assistant robot. A highlight from their discussion is that the standard ISO

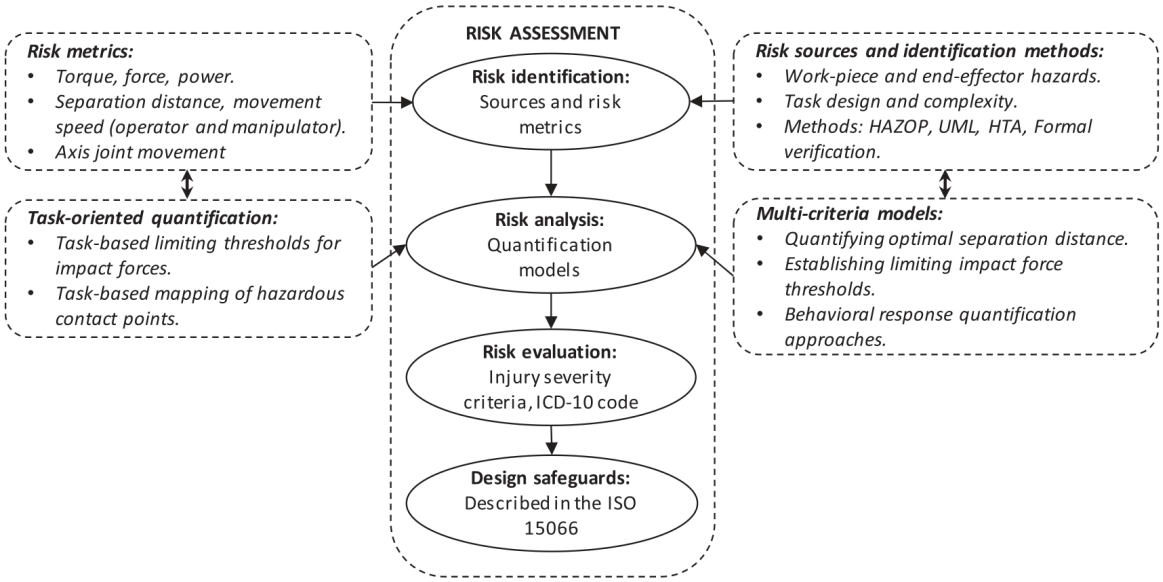


Figure 2.3: ISO 15066 proposed safety extension framework by Chemweno et al. [9]

13842 is agnostic to how the results of a robot's action or behavior are achieved. This means that the standard does not take into consideration how the software planner plans an action or sequence of actions but only the final course of action/s performed by the robot. Additionally, this work also discusses the ethicality of the robots and how it lies with the integrator of the robotic system. Moreover, this article provides vital information that should be considered while developing a standard based safety-framework and the formulation process is similar to that which we wish to achieve i.e. focus on robot detecting and identifying safety violations and not on recovery methods via software planners.

Discussing some of the negative points on the standards, we take a look at the work presented by Villaronga et al. [37] who discusses the legal implications of robotic laws. They mention that the robotic standards that are available are lacking in several areas of which the most prominent ones being in law and the estimation of the capabilities of different autonomous machines. In their work, they illustrate the intricate difficulties between governmental and non-governmental regulators for robotic laws (including service robots, self-driving cars, delivery robots, and health-care robots). One of the emphasized points in their articles is that the current robotic standards are heavily politically driven and there is no clear distinction between hard laws and soft laws for autonomous technologies (Hard laws are those laws that are reprimandable by law whereas soft laws serve more as a guideline). Moreover, there appears to be no clarity in the robotics laws themselves in terms of area of applicability (e.g. cyber-threats, psychological damage, responsibility, evolving cognition, societal seclusion, etc.). A few potential solutions suggested were the assimilation of private and public impact assessments into new policies, an evidence based-evaluation of the implemented technologies (Smart regulations), and finally a stringent policy for ethics on robotic integrators and designers. Relating to this thesis, this work highlights that existing and new impending robotic laws may not serve as a sufficient source of reliability, and hence the ability to test

robots has to be expansive and flexible enough to integrate different scenarios which in turn may serve as a positive legal argument.

2.5 Limitations of previous work

At present, there is a lack of research that is available on automated testing for robotic standard compliance. Although academic papers [31, 32, 34, 38] are available in which researchers and industrialists demonstrate standard compliance for certain robots using customized safety frameworks, planners, auxiliary sensors, and risk assessments, none of these works (to the authors' knowledge) have a generalized testing framework[2] that can extensively and expansively verify robotic actions and behaviors in accordance with the standards.

2.6 Standards Review

The aim of the standard review is to identify and select those standards that enhance safety in robotics. In total, we surveyed and selected 40 preliminary standards from over 75 standards. Identification and the selection of the standards have been done based on the relevance of the standards to the field of autonomous systems and robotics as well as recommendations suggested by researchers and industrialists in research papers and technical manuals. The title and description for all 40 standards are available in **Appendix C**. The result of the review was 40 standards which have been categorized in table 2.1.

Robot Type	**Standards Assessed**
Mobile Robots	ISO 9787:2013, ISO 18497:2018, ISO 18646-1:2016, ISO 18646-2:2019,ISO 10218-1:2011, ISO 10218-2:2011, ISO 13482:2014, ISO/TS 15066:2016, ISO/TR 20218-2:2017, ISO/TR 23482-1:2020, ISO/TR 23482-2:2019, ISO 3691-4:2020
Robot Manipulators	ISO 9283:1998, ISO 18646-3:2021, ISO 10218-1:2011, ISO 10218-2:2011, ISO 13482:2014, ISO/TS 15066:2016, ISO/TR 20218-1:2018, ISO/TR 20218-2:2017, ISO/TR 23482-1:2020, ISO/TR 23482-2:2019

Table 2.1: The evaluatory standards given in table C.1 have been separated into two categories based on their target platform, one for mobile robots and one for manipulators. As some standards contain safety descriptions for both mobile robots and robot manipulation, hence, they are mentioned in both categories.

[2]Generalized testing framework is defined as a base software that is capable of testing different types of mobile robots and robotic arms.

2.6.1 Standard Selection

For the selection of the standard, based on our review, the common safety tests identified between the different standards were as follows:

1. **Maintain Safety Distance:** ISO/TS 15066 §5.5.2, §5.5.4, ISO/TR 23842-1 §15, ISO 3691-4 §5.2
2. **Static Stability:** ISO 3691-4:2020 §5.3, ISO/TR 23482-1:2020 §11, §13
3. **Dynamic Stability:** ISO/TR 23482-1:2020 §12
4. **Limit Range of Movement:** ISO 10218-1 §5.6, §5.12, ISO/TR 23482-1:2020 §15.2
5. **Person Detection:** ISO 3691-4:2020 §5.2, ISO/TR 23482-1:2020 §18.2
6. **Response to Ground Obstacles:** ISO/TR 23482-1:2020 §15.1
7. **Safety of Localization and Navigation Stack:** ISO/TR 23482-1:2020 §16
8. **Robustness of Autonomous Actions:** ISO/TR 23482-1:2020 §17
9. **Speed Rating:** ISO 18646-1:2016 §5, ISO/TR 23482-1:2020 §15.2
10. **Stop Rating:** ISO 18646-1:2016 §6, ISO/TR 23482-1:2020 §15.1
11. **Obstacle Detection:** ISO 18646-2:2019 §6, ISO/TR 23482-1:2020 §17
12. **Obstacle Avoidance:** ISO 18646-2:2019 §7, ISO/TR 23482-1:2020 §16

From these common tests, it can be noted that ISO 23482-1 is present in all of them. This is one of the primary reasons why we have selected ISO 23482-1 as the standard for which we will test different robotic actions and behavior compliance using property-based testing. Furthermore, ISO 23482-1 is a purely evaluatory standard for different types of personal care robots encompassing both mobile robots as well as robotic manipulators and its testing mechanism as mentioned in the standard can be extended to non-personal care robots as well. Furthermore, it provides verification and validation tests for several other standards such as **ISO/TS 15066**, **ISO 13482:2014**, and **ISO 7176**.

3

Background: Property-Based Testing

This chapter serves as a brief recap to the property-based testing framework which is a core part of this thesis. **All information within this chapter has been extracted from** [2, 10]. It should be noted that the modeling of the actions in the presented framework has been changed and these changes will be elaborated in the upcoming chapters.

3.1 Overview

Property-based testing is a software validation framework that ascertains whether designated specifications are up to par. This is conducted by feeding the property-based testing framework with random synthesized input within a certain parameter range and then validating the output in simulation by comparison with the expected output in a given tolerance range.

3.2 Framework

The property-based testing framework was developed to answer the question of "How to test robot behaviors in varied execution scenarios?"

The proposed framework for solving the research question was by means of property-based testing in which a set of simulated scenarios are developed, after which then a robot executes a task and evaluates its actions by using pre-defined property-based tests as seen in Fig. 3.1. The cycle starts from a test configuration in which scenarios are defined and generated which are then fed into a property-based test suite. The property-based test suite then utilizes both an action ontology and a test configuration to set up a variety of tests which it verifies via the robot architecture and robot middleware. This cycle may be repeated indefinitely. At the end of the cycle, a report is generated that contains the results for all the tests.

3.3 Property-Based Test Suite

In the property-based test suite, tests are designed around a system's properties and it is ascertained whether those properties are satisfied or not. It has four steps:

1. Modeling the system and its expected behavior.

2. Determining the allowed range of input parameters for the tested components.

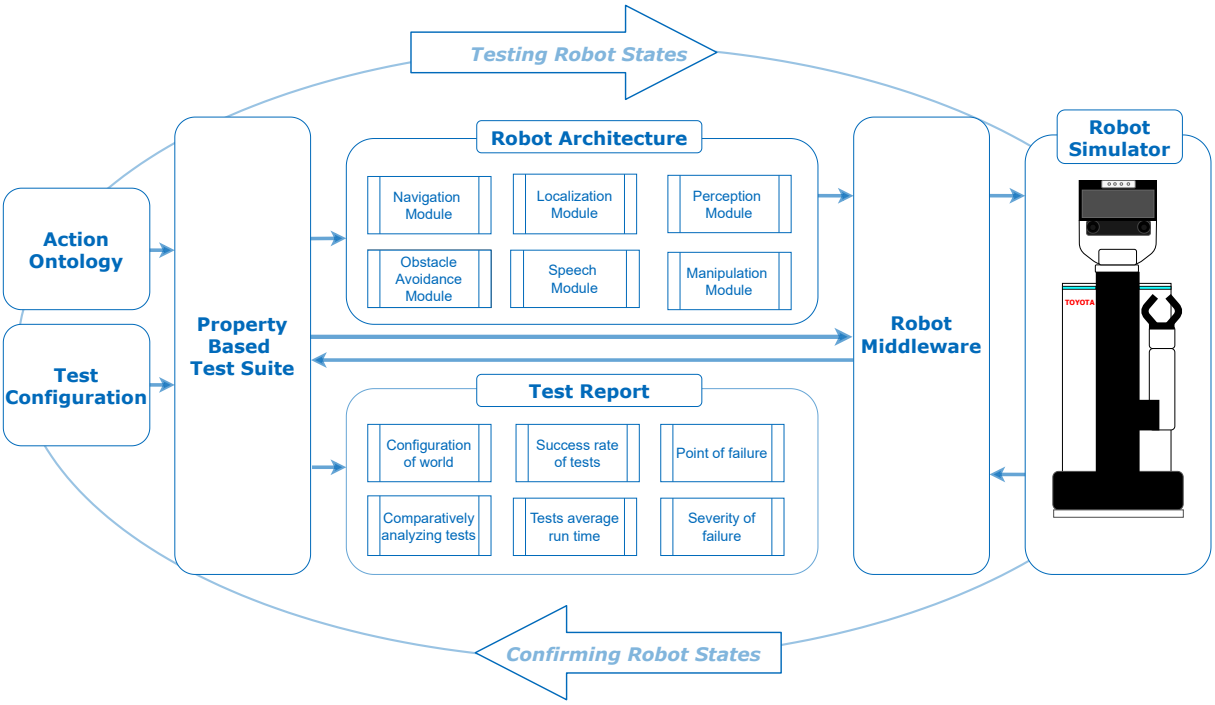


Figure 3.1: Property-based testing framework. Figure by Sohail [10]

3. Generating input parameters.

4. Validating the expected behavior of the modeled system.

The implementation is established on the hypothesis testing framework [39].

3.4 Scenario generation

In test scenario generation, a variety of scenarios are generated by spawning objects in different locations using the Monte Carlo method.

Each object has been manually designed allowing it to introduce varied aspects to the objects, such as changed size, shape, mass, inertia, etc. To ensure that items do not spawn in the same location due to randomness, the axis-aligned bounding box approach is utilized which is a fast and efficient method of determining valid spawn locations. An example of scenario generation using the axis-aligned bounding box method is given in Fig. 3.2.

3.5 Parameterization

To add more control and flexibility to the scenario generation, parameterization was enabled for scenarios via a test scenario configuration. The presented table 3.1 shows a subset of the parameters that can be configured for scenario generation. Through these parameters, the number of objects that could be spawned, the type of objects, the world environment, the type of scenario, and constraints could be

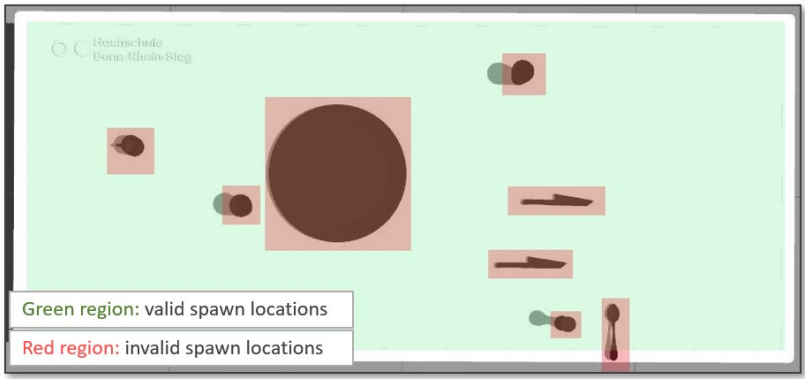


Figure 3.2: Axis-aligned bounding box method for placement of objects. Figure by Sohail [10]

determined. The test scenario configuration narrows down our test domain to what is required, making it feasible enough to perform exhaustive flexible qualitative testing. The parameters are extensible for the robot and the environment.

Parameter	**Description**
tests	List of tests to execute; each test is specified as a list of actions
test_count	Number of times to run each test
test_launcher	Path to a launch file that starts all components required by the tests
model_dir	Path to a directory of 3D models used in the tests
worlds	List of possible environments in which a test scenario can take place
model_list	Object models that can be used for manipulation
nav_obstacle_list	Object models that can be placed as navigation obstacles
nav_obstacle_count	Number of navigation obstacles to place in the environment
location_list	Names of locations to which the robot can navigate
object_surfaces	List of surfaces on which objects can be placed for manipulation
place_object_surfaces	List of possible surfaces to which objects can be brought

Table 3.1: Scenario configuration parameters. Table from [2]

3.6 Action Ontology

In the action ontology, an association is defined between a robot's action and the properties that define the action's success. Each of these associations is modeled using the web ontology language OWL.

We define various properties in the ontology such as:

- **performedIn** relates the task execution of a robot to the reference frame in which the task execution is performed
- **successProperty** describes the properties to be verified within a given task
- **hasParameter** and **needsParameter** are the requirements that are pre-defined for action and property respectively both of which are based on the context of an task.

3.7 Test Report Generation

In the test report generation, we store all the important data including the test scenario configuration in an HTML format. Additional details that are stored are the description of the tests, consistency, world properties, and various other parameters. The software used for the test report generation is the Allure framework [40].

4

Methodology

To answer the research questions in Section 1.2, the proposed thesis develops a formalization process using a Domain-Specific Language (DSL) (i.e. Robot Test Definition Language (RTDL) in this case) for translating existing standards into simulated test cases that will be verified by the property-based testing framework [2, 10] for a variety of robots. Fig. 4.1 illustrates the proposed methodology overview. The objective is to bridge the gap between standard compliance testing and automation. The following sections will enumerate the objectives and their descriptions.

The starting sections of this chapter consist of modifications to the existing modules of the Property-Based Testing framework (PBT), namely, the scenario generation, the action tests, and the complex-scenario tests. In scenario generation, a re-evaluation is performed on the simulators to assess the feasibility of new upcoming simulators such as Omniverse's NVIDIA Issac Sim. The reason for this is to address one of the deficits mentioned in the research paper [2] of the simulation and reality gap. More on this will be discussed in the upcoming sections. For action tests and complex-scenarios tests, the user-defined property definitions are redefined into hierarchical modularized property definitions that can be configured via a DSL into a concrete verifiable test. This modified version of the PBT is named as the Revised Property-Based Testing framework (RPBT). The motive for RBPT is to provide clarity and structure to the action and behavior tests as well as to enable multi-robot testing.

In the later sections, the surveyed standards in Chapter 2 are examined. The common aspects between those standards are then identified and analyzed. On the basis of the analyzed result, a standard is selected and then formalized into implementable test cases. This follows the methodology illustrated in Fig. 4.1. Initially, a feasibility analysis is performed on the standard test cases to determine which cases can be reproduced in simulation. After which, key information, as well as requirements, are extracted from the selected test cases and transformed into test definitions as can be viewed in Appendix D. These test definitions are then written into the RTDL and then verified through the RPBT.

In the final sections of this chapter, research validation is discussed which focuses on the effectiveness of the proposed methodology. Furthermore, the use cases in which the standard compliance will be tested are briefly discussed.

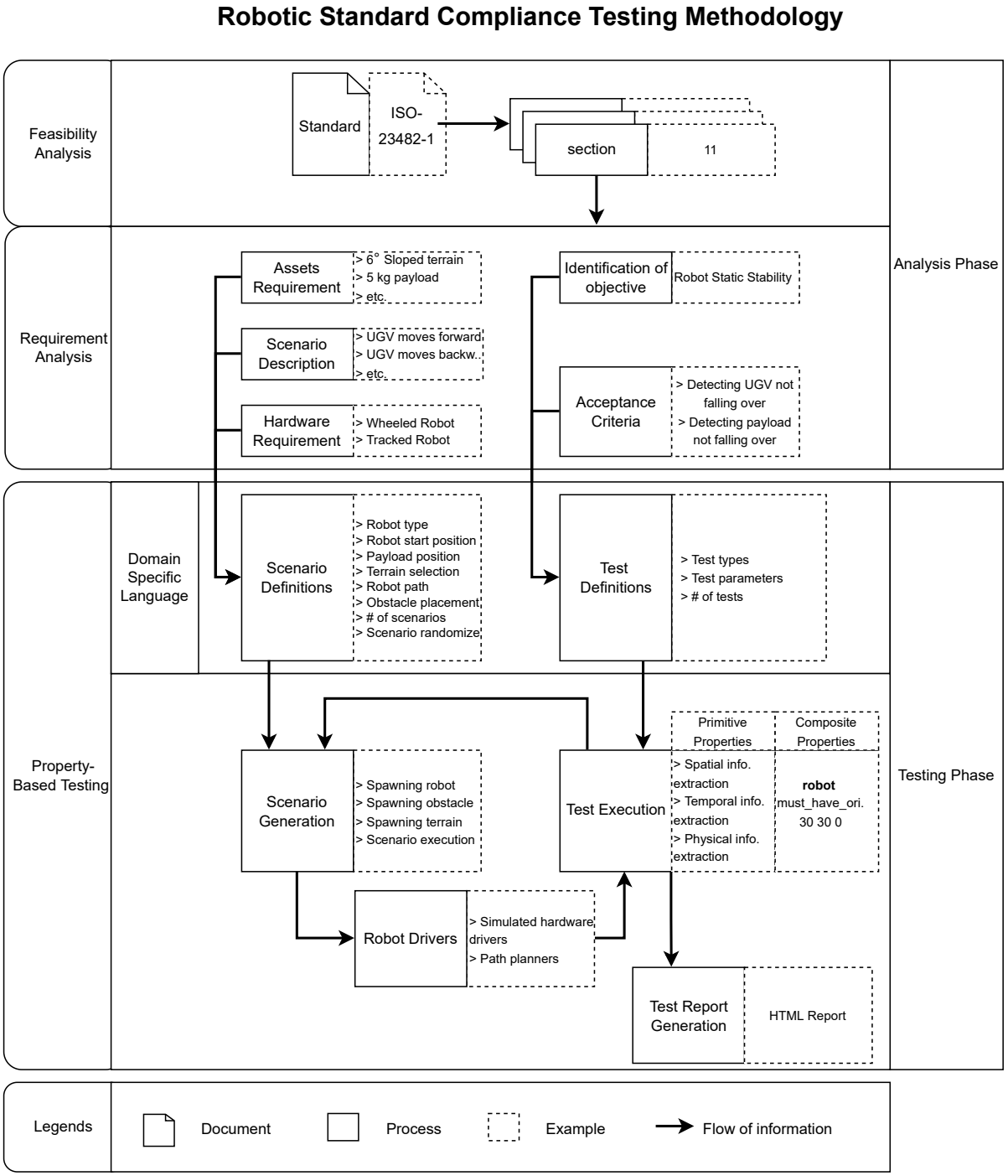


Figure 4.1: This figure illustrates a high-level overview of the proposed methodology. The workflow is based on [11]. In it, a selected standard is analyzed and relevant information is extracted after which the information is formulated into test-case scenarios conforming to the standard and then passed on to the property-based testing framework which verifies whether a robot complies with the given scenario.

4.1 Simulator

Scenario generation via simulator is a core part of enabling automation in testing robotic actions and robotic behaviors in the Property-Based Testing framework. In order to achieve more accurate and optimized simulations, a new simulator is investigated which has been developed by NVIDIA known as Issac Sim. Other simulators such as Webots, V-Rep, USARSim, and Morse will not be discussed as they have already been covered in [10].

4.1.0 Gazebo

As a refresher for the currently used scenario generation tool previously discussed in [10], Gazebo [12] is an open-source simulation tool for three-dimensional multi-agent robots (2004). The sim-tool offers good control of the information flow of robots and objects that are placed within it along with information visualizations, environment design, simplicity, and physical property assignments.

Figure 4.2: Gazebo robotic simulator [12]

It can import 3D objects in several different file formats, including .dae and .stl, the ability to alter a scene in an XML file, and it has strong community support. A few of the capabilities that are required but Gazebo lacks include the ability to manipulate 3D object meshes. In terms of disadvantages for Gazebo, it is necessary for considerable model pre-processing prior to import, it has frequent crashes, and is quite slow in performance [41].

4.1.0 Nvidia Issac Sim

Nvidia Issac Sim[1] [13] is one of the most advanced simulators for robotic systems to date and is part of Nvidia's Omniverse platform [13]. The Omniverse is a GPU-based simulator platform for researchers and industrialists. Nvidia Issac Sim incorporates numerous amounts of features including Pixar's Universal Scene Description Format (USD), Computer-Aided Design integration (CAD), Robotic Operating System (ROS), machine learning modules, and much more. It has a highly powerful physics engine capable of raytracing and dynamic physical properties assignment, etc. However, it has a few drawbacks noted by the author when being tested which are:

Figure 4.3: Nvidia Omniverse Issac Sim [13]

1. Steep learning curve

[1] It has just been released from its beta phase.

- One of the greatest advantages of Issac Sim is its compatibility with other software, however, this also proves to be one of its great barriers as one is required to be familiar with several different other software such as Pixar's USD to effectively use it.
- Another issue is its poor documentation on the implementation of simple functions for the robot and the simulator interfacing, although revisions are being performed in its documentation to cope with this issue, it seems that much time is required for the documentation to mature.

2. Extensive hardware requirement
    - It has a high minimum requirement to run the simulator when considered for an individual user e.g. Nvidia RTX series, 32 GB RAM+.
3. Small community (currently)
    - It is hard to resolve an issue when using Issac Sim since the community of users is small and most questions in its forums are unresolved.

The Nvidia Issac Sim simulator is an excellent option for the implementation and use of the property-based testing framework, however, based on the initial testing and the noted drawbacks, time is required for the Nvidia Issac Sim to mature as a system. For these reasons, the current property-based testing framework will run on Gazebo but with a planned expansion to the Nvidia Issac Simulator.

4.2 Re-formalization: Property-Based Testing

The reason for the need for re-formalization of the property definition in the property-based testing framework is to provide clarity of structure and modularity to the properties of the framework. Properties in the original Property-Based Testing framework [10] are defined as a core attribute of an action. In the revised framework instead of directly defining tests for an action or behavior, we use a set of predefined tests (i.e. the composite tests) that when combined are able to test a given action or behavior. These properties are now re-defined and categorized into two groups:

1. Primitive properties
2. Composite properties

4.2.1 Primitive properties

The primitive properties are the base building block of all the testing behaviors. They serve as an information database that can be utilized by the composite property. These properties gather a variety of information about an entity throughout the simulation. Examples of primitive properties are:

- o Spatial-property: Entity's position in the simulation in a given time step.
- o Physical-property: Entity's dimensions and physical information such as mass and inertia.

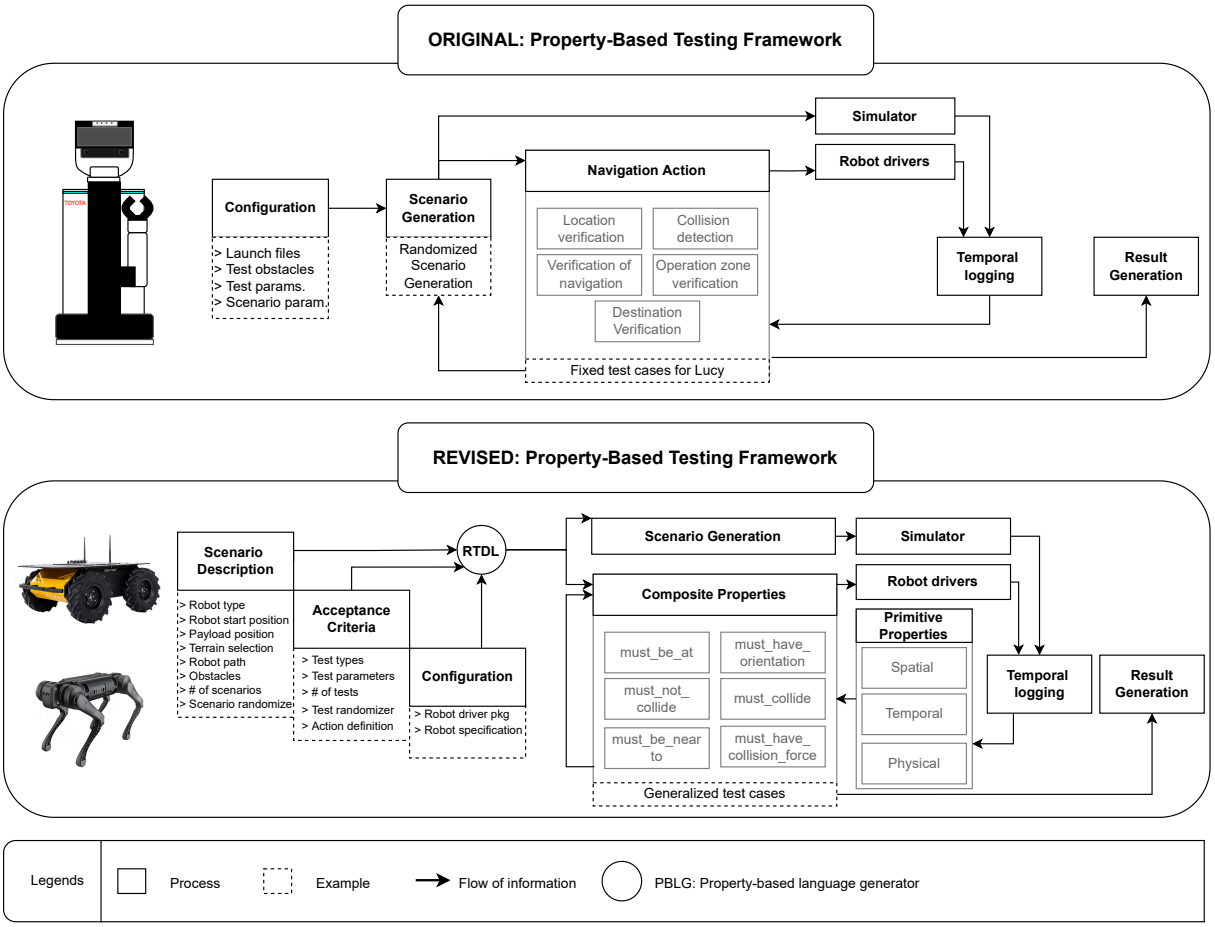


Figure 4.4: The following diagram is a comparative illustration of the original property-based testing framework and the revised property-based testing framework. The key differences are first: The ability to define scenarios and acceptance criteria for ROS-compatible mobile robots and robotic arms (as compared to fixed test cases for the Toyota HSR), and second: Defining and testing an action or behavior via modularized composite tests using a domain-specific language named as the **Robot Test Definition Language** (as compared to writing fixed tests for the testing behavior of a specific robot). Robot images from MYBOTSHOP [14]

4.2.2 Composite properties

The composite properties use primitive properties to build and verify relations for modeled behaviors. They are instantiated by the DSL which provides the expected modeled behavior. An example of a composite property is the *collision-property*. The collision-property compares the *primitive: spatial-property* and *primitive: physical-property* of two objects i.e. **object A** and **object B** throughout a given scenario. To implement the collision property, we use the DSL.

- robot **must not collide** table

In this example, collision-property compares the *primitive: spatial-property* and *primitive: physical-property* of **object A: robot** to **object B: table** to decipher whether the two objects have been in contact throughout the test scenario using the Axis-Aligned Bounding Box (AABB) method and returns a result.

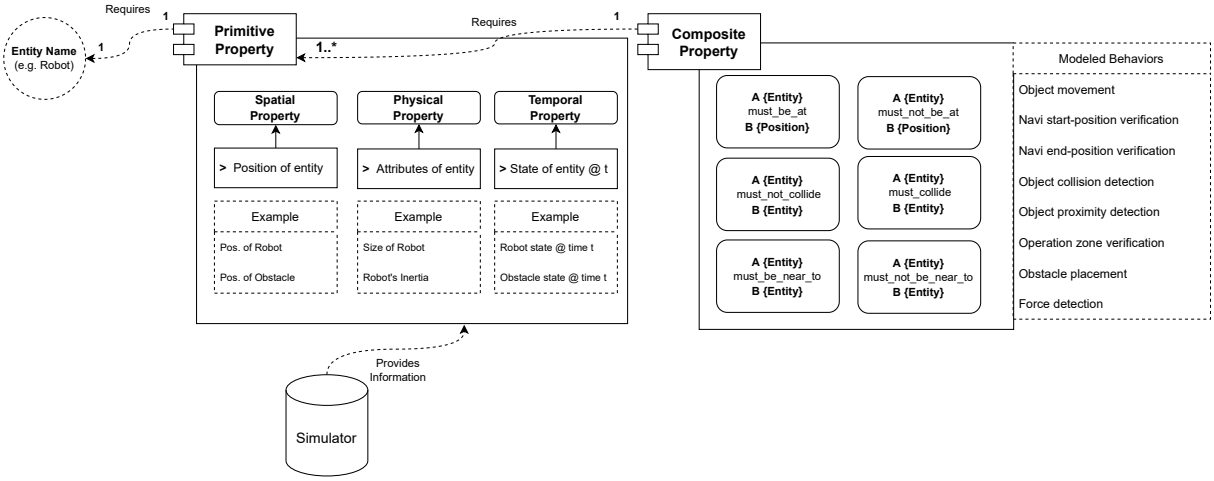


Figure 4.5: Current pool of available primitive and composite properties as well as a set of pre-modeled behaviors.

Discussing some of the composite properties[2] shown in Fig. 4.5. We start with the *must_collide* property. This composite property is used to identify collision between two objects in a given simulation scenario. e.g. when a robotic manipulator must grasp an object, the contacts between the two objects can be verified via this property. Similarly, the *must_not_collide* is used to ensure that no collision has taken place in a given simulation scenario. Typically, it is used to ensure that the robot does not collide with the environment or obstacles e.g. in navigation scenarios or when a robotic manipulator performs a task. An important composite property is the *must_be_at*. This composite property is used to verify whether an entity or a robot is present in a given area (operation zone). It is used for common checks such as ensuring the robot has navigated correctly to its assigned destination. Akin to *must_be_at* is the *must_not_be_at* composite property which is used to verify whether an entity or a robot is not present in a given area (operation zone). It is used to check if the robot has moved out of its operation zone. The *must_have_collision_force_less_than* is used to verify the maximum force exerted by a robot or an entity on another entity. This composite property has been created for compliance with ISO 23481-1 §7.2. The *must_be_near_to* is used to verify whether an entity is in proximity to another entity. It is used for perception tasks verifying whether a perceived entity is within the perception range or not. The *must_not_be_near_to* is used to verify whether an entity is in proximity to another entity. It is used to ensure that the correct amount of safe space is available between the robot and a target entity, additionally, it can be used to verify whether a target entity is within the manipulation range or not.

[2]The workings of the composite properties are explained in detail in chapter 5.

The *must_have_orientation* is used to verify whether an entity has the correct orientation (roll, pitch, and yaw) throughout a given scenario. It is used for checking the stability of either a robot or a payload during a scenario.

4.3 Translating Standards into Test Cases

An initial approach that was queried for translating standards into test cases was by the use of natural language processing techniques in which a text from the standard was passed into a summary extractor and keywords and statements would be extracted which would then define objectives as well as requirements, however, after attempting to translate the standards with an existing summary extractor, key information would be missed consistently. This proved to be a major issue as the nuances of a given standard could not be fulfilled. Due to this issue, other types of frameworks such as meta-languages were investigated.

4.3.1 Queried Translation Frameworks

Metalanguage

A metalanguage is a design language that makes a series of statements that describes the concepts, syntax, format, and grammar rules of a specific language. An example of a metalanguage is the XSL language which defines rules and formatting for the XML language. The context in which we investigate the metalanguages is for the creation of domain-specific language.

Domain-Specific Language

A domain-specific language (DSL) is a language that prescribes a set of rules to describe, concepts, format, grammar, and functionality for a **distinct domain** e.g. HTML is a DSL for the web application domain. It is rich in its features and provides all the required functionalities to program and operate within the web domain. Similar to HTML, a DSL is required that is capable of creating generalized test cases using the Property-Based Testing (PBT) framework so that robots can be tested for robotic standard compliance as well as user-defined tests.

Investigated Metalanguages

In order to retain key information from the standard and create a DSL, several *Metalanguages* were investigated:

- ANTLR [42]
  - **ANother Tooling Language Recognition** is a powerful metalanguage for formalizing and processing ordered text and is capable of translating the processed text into languages such as JAVA, C++, and C#. It achieves this by using a lexer to extract information from a given text and then generating a parse tree (an ordered graph) allowing the user to easily define their own domain-specific language.

- Eclipse Xtext [43]
  - **Xtext** is a type of extension of ANTLR and is used for the development of domain-specific languages. It performs all the functionalities of ANTLR and more. Examples of some of its extended functionalities are grammar checks, inbuilt integration in the ECLIPSE editor, editable parse trees, multi-platform support, etc.
- TextX [44]
  - **TextX** is a metalanguage developed to create domain-specific languages for Python. Its name as well as functionalities are influenced by Eclipse's Xtext. Although not as rich in features as Xtext, it provides all the necessary functionalities required to create a domain-specific language and provides well-structured and clear documentation.
- Flex [45]
  - **Fast lexical analyzer generator** is a lexical analysis software that analyzes text files and identifies patterns defined by the user. It is an efficient tool, however, it does not offer full support for the integration of different programming languages such as Python. Additionally, the community and support for Flex is considerably less as compared to the aforementioned software.

Based on our analysis, TextX was selected due to its interoperability with the existing Property-Based Testing (PBT) software stack. Additionally, it provides refined documentation, ease of use, and flexibility as compared to the ANTLR and Flex. The name of the domain-specific language that is modeled using TextX is Robot Test Definition Language (RTDL).

4.3.2 Robot Test Definition Language (RTDL)

RTDL is a domain-specific language in which the standard definitions are restructured into a form that retains the original information conveyed by the standard as well as being able to be interpreted by the Property-Based Testing framework. It acts as the preliminary step in the automation of standard compliance testing and will be used to model the expected behavior in property-based testing. The RTDL utilizes TextX [44] a meta-language which is a software modeling tool for domain-specific language in python. An example of the RTDL is provided in the overview Fig. 4.6

TextX [44] creates a metamodel[3] in run-time and uses the Arpeggio [46] parser. The metamodel contains all the data on the model language defined by the tester i.e the *grammar rules* and it uses that language to dynamically construct Python objects as can be seen in Fig. 5.5. Detailed information on the working of TextX is available in [44].

[3] A metamodel provides a series of statements that explains the rules, syntax, constraints, and relationships required for the construction of a model/language.

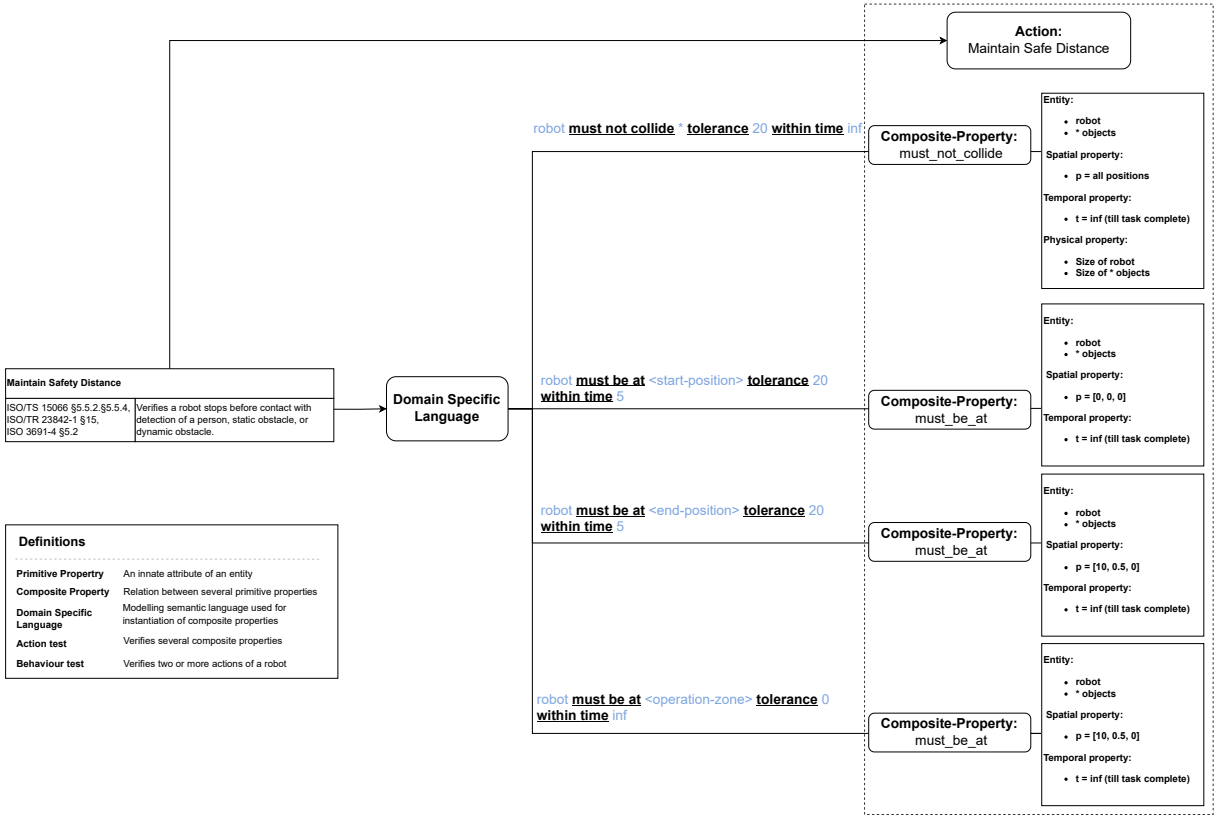


Figure 4.6: This diagram illustrates the initial framework for the formulation of property-based tests from a given custom test. The flowchart represents the framework and the flowchart on the right is an example of applying the framework.

4.4 Test Commonalities of Identified Robotic Standards

As discussed in chapter 2, a description is provided of the common tests identified in the various standards.

#	Standard	Description
1.	**Maintain Safety Distance:** ISO/TS 15066 §5.5.2, §5.5.4, ISO/TR 23842-1 §15, ISO 3691-4 §5.2	Verifies a robot stops before contact with the detection of a person, static obstacles, or dynamic obstacle.
2.	**Static Stability:** ISO 3691-4:2020 §5.3, ISO 23482-1:2020 §11, §13	Verifies a robot's stability when carrying an object on an inclined surface in all types of positions and orientations.
3.	**Dynamic Stability:** ISO 23482-1:2020 §12	Verifies a robot's stability when carrying an object that has a variable load on an inclined surface.
4.	**Limit Range of Movement:** ISO 10218-1 §5.6, §5.12, ISO 23482-1:2020 §15.2	Verifies that a robot does not move faster than the rated speed nor does it enter a restricted space.
5.	**Person Detection:** ISO 3691-4:2020 §5.2, ISO 23482-1:2020 §18.2	Verifies a robot stops before contact with a cylindrical test piece at different locations with different orientations. Requires at least three iterations.
6.	**Response to Ground Obstacles:** ISO 23482-1:2020 §15.1	Verifies a robot's stopping distance and time when obstacles are in different positions and orientations.
7.	**Safety of Localization and Navigation Stack:** ISO 23482-1:2020 §16	Verifies a robot's movement in a given point-to-point navigation task, e.g. sporadic movement, irregular stops, or potentially hazardous path planning.
8.	**Robustness of Autonomous Actions:** ISO 23482-1:2020 §17	Verifies robustness of a robot's decision-making, specifically the course of action during object detection.
9.	**Speed Rating:** ISO 18646-1:2016 §5, ISO 23482-1:2020 §15.2	Verifies if a robot moves at a *rated speed* with acceleration, de-acceleration, and constant speed.
10.	**Stop Rating:** ISO 18646-1:2016 §6, ISO 23482-1:2020 §15.1	Verifies a robot's stopping distance and time when moving in a given *rated speed*.
11.	**Obstacle Detection:** ISO 18646-2:2019 §6, ISO 23482-1:2020 §17	Verifies if a robot can detect six static objects with different orientations at the minimum and maximum range detection range declared by the manufacturer with respect to its line of sight.
12.	**Obstacle Avoidance:** ISO 18646-2:2019 §7, ISO 23482-1:2020 §16	Verifies if a robot can avoid dynamic obstacles with varying trajectories while navigating toward a target position.

Table 4.1: Identified prominent and frequently used robotic standard tests. Source [1, 4, 6–8, 17]

It can be observed that ISO 23482-1 is a common framework that contains tests from other standards

based on the complied information of frequent tests presented in Table 4.1. As a result, ISO 23482-1:2020 is selected as the primary research subject for robot standard testing that is used for assessing robot behavior compliance.

4.5 ISO 23482-1

ISO 23482-1 [1] is an informative report that elaborates on the testing of safety methods in ISO 13482 [36] for personal and service robots. It provides guidelines on the design and utilization of safety test cases. All the tests provided in this technical report cater to real-life testing.

As our proposed testing framework is property-based testing which leverages simulation for verification and validation, we shall be selecting those standards that can be verified via simulation. Table 4.2 shows which safety tests have been selected[4]. As ISO 23482-1 is designed for real-life use, several of its safety-related test criteria cannot be replicated in simulation due to the selection of our simulation platform (i.e. Gazebo) as well as the current simulation technology limitations. An example of this is §14 which states the testing of the safety control in the event of electric discharge when integrating the electro-sensitive protective equipment (ESPE).

Currently, the only feasible way to perform these safety checks is to create dedicated simulation software for these niche cases. §1-§5 provides a description and general information for IS0 23482-1 and is not included in our test cases.

4.5.1 ISO 23482-1: Feasibility Analysis

Test Type	Robot	Selected	Reason
Physical Hazard (Voltage) §6.1	Universal	*X*	Simulation limitation. The current robotic simulators are unable to measure the effect of electrical disturbances on the robot.
Physical Hazard (Acoustics) §6.2	Universal	*X*	Simulation limitation. Although some work is available on the utilization of acoustics in robotic simulators such as [47], most of them are for performing underwater localization and are not available for detecting acoustic interference of robots on humans.

[4]A note should be made on the difference between unmanned ground vehicles (UGV) and autonomous ground vehicles (AGV) in the standard tests.

Physical Hazard (Surface Temp.) §6.3	Universal	*X*	Simulation limitation. Thermal vision systems are available for robotic simulators, however, they do not compute the effect and damage of temperature on the robot.
Physical Hazard (Collision-Injury Parameters) §7.1	Mobile Robot	*X*	Simulation limitation. Robotic simulators do not currently have a human dummy that is able to detect the magnitude of injury on collision although it is possible to create a multi-link body with each link containing a sensor that individually measures the force of impact and then computes the estimated injury value, however, it is out of the scope of this thesis.
Physical Hazard (Collision-Force Control) §7.2	Mobile Robot	✓	A specialized test bench can be created for testing the force of the collision. It will be included in the test case.
Physical Hazard (Restraint Type) §8	Physical Assistant Robot	*X*	Simulation limitation. Similar to §7.1, simulators do not have modules that are able to measure stress and strain with a human body (dummy).
Endurance (Temp. and Humid. Fluctuations) §9.1	Universal	*X*	Simulation limitation. Current simulators are unable to measure the effects of corrosion, erosion, temperature, humidity, pressure, and vibration on robots.
Endurance (Locomotion) §9.2	Mobile Robot	*X*	Simulation limitation. Although work is being done to integrate this feature in upcoming simulators, current robotic simulators are unable to compute deformation and fractured robotic components during movement.

Endurance (Collision) §10	Mobile Robot	*X*	Simulation Limitation. Similar to §9.2 as the simulators cannot compute the deformation and fracturing of the robot, hence, they cannot measure its durability and resistance to such events as well.
Static Stability §11	Mobile Robot	✓	Current simulators provide accurate physics simulation, hence, this will be included in the usable test cases for the user.
Dynamic Stability (Moving parts) §12	Mobile Robot	✓	Simulation is possible, a custom obstacle that has a weight equal to the maximum payload of the AGV will be mounted as defined by the standard.
Dynamic Stability (Travel) §13.1	Mobile Robot	✓	Simulation is possible and verifies that placed weight or obstacle does not detach from the robot during movement.
Dynamic Stability (Travel-flat surface) §13.2	Mobile Robot	✓	Test is possible as verification is done on a placed obstacle with 80% maximum acceleration.
Dynamic Stability (Travel-inclined surface) §13.3	Mobile Robot	✓	Specialized scenarios will be created with different inclinations to test the robot's stability and braking capabilities. Has several sub-scenarios.
Dynamic Stability (Travel- steps and gaps) §13.4	Mobile Robot	✓	Specialized scenario with a custom model that verifies if AGV can move over given steps (i.e. staircase).
Safety-Control (Electro-Sensitive Protective Equipment) §14.1	Universal	*X*	Simulation limitation. Electro sensitivity its effects on robots during operation cannot be measured in the current robotic simulators.
Safety-Control (Slippery Environment) §14.2	Universal	*X*	Water slippage is inaccurate in Gazebo but is accurate in Nvidia Issac Sim. However, the main testing is done via Gazebo so this can be left as future work.

Safety-Control (Electromagnetic Immunity) §14.3	Universal	*X*	Simulation limitation. Similar to §14.1 current simulators are unable to measure electrical disturbance on robots.
Response to ground obstacles (Protective stop) §15.1	Mobile Robot	✓	Testing safety features implemented for the navigation stack (primarily for stopping). This standard test requirement has twelve to thirteen subscenarios that have to be implemented.
Response to ground obstacles (Distance and speed) §15.2	Mobile Robot	✓	Similar to §15.1, however, testing monitors the speed and distance of the AGV from a given obstacle. Several sub-scenarios are available.
Response to ground obstacles (Stopping before convex terrain) §15.3	Mobile Robot	✓	Testing is possible and requires the creation of a custom world with convex terrain to monitor the behavior of AGV.
Response to ground obstacles (Stopping before concave terrain) §15.4	Mobile Robot	✓	Similar to §15.3 but requires concave terrain creation and has several subscenarios to test.
Safety-related (Localization and Navigation) §16	Mobile Robot	✓	Test is possible as we monitor the AGV's response behavior while it is performing localization and navigation in a specified scenario.
Reliability (Autonomous decision making) §17	Universal	✓	Open-ended user tests to verify navigation and autonomous behavior such as object detection and reaction.
Safety-operation (Connection/Disconnection/Reconnection) §18.1	Universal	*X*	Hardware dependent as typically when the connection is cut, the commands continue to run over the network (i.e. ssh) and the robot would continue to perform its task unless they have a specific routine to monitor the connection protocol between the master PC and the robot PC. It would prove difficult to simulate and integrate this type of test into the testing framework.

Response (Multi-command/Unintended command) §18.2	Personal Care Robot	*X*	Testing with multi-control units is possible by sending standard commands and specific robot commands to check if it is complying with both of them in which case it would fail the designated test. However, this does not test an action or behavior of a robot due to which methods such as Xie [30] are more suited to testing this.
Safety-operation (Communication loss) §18.3	Personal Care Robot	*X*	As with §18.1, it would be difficult to replicate loss of control commands in the event of a cable disconnection or loss of connection wirelessly.

Table 4.2: Selected safety-tests from ISO 23482-1 [1]

4.5.2 ISO 23482-1: Requirement Analysis

The standard ISO 23482-1 provides an outline of how to perform the tests for different types of robots, however, it leaves judgment on the user of which test should be performed and how the test should be performed. It does provide examples to aid the users in the implementation of the tests. In terms of the need for requirement analysis, the standard ISO 23482-1 provides the description for test procedures in natural language e.g. "The robot is placed in the directions that represent worst-case conditions e.g. forward, backward, and sideways to perform b) and e) turns."§12 [1]. A human can interpret this text and perform the tests, however, for it to be machine-readable the tests need to be precise, clear, and well-explained. Many things are left to open interpretation in the text. To provide a clear description transferable to the property-based testing framework, we have devised a five-step procedure similar to what is shown in Fig. 4.1 in which an entire use case can be defined which are:

1. **Objective** The objective identifies the purpose of the test. It is not used by the testing framework to test the robot but is used to provide insight into the design choices and parameter selection of the test components.

2. **Target Platform** The target platform is used by the testing software to determine whether a mobile robot is being tested or a robotic arm is being tested.

3. **Required Components** The required components list out the 3D assets as well as robotic hardware components required to conduct the tests e.g. LiDAR for obstacle avoidance. This information is provided by the robot manufacturer.

4. **Scenario Description** The scenario description provides concrete information on what the robot will exactly do or is expected to do in a given test scenario.

5. **Acceptance Criteria** The acceptance criteria provide information on the required conditions for a robot executing a scenario to be deemed successful.

4.5.3 ISO 23482-1: Test Definitions

For each of the selected components of standard 23482-1, we have created test definitions that can be viewed in appendix D. It should be noted that we have attempted to keep the test cases as close as possible to the standards, however, some parts have been altered such as in §13, the measurement section is required to view whether the robot stops at the *stop command*, however, we will directly be extracting the robot's position via the simulator eliminating the need of the measurement section in the standard. Similar alterations that preserve the objective of the standard test or for simulation compatibility have been made across other tests as well.

4.5.3 ISO 23482-1: §7.2

The test definition for §7.2 of standard ISO 23482-1 is as follows:

- **Objective:** Force imparted by a robot on a safety obstacle.
- **Target Platform:**
  - AGV
  - Robotic Arms
- **Related Standards:**
  - ISO/TS 15066
  - ISO 13482:2014 §4.3, §5.10.9.1
- **Required Components:**
  - AGV
  - Robotic manipulator
  - Force sensor obstacle
  - Force measuring device
- **Scenario Description:**
  - AGV or robotic arm moving in maximum speed towards force measuring device
- **Acceptance Criteria:**

  - Detecting Force imparted by the robot is less than the values provided in ISO/TS 15066 (currently unavailable)

- **Quantity of tests:**
  - ×1
- **Variation in scenarios:**
  - None
- **Approach:**
  1. Safety obstacle asset creation
  2. Sensor addition into the robots URDF
  3. Scenario implementation
     - Design and implementation of composite property: *MustHaveCollisionForceLessThan*

In addition to the five requirements, we have described a few other aspects such as the number of tests to be performed and the proposed method of approach for testing. The same pattern and structure demonstrated are repeated for the other tests as well and are available in Appendix D.

4.5.4 ISO 23482-1: Translation into Robot Test Definition Language

Once the standards have been analyzed and categorized. Translation and insertion will be done by filling out the pre-defined templates available provided in the Robot Test Definition Language as illustrated in Fig.5.5.

4.6 User Tests

User tests carry on the original concept of property-based testing in which we provide a set of composite properties which define an action (or behavior) and test the robot in different randomized scenarios to detect anomalous behavior or undesired behavior. This test feature will be used in §17 of ISO 23482-1.

4.7 Research Validation

To validate whether the proposed solution to the research questions in Section: 1.2 enables effective automation of standard compliance, two types of evaluation will be performed on several robotic use-cases:

4.7.1 Framework Evaluation

- **Efficiency:** An investigation will be performed on the time required for testing the robots with the proposed framework. This will provide insight into whether it is better to test the robot in the real world or via the proposed framework.

- **Effectiveness:** In effectiveness, the proposed framework will be queried on whether after performing a set of tests on different robots, the framework is able to provide any beneficial insight such as:
  - Software planner faults
  - Innate mechanical design flaws
  - Software driver bugs

  If successful the framework would provide justification for updating the quality of the system.
- **Consistency:** An assessment will be made on the custom tests as well as the standard tests to see whether they generate, perform, and assess user-based tests and standard tests correctly and are able to do so consistently.
- **Compliance:** An assessment will be made to verify whether the generated scenarios by the property-based testing framework are in accordance with the requirements of the standards or user-based scenarios.

4.7.2 Standard Coverage

- In standard coverage, firstly, we will verify how many constituents of the standard can be verified with the proposed framework.
- Secondly, we will assess the time required to translate an existing standard into a usable test case.

4.8 Use-cases

For the assessment and evaluation of the proposed framework, two types of use cases will be investigated, one of which is for autonomous ground vehicles (AGV) and one of which is for a robotic manipulator. In the AGV type use case, a Husky, a Jackal, a ROVO2, and a B1 will be used as the test subject whereas in the robotic manipulator use-case a xARM6 will be used.

4.8.1 Autonomous Ground Vehicles

For this use case, the AGVs will be tested and assessed in different scenarios assigned by the standards as well as the custom tests by the users. Each of the standard scenarios has been explained in Appendix E. The expected outcome is that the AGVs are standard compliant as well as perform their assigned navigation tasks as expected by the user in different scenarios.

4.8.2 Robotic Manipulators

In this use case, the robotic manipulator similar to the AGVs will be tested and assessed in different scenarios desginated by the standards as well as the custom tests by the users. Each of the standard scenarios has been explained in Appendix E (where applicable). The expected outcome is that the robotic manipulators are standard compliant as well as perform as expected by the user in different scenarios. ‘

5

Solution

For the execution of our proposed approach, we built upon the existing Property-Based Testing (PBT) software [2, 10], however, we re-designed and implemented several aspects of the software for use of primitive and composite properties[1]. The reason being was to allow compatibility with different robots. Moreover, the interfaces and data extraction of the PBT have been updated to work with Robot Test Definition Language (RTDL). The task involved the process of integrating, and developing, new software (such as TextX) as well as testing and configuring simulated sensor hardware drivers for the robots to have a viable working use case. The following items enumerate the resources that had to be integrated or modified in our package[2]:

Table 5.1: Resources integrated and configured.

Resources	**Integration & Configuration**
Robots	Clearpath Husky Clearpath Jackal MBS ROVO2 Quadruped B1 Ufactory xARM6
Simulator	LiDAR New 3D assets Force sensor plugin Robotic gripper plugin
Software	Domain Specific Language (TextX) Primitive & composite properties Navigation stack [Movebase] Manipulation stack [Moveit]

[1]It should be noted that the details of the previous framework are briefly discussed in Chapter: 3 and only the new additions have been mentioned in this chapter.

[2]The details on the robots, tools, and environments that have been utilized are provided in the Appendix B

5.1 Tooling Framework Overview

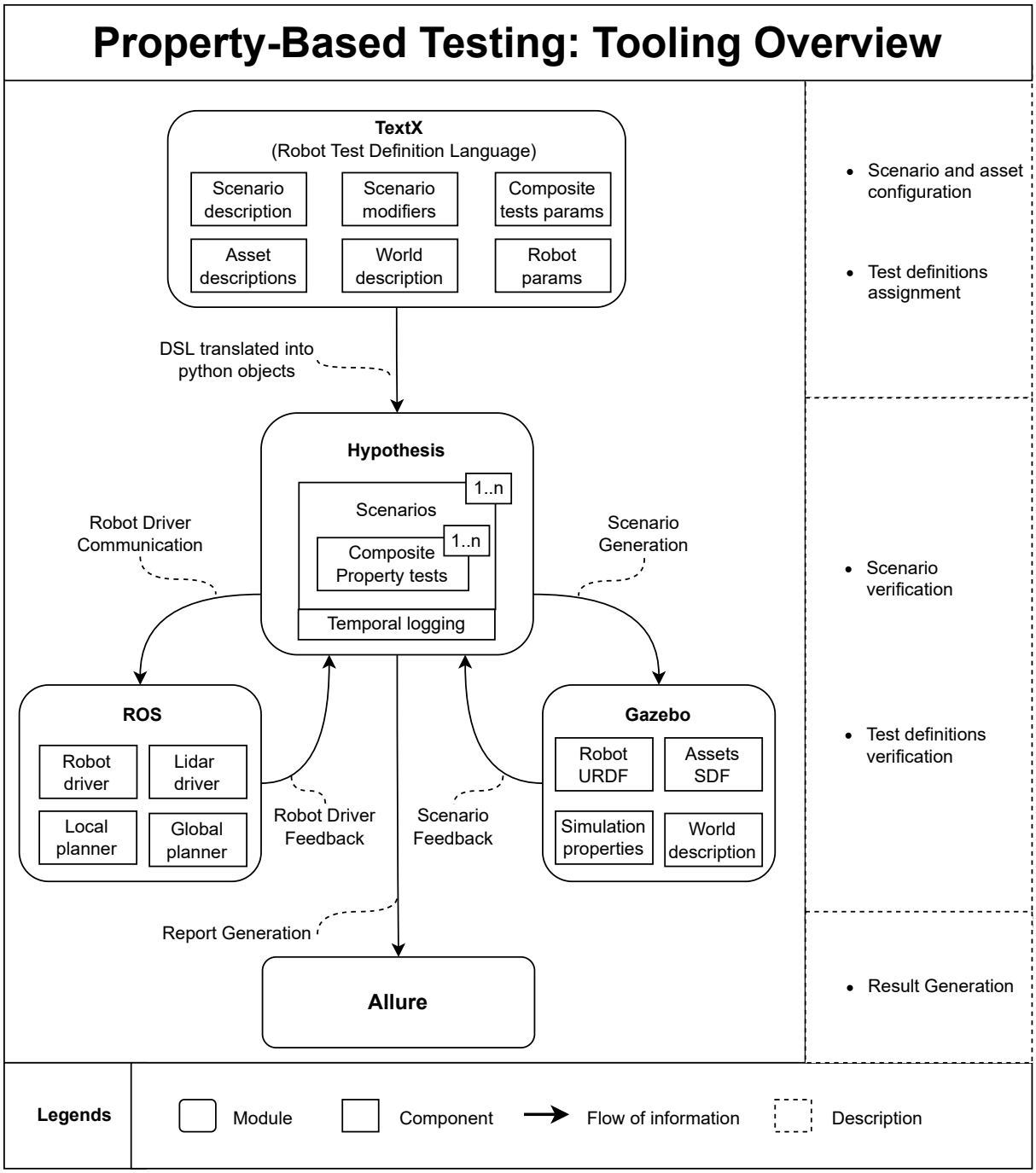


Figure 5.1: A brief overview of the tooling framework for the revised Property-Based Testing framework (PBT).

5.2 Test Configuration

For the execution of the property-based testing framework, a configuration file is provided that locates and links the robot controllers, descriptions, and sensors with the testing framework. This configuration file enables different robots to be loaded and used with the testing framework. It is important to note that the controller launch files have to be modified slightly as they are reloaded for each scenario. The following YAML file demonstrates the launch configuration for the Clearpath Jackal robot.

Listing 5.1: Property-Based Testing configuration for the Jackal robot.

```
---
 Robot:
    robot_urdf_name: jackal_robot # URDF name
    robot_spawner_name: spawn_jackal_wo_controller.launch
    robot_controller: jackal_standalone_controller.launch
    robot_velocity: /jackal_velocity_controller/cmd_vel
    robot_size: [0.990, 0.670, 0.390] # x,y,z in meters
---
```

5.3 Robot Test Definition Language

5.3.1 TextX

For the implementation of our domain-specific language, TextX has been chosen due to its versatility and rich features that it provides as discussed in chapter 4. We start by defining a set of **grammar rules** in TextX that acts as a template for applying the language.

5.3.2 Grammar Rules

The grammar rules are the building blocks of the TextX language. They provide a set of definitions/rules that can be instantiated as described by [44]. Once instantiated, a python object is created which is then passed to the property-based testing framework.

The grammar rules for the robot test definition language mainly consist of two things. The first is the scenario description rules which are used for scenario generation, scenario modification, asset descriptions, robot parameters, and world description. The second is the acceptance criteria rules which uses either single or multiple composite property tests for verification of the scenario. The amount and type of composite properties are determined by the user. When the composite property tests are used in different combinations, they are able to test an action or behavior. It should be noted that is not necessary for the tester of the standard to be familiar with the grammar rules unless they intend to modify or add new parameters for their test cases.

5.4 Robot Test Definition Language: Example

A simplified example of the use of the grammar language is provided below as well as how it is instantiated and the automated python object that is produced.

5.4.1 Grammar Rules

The complete set of rules is available in Appendix E.

Listing 5.2: A subset of demonstrative grammar rules.

```
TestInterface:
  'Test_definitions␣(S)'
    test_type+=TestType
  'Test_definitions␣(E)'
;
TestType:
    StandardScenario | UserScenario | RandomizerScenario
;
StandardScenario:
    'standard' standard = STRING
    section_number += StandardSection
;
StandardSection:
    'section' section = FLOAT
    scenario_configuration = ScenarioConfiguration
    custom_scenario = CompositeProperties
;
```

The Listing 5.2 of grammar rules demonstrate the required information by the Robot Test Definition Language when mapping information from a standard to a domain-specific language. In the listing, *Test_definitions (S)* indicates the starting of the test whereas *Test_definitions (E)* indicates the ending of the test. *test_type* has the + operator which means that the tester writing the test cases requires 1 or more *TestType*. What this means is that the tester has to select one or more types of test. In this listing, the different test types are *StandardScenario*, *UserScenario*, and *RandomizerScenario*. As an example, if the *StandardScenario* is selected, then further information is required information such as the standards section, scenario type, scenario modifier, and composite property tests. The procedure of how other grammar rules are developed is similar to the Listing 5.2. The complete set of rules is described in Appendix E.

5.4.2 Test definition: Instantiating of the Grammar Rule

Listing 5.3: Test definition of ISO 23482-1 §7.2 formalized from appendix: D

```
Test_definitions (S)
    standard 'ISO 23482-1'
        section 7.2
            scenario_configuration
                world_type 'empty'
                description 'Robot force collision measurement'
                robot_position
                    x_pos 0 y_pos 0 z_pos 0.12
                    r_ori 0 p_ori 0 y_ori 90
                manual_speed true
                    robot_speed 1.0 speed_duration 4.0
                safety_obstacle 'safety_obstacle_square'
                    x_pos 0 y_pos 1 z_pos 0
                    r_ori 0 p_ori 0 y_ori 0
            must_collide
                collision_object 'safety_obstacle_square'
            must_have_collision_force_less_than
                force_threshhold 75.0
Test_definitions (E)
```

The Listing 5.3 shows a model conforming to the grammar rules. As discussed in the grammar rules, the *Test_definitions (S)* and *Test_definitions (E)* are for the starting and ending of the test case. A selection of standard is made with the *standard* keyword as well as its *section*. The scenario description is given by the *scenario configuration* keyword in which five features are defined. The first feature is *world_type* which generates the initial environment in which the robot will perform a task scenario. The second feature is *description* which states the scenario process. The third feature is the *robot_position* which describes the starting pose of the robot in the scenario. The fourth feature is *manual_speed* which means that the robot will not use autonomous behavior for the execution of the scenario and finally, the fifth feature is *safety_obstacle* which places a 3D asset in the environment that acts as an obstacle. The next part is the validation which can have 1 or more composite property tests of the same type or different types to verify robot action or behavior. In this case, the *must_collide* is a sanity check to ensure that the robot has contact with the safety obstacle and the *must_have_collision_force_less_than* is the test for ensuring that the robot collides with a force less than the specified value. Also, no particular order of spacing for the test definition is required but the spacing is kept for readability. Other test cases are defined in a similar fashion. This test definition once executed by the RPBT will automatically create, test, and store the results in a database which then can be used to generate an HTML report.

5.4.3 Automatic Python Object

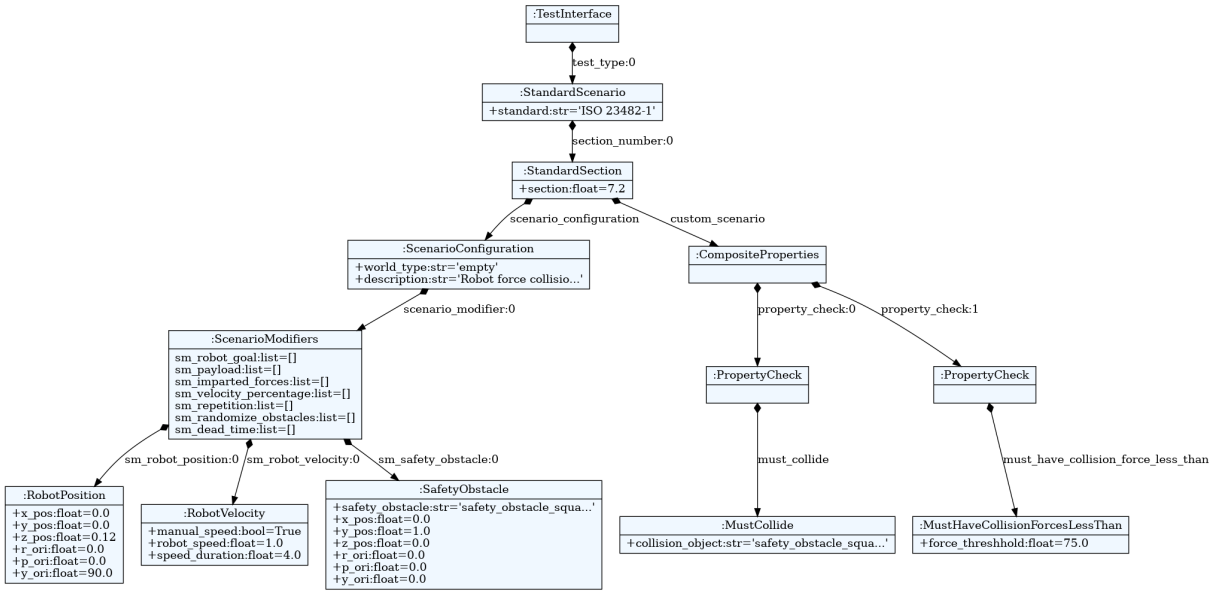


Figure 5.2: The graph illustrates the automatic generation of a python object from the test definition listing 5.3. In the graph, a test interface selects one of three modes, which can be randomized testing, user testing, and standard testing. For this case, the standard test casing is selected along with its specified section. Tools (from TextX) are provided for the user to specify the scenario and the acceptance criteria according to their requirement analysis as shown in Fig.4.1. In the graph, the scenario specification is provided by the **Scenario Configuration** and the acceptance criteria are provided by the **Composite Properties**. Examples for the test definition and their execution are provided in the Github repository.

5.5 Property-Based Test Initialization

The workflow for initialization of the property-based testing framework via the grammar language is as follows:

- The test interface initializes the entire property-based testing software. The test interface allows for different types of tests to be applied which are *StandardScenario*, *UserScenario*, and *Randmoizer-Scenario*. These scenarios can be applied multiple times in a single test run.

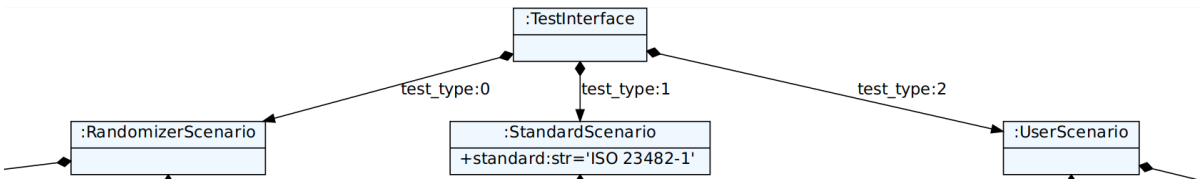


Figure 5.3: The graph illustrates the three modes that can be selected when using the **Test Interface**.

- The standard scenario consists, of four components, the first two components are description of the standard that is being tested as well as the description of the section and bear no influence on the actual test except in helping the user clarify their intent in the tests as well as to enable traceability and reproducibility to the original standard from where it was tested from. Each standard can have multiple sections to test.

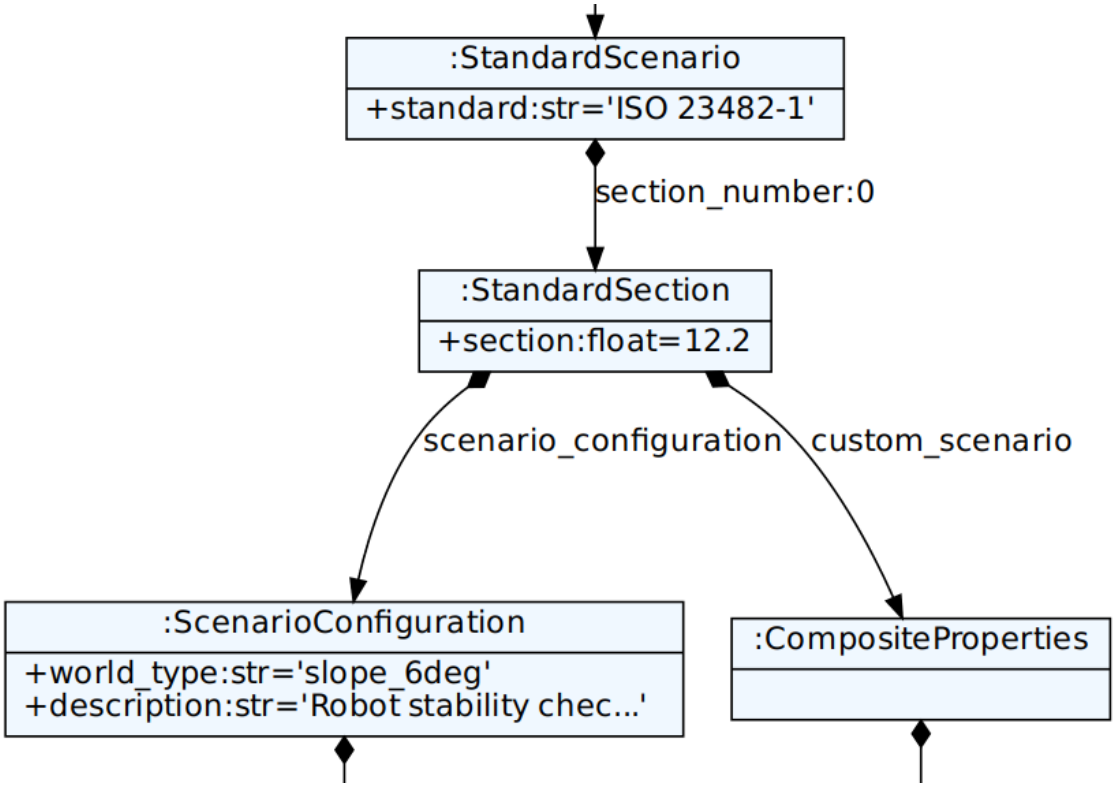


Figure 5.4: The graph illustrates the selection of standard i.e. 'ISO 23482-1' as well its section '7.2'. Currently, the scenario configuration and composite properties are formed by the users according to their requirement analysis. In here, §12 tests the robot's dynamic stability on different slopes that are designated by the manufacturers. For this test **world type** is selected as $slope\_6^o$ because the manufacturer does not provide inclined terrain clearance. The reason for the selection of the $slope\_6^o$ is because it conforms with the requirements of EN 12184:2014 [15] which provides guidelines on dynamic stability for mobile robots.

- The user scenario is similar to the standard-based test but is defined by the user to test for edge cases and known failures.

- The randomizer scenario is designed to generate target parametrized random scenarios to test different behaviors of a robot to test in a variety of scenarios.

5.6 Scenario Generation

In scenario generation, several assets were developed using Fusion 360 for the standard ISO 23482-1. In addition to asset creation, the scenario generation framework has been modified to enable the parametrization of the initial robot placement and final goal pose. Initially, these parameters were randomized but now both randomized spawns and targeted spawns can be utilized. This was an essential change for testing the specific situations as dictated by the standard. The grammar rules for scenario generation are available in appendix E. Explanation on key scenario generation grammar rules are as follows:

- **Scenario configuration** defines the type of environment for a single scenario. The world type is the environment, the description is just a comment for the user's tests, and scenario modifiers can be used to parameterize the tests.
- **Scenario modifier** has optional modifiers which influence the flow of the scenario. Each component is explained below.
- **Robot position** is designed to spawn the robot (either a mobile robot or robotic arm) in the simulator via a given pose.
- **Robot goal** is designed to provide a goal for *move_base* as well as *move_it* ROS packages. For mobile robots, the goals are given as a pose in the odometry frame, and for the robotic arms, the goals are provided as a pose in the end-of-effector frame. Robot goal is also used to provide gripper control for the robotic arms. This is achieved by selecting and passing a unique value to activate gripper mode and selecting the gripper position. Moreover, multi-goals can be provided for both types of robots.
- **Robot velocity** is used when autonomous behavior is not required and is used to test different types of actions such as braking, and high acceleration. Manual speed is a flag to overwrite the Robot's goal. Robot speed dictates the maximum velocity that is to be used and speed duration informs the time period for the manual speed to run.
- **Payload** is used to spawn a 3D asset for either the mobile robot to carry or for a robotic manipulator to pick.
- **Imparted forces** are the maximum force that will be applied to a specific target (ordinarily the robot) in the simulation. The target entity informs of the link to which the force must be applied.
- **Safety obstacle** is used to spawn 3D assets designated as obstacles. It is similar to the payload, the change in name is for clarity when reading the test definitions. Multi-spawns in a single scenario are possible.
- **Dead time** is the time in which the robot is not given any instructions after a goal has been achieved. This is important to verify different robot behaviors, for example when a robot moves

over a very slippery surface, it sometimes slides for a second or two and at that time it may violate an assigned composite-property test.

5.7 Properties

As with scenario generation, several changes have been made to the properties in the property-based testing framework to enable the testing of different robots as well as of the standards while maintaining the capability of randomized testing.

5.7.1 Primitive Properties

The primitive properties collect and store information (such as the position, orientation, inertia, etc.) of all the entities in the simulator during each time step t. The implementation of primitive properties is accomplished by the *Temporal Logging* unit. The process begins when a scenario is generated and initiated by the scenario generation unit. On scenario generation, the *Temporal Logging* unit saves the data (position, orientation, simulation-time) of each entity in the simulator at each time step **t** as a **.csv** file as well as the force collision if available at that time step. The time step is a variable that can be set to any value via our configuration file. In our test cases for the standards, we assigned the value of 0.5 seconds as the *Temporal Logging* unit save interval. 0.5 seconds was selected based on the try-and-hit method. The value was good in terms of performance without losing too much information about the scenario.

On completion of a scenario, the primitive properties would read the **.csv** file and convert it into a data frame via the pandas' library. Moreover, it accesses the 3D models from the **.sdf** and **.urdf** files for the physical information (such as inertia, mass, collision box, etc.). The information saved by the primitive properties is subsequently used in the composite properties to verify test cases for different entities according to what has been defined in the Robot Test Definition Language.

5.7.2 Composite Properties

Composite properties are used to create relations between one or more entities (e.g. robot and an obstacle) by extracting information from their respective primitive properties. When an input value is provided, the composite property is able to verify whether the established relation between entities holds true with the given input value, creating a type of acceptance test. Currently, in the developed software, all the composite properties verify if relations hold true primarily between the robot and a target entity **X**, where the target entity **X** is provided by the user. These composite property tests are translated by the Robot Test Definition Language (RTDL). A constraint is put on the structure of modeling of the RTDL which is that each scenario must have at least one or more scenario and each scenario must have one or more composite property tests. This is a sanity test to ensure that each scenario has at least one verification test. Furthermore, each composite property test may be applied multiple times in the same scenario with different parameters. Following are several composite properties that have been implemented and employed for testing the standards in different scenarios.

MustHaveOrientation

Process MustHaveOrientation requires the spatial information of the target entity to ensure that its roll, pitch, and yaw remain within a designated threshold throughout a given scenario. The information is provided via primitive properties. Simple checks of inequalities are required for verification.

Implementation We implement MustHaveOrientation checks by taking the information of roll, pitch, and yaw at each time step and verifying whether it has exceeded the given threshold.

Grammar Rule A single spotlight example of how the process of the composite property test is realized using the grammar rule.

Listing 5.4: An example grammar rule for the composite_property test MustHaveOrientation. The complete set of grammar rules are available in appendix: E

```
MustHaveOrientation:
    'entity' entity = STRING
    'angles'
    'roll' roll = FLOAT
    'pitch' pitch = FLOAT
    'yaw' yaw = FLOAT
    'tolerance' tolerance = FLOAT
;
```

MustCollide

Process MustCollide requires the information of collision between two entities. The information on the entities during a scenario is provided by the primitive properties. Several approaches can be used for verifying the MustCollide, some of which are the axis-aligned bounding box method [2], the spherical bounding box method, or through a force sensor.

Implementation The approach we have opted for utilizes the gazebo force sensor plugin to detect a collision of the robot with any entity that is in contact with the robot. Previously the axis-aligned bounding box method was utilized but was computationally expensive to perform at each time step due to which the plugin is now the primary method of detecting collisions of the robot with any entity.

MustNotCollide

Process MustNotCollide has the same process of implementation as the MustCollide property.

Implementation Similar to the MustCollide, We use the force sensor plugin to detect whether the robot has not collided with an entity.

MustBeAt

Process MustBeAt verifies a region in the simulator from the world's origin. It requires the position information of the designated entity throughout the scenario. This information is provided by primitive properties. To verify the MustBeAt property, the axis-aligned bounding box method can be used as discussed in [10]. The area is designated via the Robot Test Definition Language.

Implementation We utilized the axis-aligned bounding box method to detect the position of the robot in reference to the provided operation zone in the scenario description.

MustNotBeAt

Process The process of MustNotBeAt is similar to that of the MustBeAt property.

Implementation We have used the axis-aligned box method to verify if the robot is in a non-operation zone.

MustHaveCollisionForceLessThan

Process MustHaveCollisionForceLessThan requires specialized information on the force exerted and imparted on the entities. The method of implementation is via a force sensor plugin, which is either provided by the simulator or has to be calculated based on the mass, inertia, and acceleration of the entities in a collision.

Implementation We use the gazebo force sensor plugin to detect the maximum force applied to the robot during the collision period. It also has the ability to measure torque but for simplicity of testing, torque has not been taken into consideration.

MustBeNearTo

Process MustBeNearTo requires the spatial information of the subject entity and target entity. The information is provided by primitive properties. The euclidean distance at each time step of the subject entity to the target entity can be calculated to ensure that the minimum distance between the two entities is achieved.

Implementation As described in the process, we utilized the euclidean distance between the robot and a target entity to verify whether the robot is within the vicinity of a target object.

MustNotBeNearTo

Process MustNotBeNearTo is similar to MustBeNearto and uses the euclidean distance to ensure that the minimum distance between the subject entity and target entity is not violated.

Implementation Implementation is similar to what has been described for MustBeNearTo.

Summarizing, the current set of composite properties is able to validate robot operation zones, robot collision with the environment, robot collision force, robot slippage, etc. utilizing information acquired from the primitive properties. It utilizes the information by validating relations between different entities by applying different strategies such as measuring the euclidean distance between two entities to ensure that they are not near to each other or by checking the collision of different entities by using the Axis-Aligned Bounding Box method.

5.8 Standard: ISO 23482-1 Tests

To verify the standard with the property-based testing framework, test definitions needed to be created. The test definitions are the instantiation of the grammar rules as described in the section of Robot Test Definition Language.

To create a test definition, three information sources are required. The first information source is the ISO standard itself, the second information source is the robot's manufacturer specification, and the third information source is the user's own input. The information for the robot's manufacturer specification is commonly available from the robot manuals, an example is shown in Fig. B.1. Once the information sources are obtained, a requirement analysis is performed on them in which the objectives, scenarios, assets, and acceptance criteria are concretely defined which has been covered in chapter 4 as well as in appendix D. These test definitions are then written in the domain-specific language (Robot Test Definition Language) which generates python objects that are then passed to the property-based test suite. An example of this procedure is provided of §7.2 of ISO 23482-1 is provided in Fig.5.5. In total 249 test definitions have been developed for mobile robots[3].

5.8.1 ISO 23482-1: Test Formalization

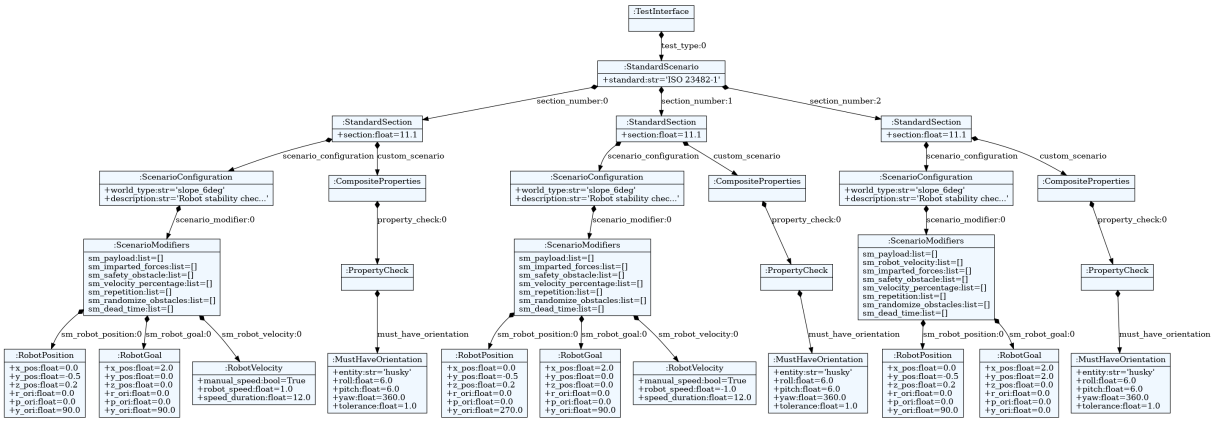


Figure 5.6: The following graphs illustrates the working of 3 of the 92 ISO 23482-1 §12 test definitions. These three tests verify the Husky's dynamic stability by the MustHaveOrientation composite property, the first case being that the Husky moves forward on a slope of 6^o, the second case being that the Husky reverses on a slope of 6^o, and the third case being the Husky turns left on a slope of 6^o. The key differences are in the RobotVelocity for the first two test cases and the goal position for the third test.

The exact translation of the test standards to simulatable test cases proved to be a challenge. The standards are designed to test the functionalities of a robot. However, several of the functionality tests are dependent on human interaction and interference which causes an issue when automating the tests e.g.

[3]It should be noted that the obstacle avoidance for the mobile robots was deactivated for tests from §7.2-13.4 to allow them to perform tests especially while ascending the 15^o platform.

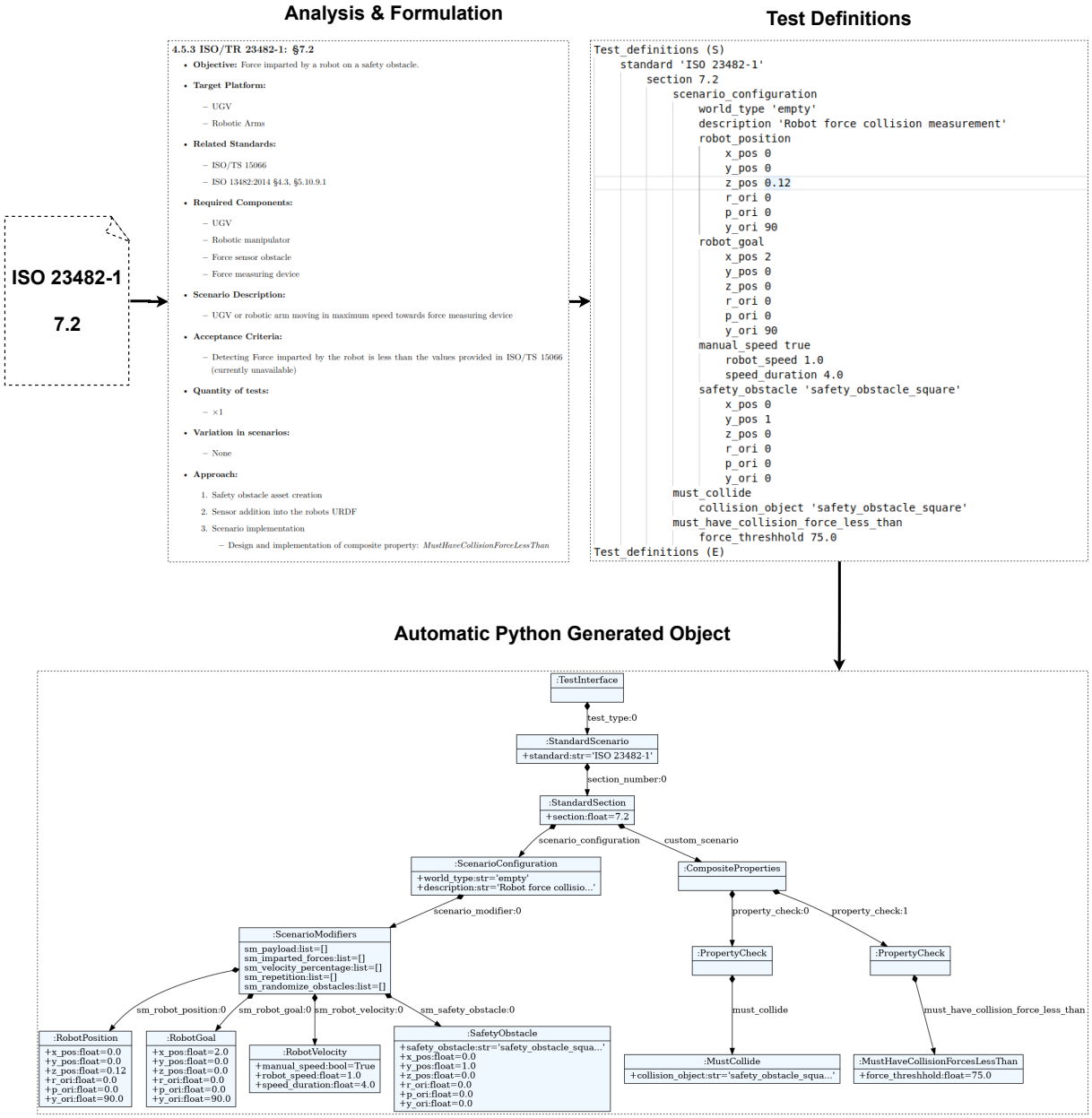


Figure 5.5: The following figure illustrates the implemented workflow for the test definitions of ISO 23482-1 §7.2 using TextX. It covers the first three phases of the methodology in Fig. 4.1 up to the domain-specific language.

§13.2 tests the maximum braking capability of the robot, from the test it is mentioned to test using all types of braking methods including emergency stops (assumingly performed by a human). This causes a hindrance to our proposed methodology as our goal is to test the robots for standard compliance without the human element enabling complete automation, hence, in our approach to translating the standard into test definitions, we've taken a variety of design decisions that enable automation while staying true to the standard as much as possible. These design decisions are further elaborated in the coming sections.

We will now go through the different sections of the standard, and provide information on how we implemented our formalized test definitions (appendix E) from the standards and the reason for our design decisions. It should be mentioned that our DSL is can be used to design and test the scenarios corresponding to the standards but is not limited only to the standards. It can allow the users to test the standards, modify scenarios from the standard, modify the acceptance criteria from the standards, design their own custom tests, or simply perform randomized behavior testing.

The first standard functionality test that we have translated is the standard test of §7.2. It tests the functionality of the maximum applicable force that a robot can apply to an object. The robot itself can be either a mobile robot or a robotic arm. The threshold for the amount of force is dependent on the tester and their criteria (mainly acquired through risk assessment). Several examples of force selection are provided as examples in ISO 15066 [5] and ISO 13482 [36]. Also, several criteria are provided for the selection of the force sensors such as the amount of time that they should activate when an object touches them. Similarly, information on the recommended stiffness of the collision objects is provided. Some things remain ambiguous such as the time to measure the collision force when contact is made. In the formalization of the test definition, we made several design decisions as there was no clarification on whether to perform the test with quasi-static contact or transient contact. We opted to go with transient contact, although going with quasi-static contact is also possible. For implementation utilizing our test definition, we had each robot's URDF extended with a collision sensor plugin from Gazebo. The collision sensor measured the perpendicular force applied to the contact surface. The forces (in N) are recorded during the entire collision sequence and the highest force is saved and passed to the composite property test. The composite property test *MustHaveCollisionForceLessThan* then takes this value and compares it to the provided threshold value.

The second standard functionality test that we have translated is the standard test of §11. It tests the functionality of the static stability of a robot. The robot has to be a mobile robot. The criteria for the selection of the terrain is provided as well as the recommended procedure for the robot placement. In terms of implementation, the *MustHaveOrientation* composite test was developed in which the stored poses of all entities in the simulated scenario (by the primitive properties) are taken and evaluated against a required orientation at time step t. If the robot or target entity violates the given condition, the test fails. §11 has been divided into two segments. The first segment runs the pre-defined scenarios as defined in appendix E without a payload, while the second segment is similar to the first segment except with the addition of a payload and an additional composite property test of *MustHaveOrientation* for verifying the payload's position. For terrain selection, we opted to go for three inclinations [6^o, 9^o, 15^o] designated by ISO 12184 [15]. An option is available to replace the terrain with the user's own custom terrain using

custom 3D meshes. In the future, we plan to add dynamic object terrain generation in our DSL to allow for on-the-spot creation of desired slopes within the language.

The third standard functionality evaluatory test that we have translated is the standard test of §12. It tests the functionality of the dynamic stability of a robot. The robot can be a mobile robot or a robotic arm. However, special test cases are provided for the robotic arm in which forces are applied directly to the arm's end-of-effector. For our implementation we perform all the same scenario and composite property tests as in §11 but with an additional scenario description in which the robot is provided with goal positions (odometry frame cartesian goal for mobile robots and end-of-effector frame cartesian goals for robotic arms) for movement as well as adding randomized forces. An in-built gazebo plugin is utilized for force generation of the randomized forces. A constant force is applied to the main frame of the robot (typically the base_link or the end-of-effector). The force is applied on both the x and y coordinates with randomized values that are limited by the scenario description.

The fourth standard functionality test that we have translated is the standard test of §13.2. It tests the functionality of the dynamic stability of a robot while braking on flat surfaces (slippery flat surfaces). It should be noted that no implementation was performed for section §13.1 because, in comparison to other standards, the assets to be used were mentioned, however, the test formation and execution were not due to which only assets were developed for this test. More on this is discussed in chapter 7. Continuing with §13.2, the criteria provided for testing the braking capability was to test it will all types of braking capabilities, e.g. braking with an emergency stop, with command velocities, and with any other means available. We opted to brake using command velocities, as we wanted to ensure automation of the test scenarios without human interference. For implementation, two separate plane tracks were modeled using the 3D modeling software Fusion 360 [48]. The special feature of these tracks was that they had different friction coefficients designed for testing robot slippage. The light color track shown in Fig. 5.7 has a 0.30μ friction coefficient and the dark color track has a 0.75μ friction coefficient.

The fifth standard functionality test that we have translated is the standard test of §13.3. It tests the functionality of the dynamic stability and braking of a robot on inclined surfaces with different motion paths ensuring it does not move beyond the expected range (The test is applicable only if required by the user). The criteria required the robot to be going at maximum speed and acceleration on an inclined terrain with a maximum slope defined by the manufacturer. For our implementation, we opted to go for the 6^o inclined terrain designated by ISO 12184 [15]. An option is available to replace the terrain with the user's own custom terrain. As mentioned before, we plan to add dynamic object terrain generation to our DSL. A section of the test required that the robot perform a 360^o when ascending and descending the sloped terrain. As no information was provided on this should be done, we defaulted to providing the robot with goals to perform the turn. Although, from the language, it seems the tests must be performed remotely, however, this cannot be confirmed and is left open to interpretation. For the acceptance criteria, no new composite tests were created and the *MustHaveOrientation* test was used. The only addition was new parameters for the test definitions to allow the robot to reach maximum speed to meet the requirements designated by the standard.

The sixth standard functionality test that we have translated is the standard test of §13.4. It tests the

functionality of the dynamic stability of a robot on inclined surfaces when moving on steps and gapped terrain at different speeds. Parameters for terrain height are given as $5cm$ for outdoor robots and $2cm$ for indoor robots if clearance is not provided by the manufacturers. We opted to use the $5cm$ clearance for our tests as some of the robots that we have tested, do not have terrain clearance information. We created two 3D assets for the standard including stairs and a gapped terrain. Parameters for the scenario as well as the acceptance criteria were tuned according to appendix E.

The seventh standard functionality test that we have translated is the standard test of §15.1. It tests the functionality of the protective halting of a robot when faced with an obstacle by non-contact sensors if available. Several obstacles are recommended for testing as shown in Fig. 5.7 if the obstacle lists are not provided by the manufacturers. For the acceptance criteria, it states that motion tracking devices are required alongside markers to visually track the robot and the obstacle, so that the distance and stopped distance can be measured, however, as we are using a simulation, we are able to directly compute the stopping distance without these devices and hence, have not included them nor mentioned them in the test definitions. For the acceptance criteria, we have used the *MustBeNearTo* composite property for measuring the halting distance. Moreover, we added several more tests to detect the halting behavior of the robot when it detects an obstacle while turning based on the §15.1 test design principle.

The eighth standard functionality test that we have translated is the standard test of §15.2 which is similar to §15.1. It tests the functionality of the speed reduction of a robot when faced with an obstacle by non-contact sensors if available. The application of the test apparatus and the procedure is the same as §15.1. The criteria for judging the speed reduction is given in ISO 13482 [36]. For implementation, we altered the composite property test of *MustBeAt* to *MustNotBeNearTo* and we provide the robot with a fixed speed and based on that judged whether the speed has reduced or not using the time step t given by the primitive properties.

The ninth and tenth standard functionality tests that we have translated are the standard tests §15.3 and §15.4. They test the functionality of the braking of the mobile robot when faced with a convex and a concave terrain respectively provided the mobile robot has terrain-sensing capabilities. For both tests, specific approach angles for the robots are designated which are [10^o, 45^o, 90^o] along with the placement of the convex/concave terrain. The convex terrain has special parameters which dictate the height of it being less than the 120% of the robot's traversable clearance height if provided by the manufacturer. The same is applicable to the concave terrain's depth in which the depth should be more than the clearance of the robot. The terrains may be changed and selected by the user to fit a certain condition or robot feature if required. The *MustBeNearTo* has been used for verifying the stopping distance although it should be noted that no pass/fail criteria are provided for these standard tests.

The eleventh standard functionality test that we have translated is the standard test of §16. It tests the functionality of the mobile robots' localization and navigation capabilities. Specialized parameters are provided by the standard test for the environment and the robot goal positions. The environment defined is a rectangular room with the longer side having a length of greater than 6 meters and the shorter side having a length of more than 3 meters. The walls of the room should be high enough that the mobile robot can detect them. It has a square obstacle (1mx1m) placed on one of the longer sides of the room.

Markers are placed in the room if required by the mobile robot. The robot controllers and planners are activated as per their requirements and the robot starts at the corner of the room. The robot is given 5 goal poses which are the four corners of the room as well as one goal being next to the obstacle. The distance of the goal poses must be 150% further from the wall and obstacle to ensure the protective stop is not triggered. These goal points are then moved by a distance of [5cm, 10cm, 20cm, 50cm] and are repeated clockwise and anti-clockwise four times. The pass/fail criteria for this are not clearly stated, instead it is stated that observation is made for unsafe motions, sudden stops, and dangerous movements. As the description of the pass/fail criteria was not explicitly made, we have utilized the *MustNotCollide* composite property for detecting undesired collisions and the *MustBeAt* property for verifying the goal position. Discussion on the potentially implied pass/fail criteria is discussed in chapter 7.

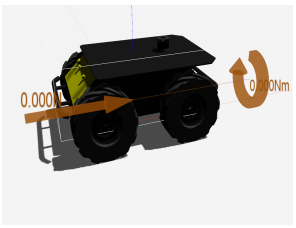
The twelfth standard functionality test that we have translated is the standard test of §17. It tests the functionality of autonomous decision-making. The setup is determined by the user based on the common use of the application and expected decision-making of the robot e.g. not colliding when navigating, stopping on detection of un-clearable terrain, etc. As §17 was a free user-based test, we opted to use the default randomized test generation for assessing the robot's autonomous behavior. All the previous assets and testing techniques from [2, 10] have been used.

5.9 Robot Controllers

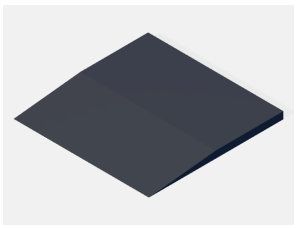
(a) Move_base demo (b) Move_it demo

Figure 5.8: Figures are from MYBOTSHOP [14]. These figures illustrate the *move_base* and *move_it* packages.

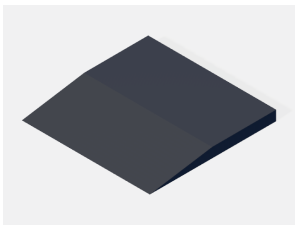
To enable fluid and generalizable control of the robots, ROS software has been used for the AGVs e.g. **move_base** is used to provide and actuate the robot to navigate towards the goal. For the robotic arms, **move_it** software is utilized. To ensure the ideal testing conditions for the robots, no custom or modified planners or drivers were used for the robots. The only addition was a force sensor plugin for measuring the impact force of the robots on an obstacle as well as the addition of LiDARs to the mobile robots to enable obstacle avoidance. An exception to this is for the B1 in which the open source controller CHAMP [49] is used as its simulated hardware drivers are still in development.



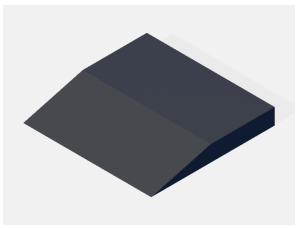
(a) Gazebo force plugin visualization §12



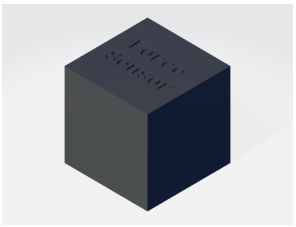
(b) 6^o slope platform §11, §12, §13.3



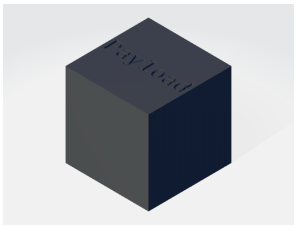
(c) 9^o slope platform §11, §12



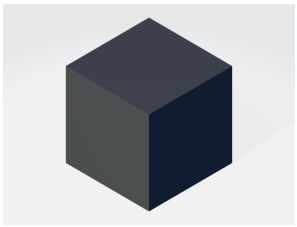
(d) 15^o slope platform §11, §12



(e) Force Sensor Obstacle §7.2

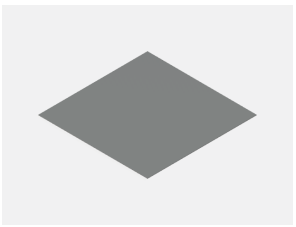


(f) $30Kg$ square payload §11, §12, §13.3

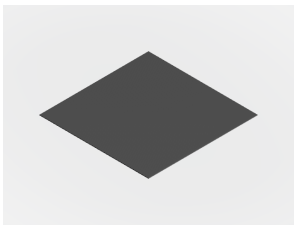


(g) $90x90cm$ Block §15.1 §15.2

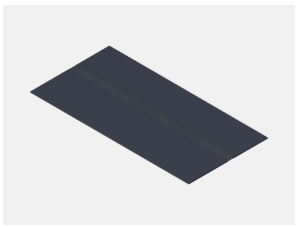
(h) $90x90cm$ Transparent Block §15.1 §15.2



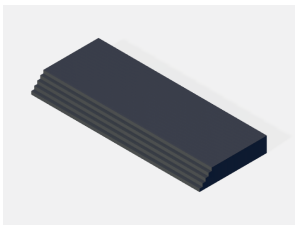
(i) 30μ friction plane §13.2



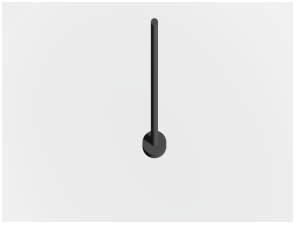
(j) 70μ friction plane §13.2



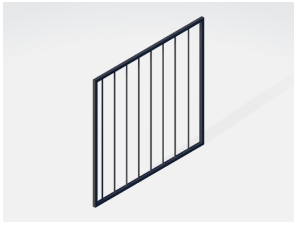
(k) Gap platform §13.4



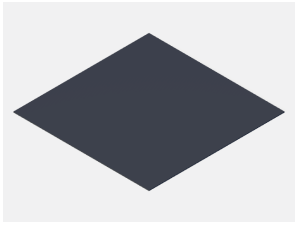
(l) Stairs platform §13.4



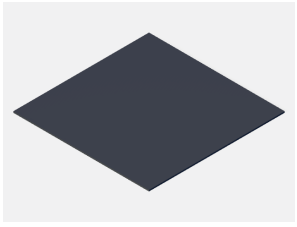
(m) Pole §15.1 §15.2



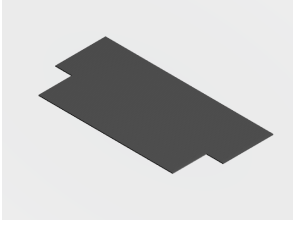
(n) Fence §15.1 §15.2



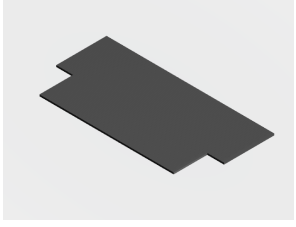
(o) $5cm$ convex plane §15.3



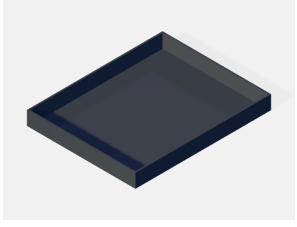
(p) $10cm$ convex plane §15.3



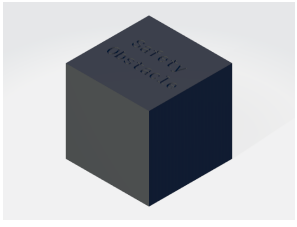
(q) $5cm$ concave plane §15.4



(r) $10cm$ concave plane§15.4



(s) Navigation room §16



(t) Square safety obstacle

Figure 5.7: Generated Assets for ISO 23482-1

6

Research Validation

For the validation of the proposed solution to the research questions in Section 1.2 of whether it enables effective automation of standard compliance, an evaluation is performed on several robotic use cases. All 245 test definitions defined in Appendix D have been evaluated for completeness for the Husky use case which is available in Appendix F. Although all test definitions are applicable to the Jackal, ROVO2, B1, and the xARM6, only the test definitions in which these robots show interesting or unique behavior as compared to the Husky have been discussed in the evaluation. Details on the robots and their software are available in Appendix B.

6.1 Usecase: Husky

The first use case consists of a Husky that performs tasks as per the translated scenario descriptions. The standard controller of Husky has been used without any alteration in most of the use cases unless otherwise mentioned. The order of the evaluation tests goes from §7.2 - §17. For each section, a table is provided which contains the success and failure of the trial runs. Additionally, each section has a sample figure illustrating the scenario types that have been conducted. The evaluation for the Husky is available in Appendix F.

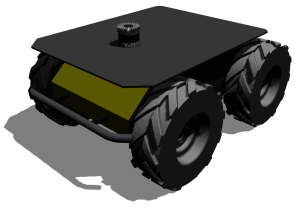


Figure 6.1: Husky [14] Gazebo

6.1.1 Synopsis

The summary of the evaluation performed in Appendix F is described in Table 6.1.

#	Standard §	Pass	Observations
1.	§7.2	20.0%	Several of the tests failed due to the dummy threshold value being placed at nearly the exact value of the maximum collision force of the Husky.

2.	§11	98.7%	Attributable to its bulky design, the Husky succeeded in all of the static stability tests and failed rarely due to physics simulation errors.
3.	§12.1 & §12.2	91.9%	Most of the test cases passed for the 92 test definitions of §12. Some failures occurred due to the robot not being at the designated location by the time it was provided. An interesting observation is that in §12.1.12 (Husky turning left on a 15° slope (ascent)), the Husky moves forward on the slope and then attempt to reverse consistently and lose its initial momentum causing it to slip in several of the §12.1.12 scenarios due to which it does not reach to its final position in time failing the test.
4.	§12.3 & §12.4	96.1%	A randomized force is applied to the base of the robot both in the X and Y directions with a varying force of up to $100N$ in these tests. It would be expected that the majority of the tests would fail, however, due to the bulky and stable design of the Husky, it maintained the correct orientation throughout the test and passed most of the tests. Also, for these tests, the path planner time was increased.
5.	§13.2	98.7%	The Husky successfully braked and did not deviate in the slippage tests. The positive results are due to its large rectangular base and fixed low speed of $1.0\frac{m}{s}$.
6.	§13.3	80.8%	The Husky successfully braked in most of the cases, however, as the braking was part of its autonomous behavior, at times, its path planner took too much time due to which some cases failed.
7.	§13.4	100.0%	The Husky successfully braked and stopped in the designated time in all of §13.4 scenarios. No planners were included for this test due to which errors such as §13.3 did not occur.
8.	§15.1	0.0%	The Husky does not autonomously brake or stop when faced with an obstacle and plans around the obstacle due to which all the tests failed. The difference between braking in this test case and other test cases is that other tests are given goal and brake position at the same location while in this case, the goal position is after the obstacle, and braking is expected before the goal position.
9.	§15.2	90.9%	The Husky successfully reduces its speed when near an obstacle. However, the default planner of the Husky plans too close to the obstacles and commonly collides with the object due to which *MustNotCollide* and *MustNotBeNearTo* have consistently failed.
10.	§15.3	0.0%	No terrain response behavior in the base version of the Husky available due to which the Husky did not reduce its speed or stop at the convex terrains.

11.	§15.4	0.0%	Similar to §15.3, no terrain response behavior in the base version of the Husky is available due to which the Husky did not reduce its speed or stop at the concave terrains.
12.	§16	20.0%	The default planner provided by the Husky package is not tuned, and it generates plans close to obstacles due to which it collides with obstacles, hence, several tests have failed.
13.	§17	-	Most test fail due to the same reasons as in §16.

Table 6.1: Summary of Husky tests from Appendix F.

6.1.2 Effectiveness

The proposed framework was able to identify the following issues in the Husky after conducting the standard compliance tests: A few hardware functionalities are missing for the Husky such as the unavailability of a payload carriage and force collision sensors. In terms of software, several functionalities are not present in the base version of Husky (i.e. standard un-modified Husky). For example, The Husky is able to detect concave terrains but doesn't autonomously stop, change direction, or reduce its speed when approaching the terrain. The same is true for convex terrains. The default local planner of the Husky cannot be used out of the box. Tuning is required for the planner as well as the LiDAR driver to perform normal obstacle avoidance.

6.1.3 Efficiency

In this subsection, the time required for testing the standard in the Husky use case is evaluated. Table 6.2 shows the execution time for the simulated use cases. The total approximate time for conducting the standard tests excluding §17 and the un-simulatable tests ranges from $1hr52min$—$2hr3min$. The variation in simulation execution time is due to one of two reasons. The first reason is that the move-base planner takes a different amount of time to plan its movement. The second reason is due to the rarely occurring simulation physics error in which the robot is unable to reach the designated location and the test simply times out.

6.1.4 Consistency & Compliance

When testing the 2450 test cases for the Husky, only a handful of times did the physics simulator not perform adequately and coherently as can be seen in Fig. 6.2. The reason for this may be due to simulation jumps when the processing of the tests causes the computer's CPU to be overburdened. However, it can conclusively be said that the test generation and execution in general performed were compliant with the standard definitions and were consistent throughout multiple runs.

Table 6.2: ISO 23482-1 standard execution time of a single test for the Husky use cases. However, the execution time is similar for other mobile robots.

#	Section	Robot Type	Number of Tests	Sim. Execution
				Avg. time (sec.)
1.	§7.2	Mobile Robots	1.0	8.0-10.0
2.	§11	Mobile Robots	48.0	440.0.-500.0
3.	§12	Mobile Robots	92.0	900.0-1010.0
4.	§13.1	Mobile Robots	—	—
5.	§13.2	Mobile Robots	8.0	80.0-86.0
6.	§13.3	Mobile Robots	9.0	240.0-280.0
7.	§13.4	Mobile Robots	12.0	90.0-120.0
8.	§15.1	Mobile Robots	32.0	2300.0-2470.0
9.	§15.2	Mobile Robots	32.0	2300.0-2470.0
10.	§15.3	Mobile Robots	5.0	30.0-40.0
11.	§15.4	Mobile Robots	5.0	30.0-40.0
12.	§16	Mobile Robots	1.0	300.0-350.0
13.	§17	Mobile Robots	—	—
			Total Tests: 245.0	Total Time: 6718.0-7376.0

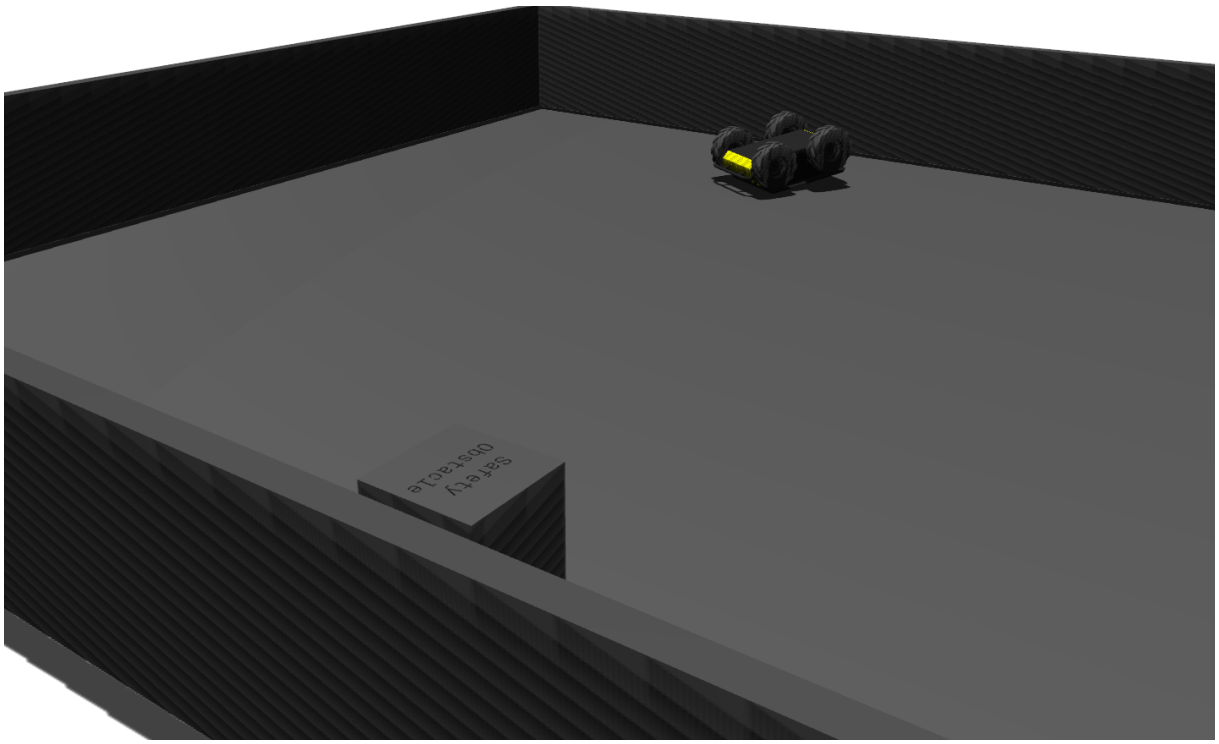


Figure 6.2: The figure illustrates the physics simulation error that causes the Husky to flip over while conducting §16 test.

6.2 Usecase: Jackal

The interesting results from when the Jackal performed some of the ISO 23482-1 tests are as follows.

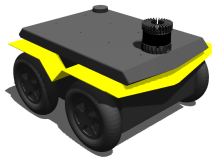


Figure 6.3: Jackal [14] Gazebo

ISO 23482-1:§12

This standard test measures the dynamic stability of the mobile robot. When the Clearpath Jackal performed this test, goal commands were given in the odometry frame for the robot to go either left or right of the robot in some of the scenarios. The robot would however violate the given command and move in the wrong direction.

We ensured that in each scenario of §12 the Jackal had its simulated hardware controller drivers reset and we made sure that at the start of the test it had its odometry at (0,0), however, when testing the Jackal, it still displayed the unexpected behavior and moved completely in the wrong direction. It should be noted that this behavior is in Jackal's native open-source package [50] which is also used in the real-robot as well.

After analysis of the Jackal's movement, we have deduced that the Jackal behaves such that it retains its odometry information when given a command to move autonomously to a given point despite the odometry giving the correct information. When testing the Husky a larger variant of the Jackal which has the same manufacturer and for the most part same software architecture in terms of software planners and configuration, the Husky navigated correctly to the goal position. For both Jackal and the Husky, the exact same test definitions were used. The traceability of the issue is shown in Fig. 6.5. When tested on the real Jackal, the issue did not occur and it appears that the issue originates in the IMU used in the simulated hardware driver. A GitHub issue has been created and submitted to the developers of Clearpath for solving this issue.

ISO 23482-1:§13.2

This standard test measures the braking capability of the mobile robot. Interestingly when the Clearpath Jackal performed this test, the slippage caused by the low friction seemed to be accurate and according to expectations when compared to the real world. The reason is that the Jackal has a maximum speed of $2.0\frac{m}{s}$ as compared to the Husky which has a maximum speed of $1.0\frac{m}{s}$ and relatively is more stable.

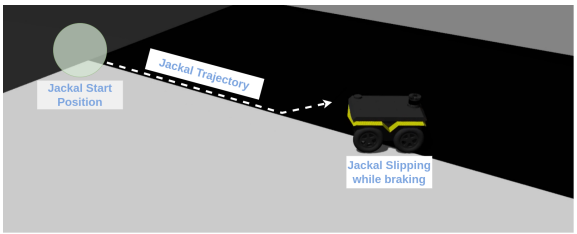
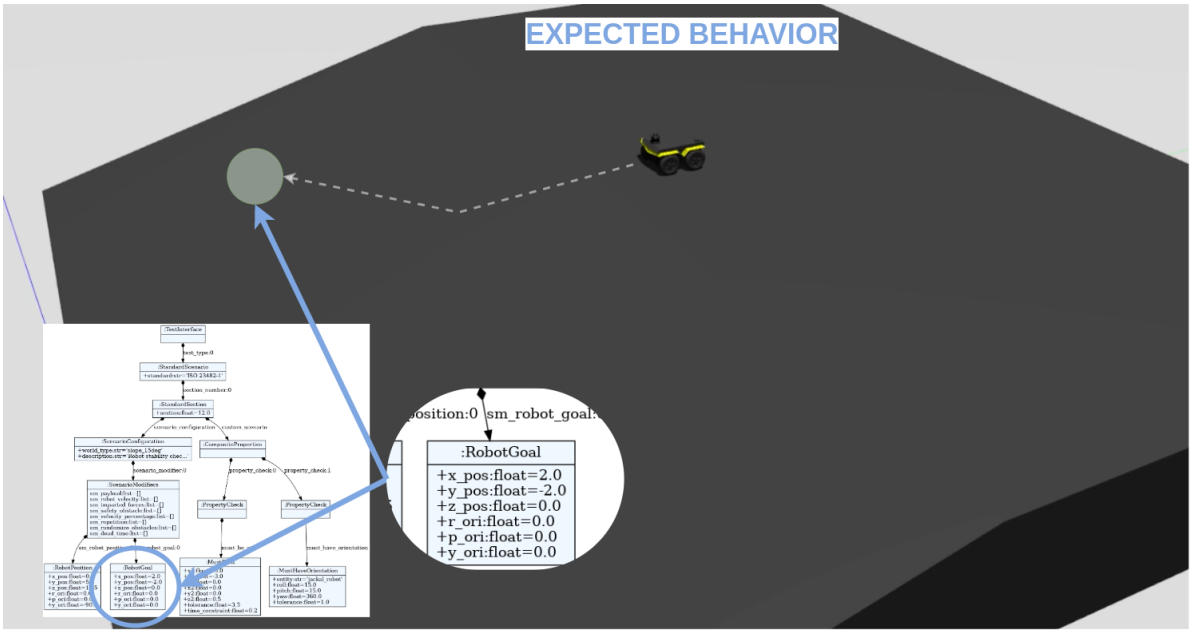
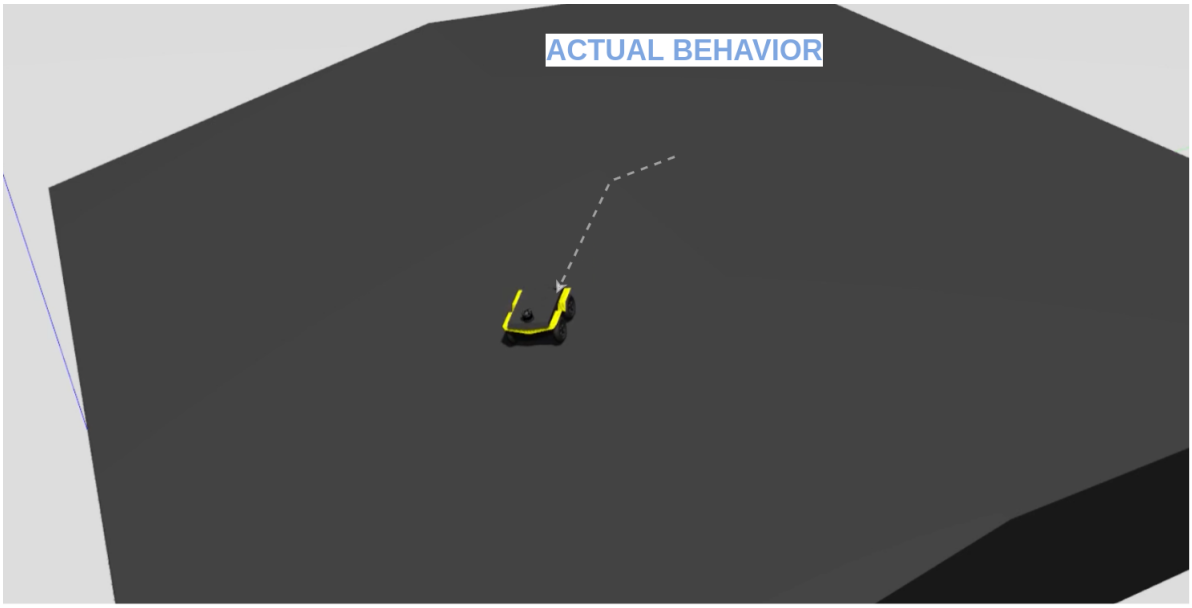


Figure 6.6: Jackal §13.2 slippage

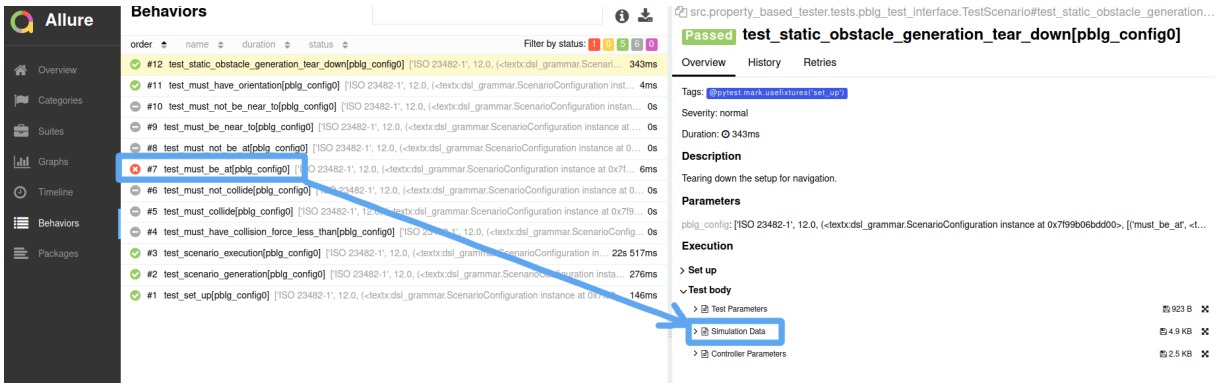


(a) The Jackal is given a goal in a global frame of X:-2 and Y2

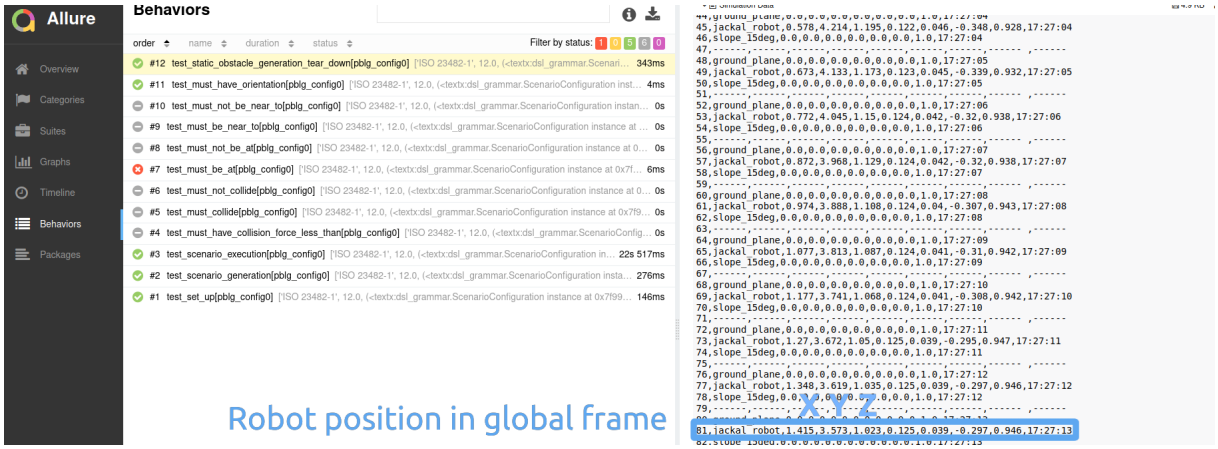


(b) The Jackal moves to X:2 and Y:2 in the global frame instead.

Figure 6.4: The following Fig. illustrates the scenario from ISO 23482-2 §12 which tests the dynamic stability of the mobile robots (Clearpath Jackal in this case). The Jackal scenario is generated using the DSL (robot test definition language) as can be seen in the Fig. In the scenario, the robot is given a goal in its odometry frame of moving 2m forward and 2m right (X:-2, Y:2 in global frame). However, the robot moves 2m forward and 2m left instead of right. The green circle represents the target goal position. The same test definition was provided to Husky (a larger variant of Jackal from the same manufacturer) and other robots and performed the task correctly.



(a) Composite property test **MustBeAt** fails. The data of the scenario is logged in the last test which tears down the scenario's generated assets.



(b) Opening the logged simulation data, we see that the robot has moved toward X:1.4, Y:3.5, instead of towards X:-2, Y:2.

Figure 6.5: The Fig. illustrates a report generated from allure in which the error is traced. The expected position is X:-2, and Y:2 in the global frame, however, the robot has moved to X:2, and Y:2 in the global frame. The **move_base** cuts the goal position if the robot is within the 0.5m in euclidean distance, hence, the values are not exactly X:2 and Y:2 and are instead X:1.4 and Y:3.5.

6.3 Usecase: MBS ROVO2

A single pass evaluation was performed on the MBS ROVO2 (information on the robot is available in the appendix: A). The standard tests revealed a bug in the hardware driver of the ROVO2. The bug was that the robot was unable to move in certain directions. This bug was discovered when performing §12 tests of ISO 23482-1.

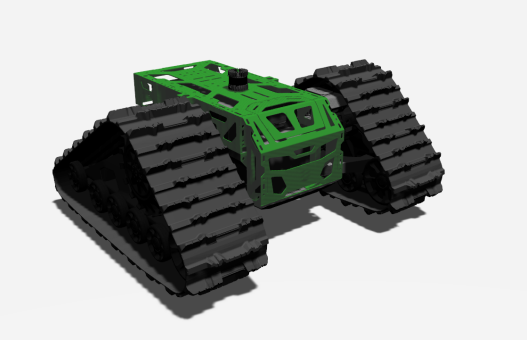


Figure 6.7: MBS ROVO2 [14] Gazebo

An example of the bug is that when the robot is given a goal of moving $2m$ ahead and $2m$ left, the robot would not move, in other words, there were certain ranges of angles e.g. 132^o—134^o where the robot didn't move (dead zones) due to an issue in the driver. The same behavior was present in the real robot as well. This was one of the more intriguing results from the ROVO2 which aided the developers in the MYBOTSHOP organization in fixing and improving their drivers. Other evaluations revealed that the robot required tuning in its local path planner as it would remain in a loop of re-planning when coming near both an obstacle.

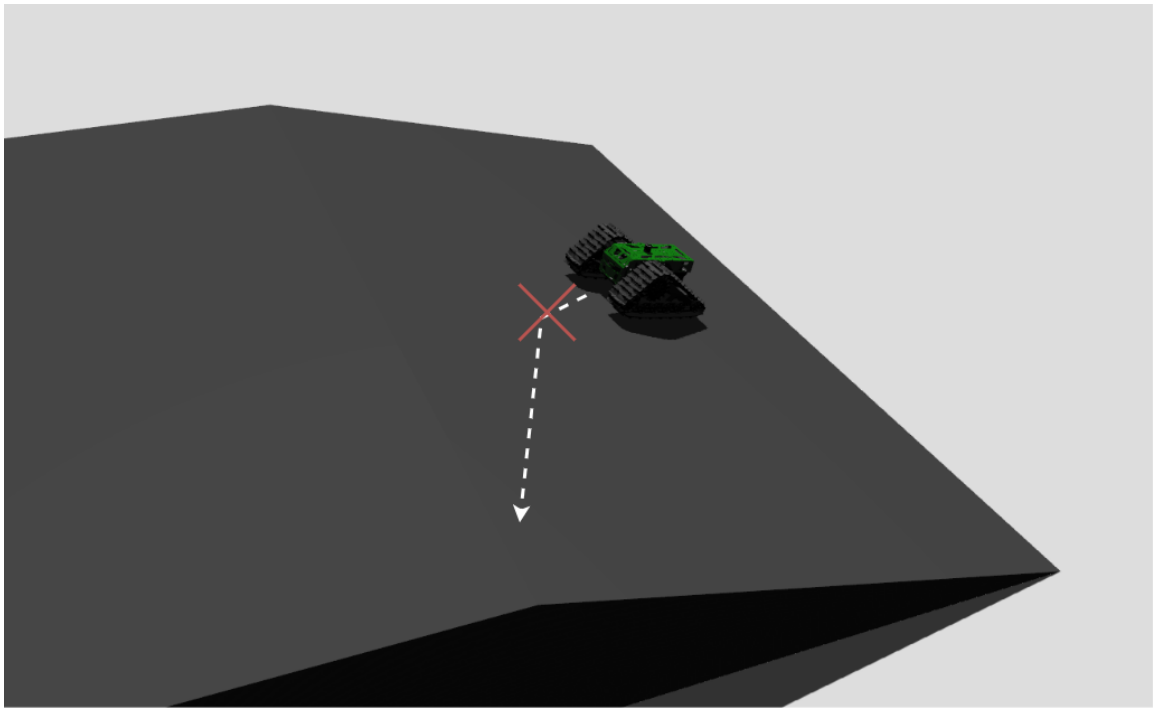


Figure 6.8: §12 ROVO2 dynamic stability test. ROVO2 getting stuck in the dead zone due to a bug in its driver. The white arrow represents its movement trajectory and the red cross illustrates where it doesn't move due to the bug in the driver.

6.4 Usecase: xARM6

For the xARM6, no pre-configured planner was available that could dynamically re-plan around obstacles. Due to this, the robot could only be tested in a few of the standard use cases, examples being §7.2 for the testing transient-static force being imparted on a force sensor obstacle, §12 testing the stability of a robotic arm when a randomized force is applied to its end of effector while it is performing a task of picking and placing an obstacle, and §17 testing the planner of the pick and place of the robotic arm. Several issues from the previous work [10] of the robot being unable to grasp due to simulation errors have been resolved by using gripper libraries ensuring correct grasping of objects. No interesting results were acquired from the tests conducted for the robotic arm.

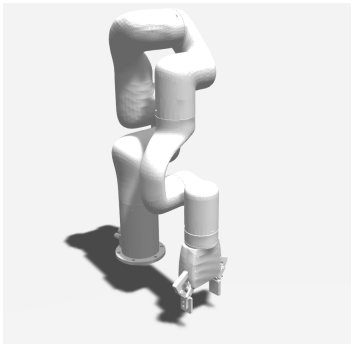
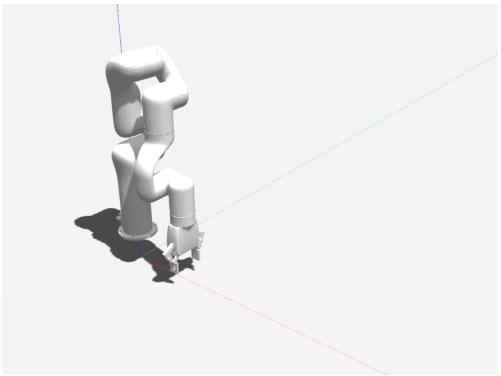
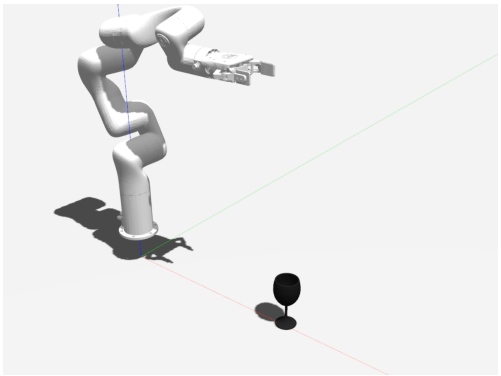


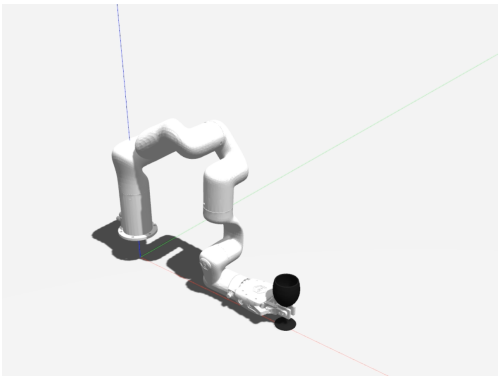
Figure 6.9: xARM6 [14] Gazebo



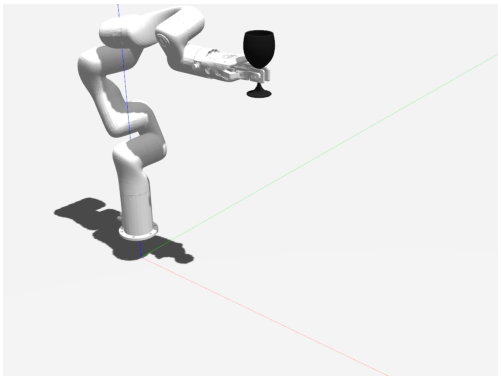
(a) xARM6 home pose



(b) xARM6 pick pose



(c) xARM6 picking wine cup



(d) xARM6 holding wine cup

Figure 6.10: The following figures illustrate §17 compliance tests in which the user is testing if the robotic arms planner can successfully grasp a wine cup without tipping it over.

6.5 Usecase: B1

For the Unitree B1, the open-source quadruped controller Champ [49] has been utilized for movement as the manufacturers controllers are still in development. The B1 has been tested in a few standard tests including §7.2, §12 (a small subset), and §13.4. For §12 interesting results were observed in which the quadruped robot could traverse only up to 9^o of slope. On slopes such as 15^o it could not actually move up the terrain. Additionally, when it attempted to turn in the slopes it would fall over. The reason being that the CHAMP controller does not integrate dynamic stability in the versions that has been used when conducting these tests.

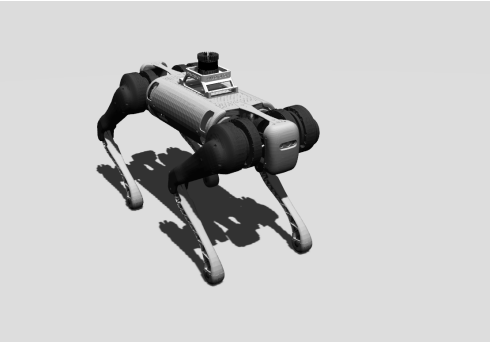


Figure 6.11: Quadruped B1 [14] Gazebo

When conducting tests of ISO 23482-1 §13.4, a B1 controller issue can be seen in which when given a brake command, it would immediately, switch to a static stance causing a jerk motion and causing the robot to fall over[1]. This issue is commonly resolved by allowing the robot to complete its current gait cycle but this feature is missing from the CHAMP controller.

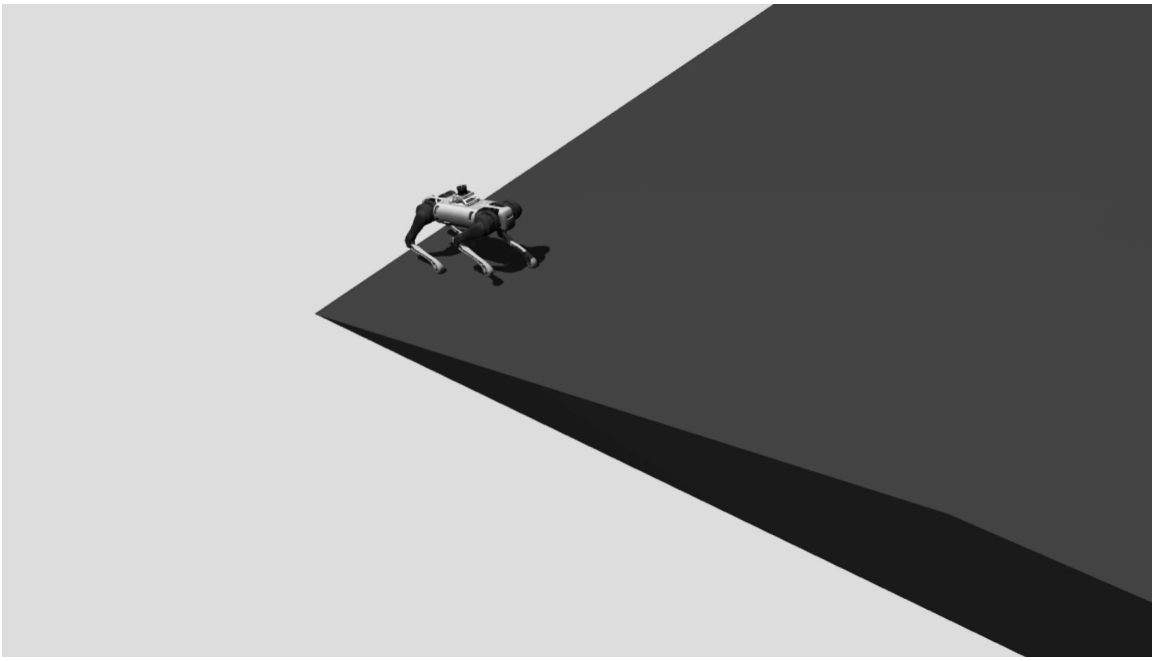


Figure 6.12: §12 B1 dynamic stability test. The B1 is able to move up the 9^o of slope, however, it is unable to perform turns in the slope due its static gaits.

[1]The issue can be seen in the GIFs in the project repository project repository .

6.6 Standard Coverage

In this section, we will evaluate how many standards have been translated and implemented from ISO 23482-1.

Table 6.3: ISO 23482-1 standard formulated and covered for the different robots. Standards the ISO 23482-1 cover are ISO 13482 as well as aspects from ISO 15066, ISO 7176, ISO 11202, and ISO 10218. ✓are for the tests that the robots are able to comply with the standard. - represents tests that are inapplicable as they provide no test procedure. X represents tests that are simulatable but the robots cannot comply with them or fail to comply with them due to missing features (such as halting on detection of concave terrain). O represents tests that are not simulatable. ∞ represents tests that are based on the user and the quantity is up to the user.

#	Section	Formalized	Husky	Jackal	ROVO2	B1	xARM6
1.	§1	-	-	-	-	-	-
2.	§2	-	-	-	-	-	-
3.	§3	-	-	-	-	-	-
4.	§4	-	-	-	-	-	-
5.	§5	-	-	-	-	-	-
6.	§6	O	O	O	O	O	O
7.	§7.2	✓	✓	✓	✓	✓	✓
8.	§8	O	O	O	O	O	O
9.	§9	O	O	O	O	O	O
10.	§10	O	O	O	O	O	O
11.	§11	✓	✓	✓	✓	X	-
12.	§12	✓	✓	✓	✓	X	✓
13.	§13.1	-	-	-	-	-	-
14.	§13.2	✓	✓	✓	✓	✓	-
15.	§13.3	✓	✓	✓	✓	X	-
16.	§13.4	✓	✓	✓	✓	X	-
17.	§14	O	O	O	O	O	O
18.	§15.1	✓	X	X	X	X	O
19.	§15.2	✓	✓	✓	✓	✓	-
20.	§15.3	✓	X	X	X	X	O
21.	§15.4	✓	X	X	X	X	O
22.	§16	✓	✓	✓	✓	✓	X
23.	§17	✓	∞	∞	∞	∞	∞
24.	§18	O	O	O	O	O	O

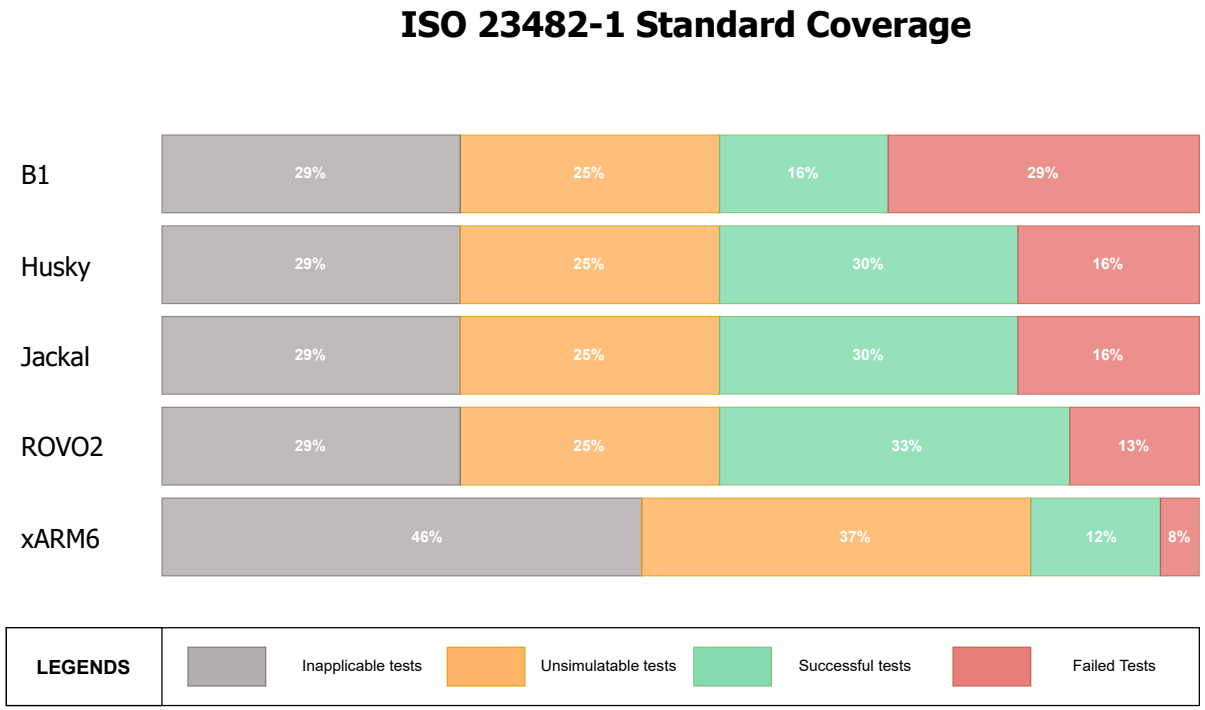


Figure 6.13: This figure illustrates the standard coverage for ISO 23482-1 of the different robots. The gray bar indicates inapplicable tests from the standard that do not provide test procedures but only definitions or recommended environment parameters etc. The orange bar is for tests that test certain functionalities which cannot be simulated e.g. electromagnetic interference on the robot's system. The green bar is for tests that the robot has successfully performed in accordance with the standard e.g. remaining statically stable on a 6^o slope or navigation around obstacles. The red bar is for failed tests in which the robot could not comply with the tests due to a lack of software functionalities. e.g. halting when faced with impassable concave terrain.

7

Results

(a) Clearpath Husky

(b) Quadruped B1

(c) Ufactory xARM6

(d) MBS ROVO2

(e) Clearpath Jackal

Figure 7.1: The robots used for standard compliancy tests. Figures are from MYBOTSHOP [14].

The results of the evaluations performed are that the Revised Property-Based Testing framework (RPBT) identified bugs within the different robot drivers including the Clearpath Jackal and MBS ROVO2. Furthermore, the conducted tests revealed that none of the tested robots were standard compliant and all of them had missing vital hardware and software functionalities. It should be noted that the objective of conducting the evaluation on the mobile robots and robotic arms was to verify the effectiveness of the proposed methodology for validating the standard requirements using natural language (Robot Test Definition Language).

7.1 Discussion

In the evaluation of the robots, on the basis of the standards, the default controllers were used for the robots (except B1 as its controller was under development) with slight changes in their software to enable testing of certain scenarios designated by the standard. These slight changes include the addition of a simulated Ouster LiDAR for obstacle detection and avoidance, a force sensor, and modifications to their launch files. The evaluatory tests performed on the robots cover the standard ISO 23482-1, nevertheless, room for improvement in test definitions remains. A soft certification[1] can be provided for robots that comply with standards with the condition that the scenario definitions are complete and comprehensive, according to the standards, and that the robots themselves are correctly configured.

7.1.1 Highlights

In the evaluation of the commercial mobile robot Jackal, a bug was discovered in its driver. The problem caused by this bug is that, when the Jackal's simulated hardware controller is reset, the Jackal's position should also be reset to the robot's current position and orientation. However, in the case of the Jackal, its position is reset but its orientation in the odometry is not. An example of this behavior is if a command is issued to the Jackal to move $3m$ to the left in its odometry frame and the robot is placed at 90^o orientation as well as its simulated hardware controllers are reset. In this case, it should start by moving $3m$ to the left, but instead, it moves $3m$ to the right. The Husky robot which is a larger robot variant of the Jackal does not share this problem. Furthermore, when tested on the real robot, this issue did not appear and it seems to be an issue in the simulated version of the Jackal only. A GitHub issue has been created and submitted to the developers of Clearpath Jackal in order to solve this fault.

An evaluation was performed on an internal project in the MYBOTSHOP organization, and the proposed software was able to help the developers identify a driver bug in which the robot **MBS ROVO2** would not move when given a command to move in a series of certain directions e.g. 192^o and 194^o degrees. This bug was identified in §12 tests in which the robot was assigned to move in a variety of directions to check its dynamic stability on a slope.

The property-based testing software revealed several missing key features from the tested mobile robots. These features were adequate obstacle avoidance, terrain adaption (not going in a ditch), path planning cancellation, etc.

7.1.2 Challenges

An issue while testing the different robots was that most of the software for the robots that were available was developed for academic and research purposes and, hence, none of the planners that are provided with the default robot software development kits were able to perform satisfactorily in the standard tests for autonomous behaviors.

[1]The term 'soft certification' in this thesis is defined as a certification granted to robots that pass standards tests. However, this certification does not necessitate that the real robots are certified.

To develop and enable generalized behavior testing of different types of mobile robots from different manufacturers is a challenging task. Especially due to some missing software features, features such as resetting the odometry and controllers of the robot when it encounters **kidnapped robot problem**. This issue holds true in simulation as well when switching from one scenario to another. To counter this problem, the simulated hardware controllers of the robots were reset for each scenario in the simulation resetting its odometry. Some robots can be reset without resetting their controllers such as the Turtle bot 3 but the majority of the robots have their odometry reset with controllers, hence to enable generalizability, the hardware drivers were reset.

7.1.3 ISO 23482-1

With regard to the standard ISO 23482-1 [1] analyzed for this thesis: it has several issues, some of the prominent ones being a lack of clarity in formulation, structure, and language. Most of the presented test cases by the standard are open to interpretation. Despite being a standard for testing different types of personal care robots, there is a firm focus on mobile robots and a shortage of test cases for other types of robots such as robotic arms and unmanned drones. Specific issues that were faced for each of the selected components of the standards are discussed below.

Section 7.2

The definition for robots that utilize *force control* is unclear. The vagueness stems from the term *force control* as it is not defined nor is its definition indicated in the text. Moreover, it is unclear where this sensor is used, how it influences the robot when the robot makes contact, and where it should be positioned during a collision.

Force transducers and pressure sensors are mentioned but it is not clarified which one to use and in what cases should one be preferred over the other. Clarity is required on whether quasi-static contact should be tested or transient contact or whether both should be tested in each scenario. Clarification is required on the performance of tests for mobile robots and robotic arms when performing the force collision procedure. It is not explained where the pressure sensor must be placed during the test. Another issue is to decide at which point in time the velocity of an accelerating robot should collide with a force sensor. Should it be measured at the moment of impact, or perhaps when the collision force is at its maximum, or maybe an average of the entire force during the collision time frame? The bullet points presented in Section 7.2 are inconsistent. Despite this section being a procedure on how to conduct the experiment, bullet **b** and bullet **e** seem unrelated to each other and are statements instead of the actual test procedure.

Section 11

In this section, the containment unit for a mobile robot carrying a payload is not defined. The pass/fail criteria defined by the section are based on ”potentially dangerous manner” [1] and ”mechanical instability” (such as falling off). Another issue is the payload definition: it refers to the manufacturer’s information on

the weight of the payload, the placement of the payload, and the terrain on which the robot must carry the payload. However, no statement is given on the size and type of the payload that a robot should be able to carry or be tested with.

Section 12

Issues for Section 12 are similar to those in section 11 i.e. the payload and pass/fail criteria. Moreover, no values, formulation, or concrete methodology are provided to determine whether a maneuver performed by a mobile robot is within an acceptable range or not. Despite the standard being issued for personal care robots, most of the tests are based on vehicle standards and do not prioritize autonomous behavior while testing the scenarios e.g. when testing the stability on a slope the standard mentions moving the robot along the sides and edges, etc. leaving the actuation motion open to interpretation. It may be left out due to the focus on functionality instead of the actuating motion, however, clarification should be made and as this standard is for robotics systems, perhaps more focus should be given to the use of autonomous motion.

Section 13.1

The previous section of the standard provided a consistent style and test structure such as **Principle**, **Apparatus**, **Procedure**, and **Pass/Fail** criteria. However, for this section, the procedure seems to contain information that should be in the apparatus section. Moreover, the procedure does clearly explain how the test is to be conducted. Elements and assets for the tests are mentioned, however, the method of approach for executing the tests is missing (we assume that this section may be a prelude to sections 13.2, 13.3, and 13.4, however, this has not been done in previous sections). Overall, this section is inconsistent, incoherent, and ambiguous and should be revised in future versions to be more in line with the previous structure.

Section 13.2

The braking mechanism for the robot that is to be used is not explained in the standard e.g. disc brakes, electrical brakes, hydraulic brakes, etc. It has been mentioned that all available braking methods should be tested, however, no clarification is made on what is meant by 'braking methods', is it referring to emergency braking, and autonomous braking, or perhaps something else? [2]

Section 13.3

The information on the robot deviating from its designated trajectory path is not elaborated. It mentions from what is assumed to be the pass/fail criteria of this section that the robot must not move out from the "allowed width in its specification" [1]. Assuming by specification, the standard is talking about the robot manufacturers' specification, what should be done in the case when the manufacturers do not provide this information (as it happened with us)? A simple formulation or methodology or even a simple

[2] An explanation is provided of braking methods in §13.3 but it is not certain whether that is applicable to §13.2 as well.

reference should be provided to deal with these discrepancies. Conducting motions tests in which the robot performs 360^o turns are not specified. An example should be provided for the robots that have stability control systems and perhaps a diagram should be provided as done so for the previous sections.

Section 13.4

The presented tests in this section are similar to that of requirements in §15.3 and §15.4. It seems that it can easily be merged into those sections as the tests are similar in nature. The definition for ”maximum power output” [1] is ambiguous in the context of the procedure.

Section 15.1

The structure of this section is very well formed and information is presented in a coherent manner as compared to the previous sections. No pass/fail criteria were provided other than a short mention of ”sufficiently reduced risk of collision” [1] in the principle section.

Section 15.2

The addition of the safety criteria in the Appendix of the cited ISO 3482:2014 [36], Annex C would be a great addition. It is not clarified if the sensor information is taken into account when halting or reducing the speed in both §15.1 and §15.2. The stopping or reduction in speed has not been defined for the robot but instead deferred to the manufacturer's technical specification. In case the manufacturers have not provided any such information what should be done? No alternative methodology or formulation is provided.

Section 15.3 & Section 15.4

Information on what is the recommended distance of when the robot should stop when faced with a convex or concave terrain is not provided. The test is used for testing the sensing capability of the robot yet for stopping it mentions a protective stop, which we assume to be conducted by the human. Further clarification is required on these points. The pass/fail criteria as in the previous sections are not mentioned or explained.

Section 16

The pass/fail criteria are not explicitly defined and examples of what constitutes a failing test case are missing. Terminology such as ”potentially hazardous movements” are used without being explained what they constitute.

Section 17

Finally, The apparatus for conducting section §17.2.2 is missing.

Concluding the discussion on the standard, we find the composition of the standard puzzling. Each section seems to be composed of different authors, who do not use a consistent style or methodology for presenting the tests. Some tests provide a high amount of details with proper language and representations while others are vague, disjointed, and ambiguous. There is also the issue of conflicting language in the standard e.g. whether a robot should stop when faced with an obstacle or reduce its speed.

7.1.4 Applicability for the robots to the standards

The standard can be applied to a variety of mobile robots and robotic arms. In our test cases, we have tested the standard on wheeled robots, tracked robots (which may not be as accurate due to the use of small virtual wheels), quadrupeds, and a robotic arm. As for the question of whether one should apply the standard for testing their robots? We would answer that question as a yes. Despite the ambiguity and incoherence of the standard, it presents an excellent compilation of different functionality tests which may prove to be useful to users, testers, developers, integrators, and manufacturers.

8

Conclusion

In this thesis, a formalization process was developed as shown in Fig. 4.1 using a domain-specific language (DSL) (i.e. Robot Test Definition Language (RTDL)) for translating existing standards into simulated test cases that have been verified by a revised version of the property-based testing framework [2, 10]. The objective of this thesis was to bridge the gap between standard compliance testing and automation.

In order to select a standard to demonstrate the proposed methodology, a survey was conducted in which over 40 relevant robotic safety standards (Appendix C) were inspected based on their technical relevance to the field of autonomous systems and robotics as well as from recommendations suggested by researchers and industrialists in papers and technical manuals. The common tests were compiled from several of the prominent standards (Table 4.1) and ISO 23482-1 was selected as the primary test bench. The reason this standard was selected was due to it being a purely evaluatory standard containing testing information for different types and varieties of personal care robots. Moreover, it encompassed tests for several other standards, such as ISO 13482 ("safety requirements for personal care robots") [36], ISO 15066 (safety requirements for cobots in industrial environments) [5], as well as elements from ISO 11202 (which provides guidelines for acoustic noises), ISO 15066 (for biomechanical collision tolerances) [5], and ISO 7176-13 (for recommended test surface friction coefficients).

An in-depth review and analysis were performed for ISO 23482-1 from which concrete test cases (Appendix D) were created for the revised property-based testing framework from the standard's otherwise obscure documentation. These test cases were then used to develop test definitions via the RTDL. These test definitions were then integrated and tested in various robots such as the Clearpath Husky, Clearpath Jackal, MBS ROVO2, B1, and the Ufactory xARM6. Due to time limitations, a complete evaluation was only performed for the Clearpath Husky (Appendix F).

The result of this thesis was that standard compliance was validated for several robots from the selected simulatable test cases extracted from the standard ISO 23482-1 which revealed a number of missing hardware and software functionalities such as no payload carriage and missing terrain response behavior. Additionally, the shortcomings of the ISO 23482-1 have been discussed and highlighted which may be taken into consideration when testing for standard compliance in robots as well as for improving ISO 23482-1 in new editions. Furthermore, the proposed thesis methodology was able to detect a driver bug in the Clearpath Jackal as well as in the MBS ROVO2 of an internal project in the MYBOTSHOP [14]

organization of which was reproducible in the real robot.

8.1 Contributions

Summarizing, the contributions of this thesis are.

Informed Automation for standards

This thesis enables parameterizable automated testing for standards such as ISO 23482-1, improving the quality of the robotic systems before they are deployed as well as not only saving resources but also time and cost. It accelerates testing, provides hypothetical coverage, and enables diversified testing with which hidden faults in the system can be revealed as well as allowing for a bug-free system.

Feature and Functionality Check

The testing of robots using the proposed thesis enables researchers, integrators, and manufacturers to identify shortcomings in their robots in the context of both hardware and software e.g. when testing a scenario in which a robot equipped with a LiDAR does not stop or reduce its speed when nearing a ditch, it can be identified that it is missing a software feature of terrain response, another example being when the robot topples over while performing an emergency halt in slippery terrain, then it can be deduced that the robot may require a hull redesign with appropriate mass distribution to ensure stability, and so forth.

Applicable to different types of robots

The proposed framework has the capability of testing a variety of mobile robots such as wheeled robots (Clearpath Husky, Clearpath Jackal), tracked robots (MBS ROVO2), quadruped robots (Quadruped B1) as well as robotic arms (Ufactory xARM6). Moreover, several hundred test cases can not only be implemented with minimal effort but also rapidly tested for these different robots with customizable scenarios and acceptance criteria provided by the user via the property-based testing framework (e.g. the standard ISO 23482-1).

8.2 Limitations

Several limitations that hinder the proposed thesis when considering standard compliance testing in robots are as follows:

Standard Deficiency

The standards such as ISO 23482-1 are not mature enough to be used as a baseline to ensure **complete** safety and security for robots. This issue is not limited to ISO 23482-1 but extends to other standards such as ISO 15066 and ISO 13482 as pointed out by Herrmann [34] and Valori [4] respectively. The standards need several revisions in which their requirement conflicts with other standards are resolved, as well as rectification in their own structure and clarity (particularly for ISO 23482-1). These issues in the standard affect the safety of robots when evaluated only for standard compliance via the revised

property-based testing framework, however, if the user integrates their own test and takes advantage of the randomized testing, more safety and reliability can be achieved for robotic actions and behaviors as compared with sole use of tests from the standards.

Missing Robot Functionalities

All the robots that have been tested in this thesis are not fully standard compliant as the robots are missing several functionalities of hardware and software. An example of missing hardware functionality is the missing carriage to ensure that the payload does not slip during terrain traversal. As for a software example, the unavailability of hazardous terrain response planners when the robot is faced with a convex or a concave terrain. The issue that this presents is the lack of a real robot that is fully standard-compliant that can be used as a benchmark for testing standard compliance in other robots.

The missing functionalities can be integrated into the robots by the manufacturers, however, it increases the cost of the already much expensive robot. As a consequence, the manufacturers do not include them in the robots' base package (both hardware and software). An idealized robot that is completely standard compliant can be developed in simulation, however, that is out of the scope of this thesis.

Simulation Gap

One of the most prominent limitations of the proposed thesis is the sim-to-real gap. The limitation is that the tests performed using the simulated workbench do not translate over to the real world accurately, especially when testing hardware functionalities such as halting on low friction surfaces. This issue is expedited even more for the standard tests that are un-simulatable e.g. testing acoustic interference on the robot. However, this limitation is becoming less of an issue with the continuous development of powerful simulators such as the Nvidia Issac Sim [13] that are able to mimic real-world environments very accurately.

Robot Configuration

The efficacy of this thesis's framework is limited to how well the user can accurately depict their robot in simulation. Incorrect or inaccurate configuration of the robot or the simulator will potentially provide wrong or misleading results when performing standard compliance tests. The configuration includes accurate information on the robot's Universal Robot Description Format (URDF), an accurate configuration of its simulated hardware drivers to its real-world drivers, and finally correct simulation parameters.

8.3 Future work

Comprehensive Domain Specific Language

The Robot Test Definition Language (RTDL) which is a domain-specific language has presently a limited set of grammar rules for scenario generation and testing. It would be ideal to expand the RTDL to include grammar rules for asset definition and generation, add the ability to select between different algorithms for the same test type (e.g. test collision with Axis-Aligned Bounding Box method or with a force contact

sensor), add moving actors for dynamic obstacle avoidance, as well as have a large variety of preset test cases (more than what is currently available) to test the often failing hardware and software functionalities.

New Simulation Platform

To reduce the sim-to-real gap, a port from the current Gazebo simulator to a more powerful and efficient simulator such as the Issac Sim [13] is required. This port will enable faster testing than Gazebo (due to GPU-based physics processing) as well as provide realistic dynamic actors, and more accurate physics simulation. However, the current version of Nvidia Issac Sim is not documented well but has been improving incrementally and it is expected that by the mid of 2023, sufficient documentation would be available to smoothly port the current revised property-based testing framework into the Issac Sim.

Assets Generation

In the current version of this work, the environment and obstacles are placed in the simulation using pre-modeled mesh files which are designed in CAD software: Fusion 360 [48]. However, as a future work, the integration of asset generation via the DSL would enable the user to directly provide the parameters for generating desired environments and obstacles bypassing the need for simple 3D models e.g. the transparent cube mentioned in ISO 23482-1.

Standard Rectification

A document containing a proposed rectification and solution to the highlighted issues in the ISO 23482-1 can be formulated and submitted to the ISO/TC 299 robotics technical committee to ensure clear and intelligible testing in future editions.

Fully Standard Compliant Robot

An idealized simulated robot can be developed as a benchmark for comparison with other robots. It should have be able to pass all the simple test cases such as stability and autonomous navigation as well as include all the basic safety hardware and software functionalities.

A

Foundation

A.1 Definitions

- **Machinery Directive: Robots** [4] European Machinery Directive 2006/24/EC [51] defines a robot as a multi-linked component system being actuated by a drive system. This type of robot is considered to be partly completed machinery and is deemed as complete machinery only when it is fully integrated into its designated application.
- **Technical Committee (TC)** [4] Technical committees are in command of the development of soft standards. These standards do not require mandatory compliance. An example is the ISO TC299 which defines terms pertaining to Robotics.
- **Technical Specifications (TS)** [4] Technical specifications inform on the current state-of-the-art work that may be included in the international standard.
- **Technical Reports (TR)** [4] Technical reports are informative reports on what is perceived to be state-of-the-art in a specific topic.
- **Property-based Testing (PBT)** [2] Property-based testing is a simulated randomized testing framework for verifying the actions and behaviors of a robot.
- **Robot Test Definition Language (RTDL)** Robot Test Definition Language is a domain-specific language used to create and test robot actions and behaviors in a variety of scenarios.
- **Verification** Verification is defined in this thesis as the successful execution of designated tests or requirements.
- **Validation** Validation in this thesis is defined as a measure of determining whether a designed approach or verification satisfactorily meets an expected solution.

B

Project Setup

B.1 Robots

All the technical information of the evaluated robots can be found at MYBOTSHOP [14].

B.1.1 Clearpath Husky

The Clearpath Husky [52] is a Canadian-manufactured mid-sized wheeled robot that is used for research and development in academia and industries. Commonly, Huskies are fitted with different sensors such as LiDARs, GPS, Depth-Cameras, etc. Additionally, due to their size, robotic arms can be mounted on them enabling mobile manipulation tasks. An example of the mobile manipulation platform for the Husky can be viewed in its documentation [53].

The software used to operate the Husky is done via ROS. The ROS packages for the Husky are currently open source and are available up to the ROS Noetic distribution. Pre-requisites for running the Husky packages are Ubuntu 20.04 and the base ROS Noetic distro. All the packages for the Husky can readily be installed via its binary packages:

Listing B.1: Husky installation

```
sudo apt-get install ros-noetic-husky*
```

The navigation for the Husky is accomplished via a pre-configured ROS package (i.e. move_base)

B.1.2 Clearpath Jackal

The Clearpath Jackal [54] is the smaller variant of the Clearpath Husky. It is an economical alternative for the Husky and is mostly used for reconnaissance missions. A notable difference between the Husky and the Jackal other than its size is its speed. The maximum speed of the Husky is limited to $1m/s$ whereas the maximum speed of the Jackal is $2m/s$.

In terms of software, the Jackal is similar to the Husky and is controlled via ROS. The ROS packages for the Jackal are currently open source and are available up to the ROS Noetic distribution. Pre-requisites for running the Jackal packages are Ubuntu 20.04 and the base ROS Noetic distro. All the packages for the Jackal can be installed via its binary packages:

Listing B.2: Husky installation

```
sudo apt-get install ros-noetic-jackal*
```

Akin to the Husky, the Jackal navigates via move_base.

B.1.3 MBS ROVO2

The MBS ROVO2 [55] is the autonomous version of the ROVO2 which is an industrial tracked robot. The MBS ROVO2 is a large robot that can reach speeds of $5.5m/s$ and tow up to $500Kgs$ of payload. Currently, the simulated version of the MBS ROVO2 utilizes small virtual wheels to replicate its tracked wheel motion in the simulator due to simulator limitations. The MBS ROVO2 is normally fitted with auxiliary sensors such as LiDARs as well as dept-cameras for autonomous navigation.

The software for the MBS ROVO2 is closed-source, hence, information on it cannot be provided.

B.1.4 Ufactory xARM6

The Ufactory xARM6 [56] is a robotic arm manufactured in China. It is able to perform all types of manipulation tasks with high accuracy. The current simulated version of the xARM6 requires several extra plugins to work correctly in the simulator, however, this limitation can be attributed directly to the simulator itself. Several variants of the xARMs are available such as the $5dof$ arm, $6dof$ arm, and $7dof$. These robotic arms are comparable to the brand robotic arms such as Franka Emika's Panda, Universal Robots UR5, Kinova arm, etc. Nevertheless, the notable negatives for the xARM6 are that it does not have an in-built force-torque sensor and has less workspace reachability as compared to the other robotic arm. As for the positives, the xARM6 is highly cost-effective, the software development kits are readily available and are much easier to use as compared to the other more expensive brands.

The integration and use of the simulated version of the xARM is not a straightforward process and requires several other simulator plugins to function correctly. The xARM6 gazebo simulator package is open-source, however, it has to be built from source. The software can be found in MYBOTSHOPs inventory [56]. The xARM6 can be controlled by different software (Blocky/Python/ROS/C++) including ROS which was done in this Thesis. To control the xARM6 the move_it package was employed.

B.1.5 Quadruped B1

The Quadruped B1 [14] is a four-legged robot and is manufactured in China. It is a cost-effective quadruped robot as compared to other competitor robots such as Boston Dynamics Spot or ANYbotics ANYmal. The main difference between these quadrupeds from different manufacturers is the maturity of their software drivers and functional features. Quadrupeds such as the B1 are especially hard to use in simulators due to their motion being dependent on foot contact actuation instead of the commonly used wheel actuation.

For software, the B1 drivers utilized are from CHAMP [49]. The B1 drivers have not been used as the manufacturers are still developing their version of the simulated hardware drivers. Moreover, the B1

assets are available in the MYBOTSHOP [14] inventory, however, a complete version of it working with the CHAMP simulator is currently, unavailable.

B.2 Software requirements

The following items are required for our project setup.

B.2.1 Operating system (OS)

- Ubuntu 20.04

B.2.2 Applications

We will be using the following applications for our software package:

- Simulation Environment — Gazebo & ~~Nvidia Issac Sim~~
- Middle-ware — Robot Operating System (ROS Noetic)

B.2.3 Libraries

Essential libraries that are utilized and are required for running the package.

- numpy
- pandas
- Hypothesis library
- pytest
- Allure
- torch
- TextX

B.3 Hardware Requirements

These constitute the hardware that was used for the developed software package.

- 16 Gb RAM
- AMD Ryzen 5 5600H CPU @ 3.30GHz
- Nvidia GeForce RTX 3060
- 250 Gb SSD

B.4 ROS Packages

B.4.1 Robots

- Husky
- Jackal
- xARM6
- ROVO2
- B1

B.4.2 Auxiliaries

- Ouster

B.5 Husky Technical Specification

CLEARPATH ROBOTICS

3 SYSTEM SPECIFICATIONS

Key specifications of Husky are shown in Table 2 .

Dimensions	990 mm length 670 mm width 390 mm height	39 in length 26.4 in width 14.6 in height
Track	555 mm	21.9 in
Wheelbase	512 mm	20.2 in
Weight	50 kg	110 lbs
Maximum Payload[1]	75 kg	165 lbs
All-terrain payload[2]	20 kg	44 lbs
Speed (max)	1.0 m/s	3.3 ft/s
Ground clearance	130mm	5 in
Climb grade	45°	100% slope
Traversal grade	30°	58% slope
Operating Ambient Temperature	-10 to 30° C	14 to 86° F
Operating time	3 hours typical 8 hours standby (no motion)	
Battery	24V 20Ah Sealed Lead Acid	
Battery charger	Short-circuit, over-current, over-voltage and reverse voltage protection	
Charge time	10 hours	
User Power	5V / 12V / 24V Each fused at 5A	
Communication	RS-232 115200 Baud	
Wheel Encoders	78,000 ticks/m	
Internal Sensing	Battery Status Wheel odometry Motor currents	

Table 2: Husky System Specifications

[1] Continuous operation on relatively flat terrain with wide turns
[2] Vehicle climbing 30° grade with high-mounted payload, or turning in place in high-friction conditions

Rev. 1.1.3 8 Husky UGV

Figure B.1: Figure is from Clearpath Husky Software Manual [16]. Manufacturer specification for Husky robot.

C

Standards Review

The following chapter comprises of standards identified to be relevant when considering safety in autonomous systems. Identification and the selection of the standards have been done based on the relevance of the standards to the field of autonomous systems and robotics as well as recommendations suggested by researchers and industrialists. A brief description is provided for all the selected standards. Many of the standards that have been surveyed are from the International Standard Organization (ISO).

#	**ID**	**Standard**	**Description**
1.	ISO 51:2014	Safety aspects - Guidelines for their inclusion in standards	This standard provides a generalized template on safety aspects and risk reduction techniques for users such as designers, integrators, manufacturers, distributors, regulators, etc. In terms of robotics, this standard is primarily considered during the designing and conceptualization phase. [57]
2.	ISO 8373:2021	Robotics — Vocabulary	ISO 8373 is a descriptive standard that provides definitions of robotics as well as terms relating to robotics. [58]
3.	ISO 9283:1998	Manipulating industrial robots - Performance criteria and related test methods	ISO 9283 deals exclusively with robotic industrial manipulators. It provides a convention on the coordinate system, terminology, mechanical interface, safety, characteristics, tests, and test report formats for robotic manipulators bridging the standard between users and manufacturers. [59]

4.	ISO 9409-1:2004	Manipulating industrial robots - Mechanical interfaces - Part 1: Plates	ISO 9409-1 provides design parameters for the mechanical coupling of the robotic manipulator. The standard is typically considered when designing robotic manipulators and is used for the interchangeability of different end-effectors. [60]
5.	ISO 9409-2:2002	Manipulating industrial robots - Mechanical interfaces - Part 2: Shafts	ISO 9409-2 is an extension of ISO 9409-1 and provides definitions and parameters for the robotic arm's cylindrical shafts. The objective is to maintain a standard convention and coordinate system for the interoperability of robotic arms. [61]
6.	ISO 9787:2013	Robots and robotic devices - Coordinate systems and motion nomenclatures	ISO 9787 provides a generalized and universal convention for robotic locomotion defined in ISO 8373 in terms of coordinate systems, terminologies, testing, and expected programming standards. [62]
7.	ISO 9946:1999	Manipulating industrial robots - Presentation of characteristics	ISO 9946 provides guidelines on the presentation of robots by industries. It is an informative standard that aids users in the comparison of various robots. [63]
8.	ISO 10218-1:2011	Robots and robotic devices-safety requirements for industrial robots-part 1: robots	ISO 10218-1 provides a general guide for industrial robots (although applicable in other robots). The guide describes requirements that need to be met for a safe robot as well as safe design features, hazards, risks, precautionary measures, etc. excluding only noise pollution of the robot. [8]
9.	ISO 10218-2:2011	Robots and robotic devices-safety requirements for industrial robots-part 2: robot systems and integration	ISO 10218-2 extends ISO 10218-1 and provides information on the deployment phase of industrial robots, specifically the integration of robotic cells into the industries. Integration includes several aspects such as designing a robotic cell, manufacturing, deployment, operation, preservation, retirement, and safety-hazard analysis. [17]

10.	ISO 11593:2022	Robots for industrial environments - Automatic end effector exchange systems - Vocabulary	ISO 11593 serves as a dictionary for automatic end-effector systems. Along with the provision of vocabulary and definitions, it provides information on the convention to be used for symbols and units as well. [64]
11.	ISO 12100:2010	Safety of machinery-general principles for design-risk assessment and risk reduction	ISO 12100 provides definitions and methodologies for assessing risks and hazards in general machinery as well as presenting safety requirements for machine designs. It is used as a foundational standard upon which other standards build. In the context of robotics, it is used mostly in the design phase of building a robot. [65]
12.	ISO/TR 13309:1995	Manipulating industrial robots - Informative guide on test equipment and metrology methods of operation for robot performance evaluation in accordance with ISO 9283	ISO 13309 is an informative standard that provided information on the operation of the latest technology at the time as well as their application. This standard particularly describes the tech. used in ISO 9283. [66]
13.	ISO 13482:2014	Robots and robotic devices-safety requirements for personal care robots	ISO 13482 provides information on risk analysis, safety designs, and precautionary measures on three classes of personal robots (related primarily to human care). A personal care mobile robot, a personal care physical aid robot, and a person mobility robot. [36]
14.	ISO 13849-1:2015	Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design	ISO 13849-1 provides instructions on design safety and performance requirements for control systems of machinery including the software control system (e.g. relays-solenoids, motor controls, valves, PLCs, switches, and other equipment). As all robots have different types of motors as well as sensing equipment, this standard dictates the minimum viable safety and performance requirement for low-level hardware components of a robot. [67]
15.	ISO 13849-2:2012	Safety of machinery - Safety-related parts of control systems - Part 2: Validation	ISO 13849-2 is an extension of ISO 13849-1 and specifies an evaluatory process for the safety and performance requirements for control systems as stated in ISO 13849-1. [68]

16.	ISO 13855:2010	Safety of machinery-positioning of safeguards with respect to the approach speeds of parts of the human body	ISO 13855 lists information on the methodology for determining the minimum speed, distance, and hazard zone construction for machinery from humans. A note on this standard is that it is well-tested and has proven to save many limbs and lives. Parts of this standard are foundational for several robotic standards which also require the restriction of a robot's movement on human detection. [69]
17.	ISO 13851:2019	Safety of machinery-two-hand control devices-principles for design and selection	ISO 13851 specifies the information on the design of a two-hand control device (THCD). THCD is used to ensure that the hands of the users are positioned in a safe location while a hazardous machine is being operated. This standard is primarily used only for design consideration and implementation in industrial robots. [70]
18.	ISO 14539:2000 [71]	Manipulating industrial robots - Object handling with grasp-type grippers - Vocabulary and presentation of characteristics	ISO 14539 is part of a set of standards for industrial robotic manipulator arms. This particular standard covers definitions as well as characteristics that constitute an industrial robotic arm and its end-effectors (it also includes object handling by the manipulator). [71]
19.	ISO 14971:2019	Medical devices - Application of risk management to medical devices	ISO 14971 is a medical device standard that provides information on the terminology as well as the risk analysis methodologies for medical machinery. In robotics, this standard is used in design consideration, safety analysis, and risk analysis for medical robots and physical aid robots. [72]
20.	ISO/TS 15066:2016	Robots and robotic devices-collaborative robot	ISO/TS 15066 is a robotic standard for those robots that operate with or close to humans in an industrial environment. It extends ISO 10218-1 and ISO 10218-2 and provides guidelines for safety requirements for industrial collaborative robots. [5]

21.	ISO 18497:2018	Agricultural machinery and tractors-safety of highly automated machinery	ISO 18497 specifies information on safety-requirement and risk analysis for autonomous agricultural vehicles and machinery. [73]
22.	ISO 18646-1:2016	Robotics - Performance criteria and related test methods for service robots - Part 1: Locomotion for wheeled robots	ISO 18646-1 provides information on how to evaluate the mobility performance of an autonomous wheeled robot in indoor spaces. [74]
23.	ISO 18646-2:2019	Robotics - Performance criteria and related test methods for service robots - Part 2: Navigation	ISO 18646-2 extends ISO 18646-1 and evaluates the navigation of autonomous land vehicles. The evaluatory process mainly takes place by verification of the robot's final goal pose as well as its repeatability. [75]
24.	ISO 18646-3:2021	Robotics - Performance criteria and related test methods for service robots - Part 3: Manipulation	ISO 18646-3 specifies the convention and evaluation of manipulators for service robots in indoor environments which can be applied to outdoor environments as well. [76]
25.	ISO 18646-4:2021	Robotics - Performance criteria and related test methods for service robots - Part 4: Lower-back support robots	ISO 18646-4 provides information on the objective and application of lower-back support robots. It does not include medical robots nor does it provide information on validation processes for safety analysis. [77]
26.	ISO 19649:2017	Mobile robots - Vocabulary	ISO 19649 is an informative standard that lists out the definitions and terminologies for mobile robots (e.g. navigation and locomotion) both in the industrial sector as well as the service sector. [78]
27.	ISO/TR 20218-1:2018	Robotics-safety design for industrial robot systems-part 1: end-effectors	ISO/TR 20218-1 specifies the information for safety analysis for end-effectors that are to be installed and integrated into industrial robotic systems. [79]
28.	ISO/TR 20218-2:2017	Robotics-safety requirements for industrial robots-part 2: manual load/unload stations	ISO/TR 20218-2 provides information on robotic systems that load and unload cargo in a station that is commonly designated as a hazard zone and has restricted access. Additional information for robotic systems with loading and unloading stations is available in ISO 10218. [80]

29.	ISO/CD 22166-201 [Under Development]	Robotics — Modularity for service robots - Part 201: Common information model for modules	ISO/CD 22166-201 is an upcoming unpublished robotic standard that specifies information on the modularization of service robots in terms of software modules. Specifically, it provides guidelines on software safety, security, reusability, compatibility, interfaces, and specific functions for a robot to be able to execute certain use-case scenarios. [81]
30.	ISO/TR 23482-1:2020	Application of ISO 13482-part 1: safety-related test methods	ISO/TR 23482-1 is a general evaluatory standard for personal care robots defined in ISO 13482 (applicable to robots other than personal robots as well). It specifies information on verification and validation methods for different types of robots such as mobile robots, wearable robots, manipulators, etc. [1]
31.	ISO/TR 23482-2:2019	Robotics - Application of ISO 13482 - Part 2: Application guidelines	ISO/TR 23482-2 informs on designing of personal care robots given in ISO 13482 as well as performing risk-analysis on the use of these robots. Moreover, it clarifies the new clauses and requirements in ISO 13482. [82]
32.	ISO 3691-4:2020	Industrial trucks-safety requirements and verification-part 4: Driverless industrial trucks and their systems	ISO 3691-4 is a standard that provides information on safety and risks analysis for autonomous trucks (including autonomous mobile robots, cart systems, tuggers, etc.; anything that constitutes transportation). [6]
33.	ISO 31000:2018	Risk management - Guidelines	ISO 31000 is a meta standard that provides generalized guidelines on how to mitigate risks faced by any type of organization. It is highly referenced by robotic safety standards as it is a vital building block. Moreover, the principles provided by the standard can directly be used when developing safe and reliable robots. [83]

34.	ISO 31101 [Under Development]	Robotics - Application services provided by service robots - Safety management systems requirements	ISO 31101 is an upcoming standard that provides information for robotic application service providers (i.e. those who have bought robots from a manufacturer and then operate and provide the robots as a service) on safety-management systems as well as a methodology to deal with residual risks. The difference between this standard from ISO 13842 is that it is exclusively for application service providers and that its safety management system is based on the plan-do-check-act (PDCA) cycle. [84]
35.	2006/42/EC	Machinery Directive	2006/42/EC is a European directive for machinery that specifies regulations for the interoperability and safety of machines with the EEA. Robots and their individual components have to comply with this standard to operate within the EEA region. [51]
36.	2009/104/EC	Use of Work Equipment Directive	2009/104/EC provides information on the requirements for safety in work equipment. This standard also holds true for physical aid robots as well as wearable robots. [85]
37.	89/654/EC	Workplace Directive	89/654/EC specifies the hygienic and safety requirements for workplaces. In context of robotics, a safety and hazard analysis has to be performed for compliance with this directive. [86]
38.	2001/95/EC	Product Safety Directive	2001/95/EC informs on the requirements for general products that are provided to the market for normal consumers. Robots that are to be distributed as a product in the EEA region have to comply with this directive. [87]
39.	2006/95/EC	Low Voltage Directive (LVD)	2006/95/EC provides information on safety for low voltage (AC:$50 - 1000V$, DC:$75 - 1500V$) electrical equipment and electronics. Similar to 2006/42/EC robotic components must be assessed for their conformity with the directive. [88]

40.	2004/108/EC	Electromagnetic Compatibility Directive (EMC)	2004/108/EC specifies compatibility regulations for electronic devices in the context of electromagnetic interference (EMI) that are to be operated in the EU region. This regulation is essential for robots manufacturers as most robots that operate in outdoor environments are affected by EMI which causes disruption in wireless communication, GPS, and IMU. [89]

Table C.1: International standards and European standards (Information for European standards [3]) used for design and safety aspects for autonomous robots.

D

ISO 23482-1 Test Definitions

D.1 ISO/TR 23482-1: §7.2

- **Objective:** Force imparted by a robot on a safety obstacle.
- **Target Platform:**
  - Mobile robot
  - Robotic Arms
- **Related Standards:**
  - ISO/TS 15066
  - ISO 13482:2014 §4.3, §5.10.9.1
- **Required Components:**
  - Force sensor obstacle
  - Force measuring device
- **Scenario Description:**
  - Mobile robot or robotic arm moving in maximum speed towards force measuring device
- **Acceptance Criteria:**
  - Detecting Force imparted by the robot is less than the values provided in ISO/TS 15066 (currently unavailable)
- **Quantity of tests:**
  - $\times 1$
- **Variation in scenarios:**
  - None

- **Approach:**
  1. Safety obstacle asset creation
  2. Sensor addition into the robots URDF
  3. Scenario implementation
     - Design and implementation of composite property: *MustHaveCollisionForceLessThan*

D.2 ISO/TR 23482-1: §11

- **Objective:** Mobile robot's static stability with and without load.
- **Target Platform:**
  - Mobile robot
- **Related Standards:**
  - ISO 13842:2014 §5.10.2.1
  - ISO 7176-1:2014 §3.2
- **Required Components:**
  - Robot
  - Load (Maximum payload for the robot)
  - Slope [6°, 9°, 15°]
- **Scenario Description:**
  - Mobile robot facing forward on a 6° slope (ascent)
  - Mobile robot facing backward on a 6° slope (ascent)
  - Mobile robot facing right on a 6° slope (ascent)
  - Mobile robot facing left on a 6° slope (ascent)
  - Mobile robot facing forward on a 9° slope (ascent)
  - Mobile robot facing backward on a 9° slope (ascent)
  - Mobile robot facing right on a 9° slope (ascent)
  - Mobile robot facing left on a 9° slope (ascent)
  - Mobile robot facing forward on a 15° slope (ascent)
  - Mobile robot facing backward on a 15° slope (ascent)
  - Mobile robot facing right on a 15° slope (ascent)
  - Mobile robot facing left on a 15° slope (ascent)

  - Mobile robot facing forward on a 6° slope (descent)
  - Mobile robot facing backward on a 6° slope (descent)
  - Mobile robot facing right on a 6° slope (descent)
  - Mobile robot facing left on a 6° slope (descent)
  - Mobile robot facing forward on a 9° slope (descent)
  - Mobile robot facing backward on a 9° slope (descent)
  - Mobile robot facing right on a 9° slope (descent)
  - Mobile robot facing left on a 9° slope (descent)
  - Mobile robot facing forward on a 15° slope (descent)
  - Mobile robot facing backward on a 15° slope (descent)
  - Mobile robot facing right on a 15° slope (descent)
  - Mobile robot facing left on a 15° slope (descent)

- **Acceptance Criteria:**
  - Detecting Mobile robot not falling over
  - Detecting Mobile robot's payload not falling

- **Quantity of tests:**
  - ×48

- **Variation in scenarios:**
  - Yes

- **Approach:**
  1. ×3 slope asset creation
  2. Scenario implementation
     - Reuse of composite property: *MustHaveOrientation* as well as design and implementation of composite property *MustBeAt*.

D.3 ISO/TR 23482-1: §12

- **Objective:** Robots' dynamic stability when forces are imparted on it with and without load.

- **Target Platform:**
  - Mobile robot
  - Robotic Arms

- **Related Standards:**
  - ISO 13842:2014 §5.10.2.1
- **Required Components:**
  - Robot
  - Load (Maximum payload for the robot)
  - Slope [6°, 9°, 15°]
  - Force generator
- **Scenario Description:**
  - Mobile robot moving forward on a 6° slope (ascent)
  - Mobile robot moving backward on a 6° slope (ascent)
  - Mobile robot turning right on a 6° slope (ascent)
  - Mobile robot turning left on a 6° slope (ascent)
  - Mobile robot moving forward on a 9° slope (ascent)
  - Mobile robot moving backward on a 9° slope (ascent)
  - Mobile robot turning right on a 9° slope (ascent)
  - Mobile robot turning left on a 9° slope (ascent)
  - Mobile robot moving forward on a 15° slope (ascent)
  - Mobile robot moving backward on a 15° slope (ascent)
  - Mobile robot turning right on a 15° slope (ascent)
  - Mobile robot turning left on a 15° slope (ascent)
  - Mobile robot moving forward on a 6° slope (descent)
  - Mobile robot moving backward on a 6° slope (descent)
  - Mobile robot turning right on a 6° slope (descent)
  - Mobile robot turning left on a 6° slope (descent)
  - Mobile robot moving forward on a 9° slope (descent)
  - Mobile robot moving backward on a 9° slope (descent)
  - Mobile robot turning right on a 9° slope (descent)
  - Mobile robot turning left on a 9° slope (descent)
  - Mobile robot moving forward on a 15° slope (descent)
  - Mobile robot moving backward on a 15° slope (descent)
  - Mobile robot turning right on a 15° slope (descent)

  - Mobile robot turning left on a 15° slope (descent)
  * Robotic arm fully extended vertically with a maximum load
  * Robotic arm fully extended horizontally with a maximum load
  * Robotic arm fully extended diagonally with a maximum load

- **Acceptance Criteria:**
  - Detecting mobile robot not falling over
  - Detecting mobile robot's payload not falling
  - Robotic manipulators payload not falling

- **Quantity of tests:**
  - ×92

- **Variation in scenarios:**
  - Yes

- **Approach:**
  1. Addition of simulator plugin to impart random forces on the mobile robot as well as the robotic arm to directly cause it to fall or its payload
  2. Scenario implementation
     - Reuse of composite property: *MustHaveOrientation* as well as composite property *MustBeAt*.

D.4 ISO/TR 23482-1: §13.1

- **Objective:** Ambiguous

- **Target Platform:**
  - Mobile robot

- **Related Standards:**
  - ISO 13842:2014 §5.10.3.1

- **Required Components:**
  - Robot
  - Load (Maximum payload for the robot)
  - Flat surface
  - Slippery flat surface

  - Inclined surface
  - Stepped surface
  - Gapped surface
  - 15 travel patterns
- **Scenario Description:**
  - None
- **Acceptance Criteria:**
  - None
- **Quantity of tests:**
  - –
- **Variation in scenarios:**
  - N/A
- **Approach:**
  1. Flat surface, slippery surface, inclined surface, stepped surface, and gapped surface asset creation
  2. Scenario implementation
     - None

D.5 ISO/TR 23482-1: §13.2

- **Objective:** Mobile robot's stability response in acceleration.
- **Target Platform:**
  - Mobile robot
- **Required Components:**
  - Load (Maximum payload for the robot)
  - Flat surface
- **Scenario Description:**
  - First scenario: Mobile robot braking at max. speed
  - Second scenario: Mobile robot during motion accelerating to max. speed
  - Third scenario: Mobile robot accelerating to max. speed from a stationary position

  - Mobile robot moving in $80\% \pm 10\%$
  - Load placed on the robot

- **Acceptance Criteria:**
  - Detecting Mobile robot not falling over
  - Detecting Mobile robot's payload not falling

- **Quantity of tests:**
  - $\times 6$

- **Variation in scenarios:**
  - Yes

- **Approach:**
  1. Scenario implementation
     - Reuse of composite property: *MustHaveOrientation* as well as composite property *MustBeAt*.

D.6 ISO/TR 23482-1: §13.3

- **Objective:** Mobile robot's stability response in inclined terrain with acceleration.

- **Target Platform:**
  - Mobile robot

- **Required Components:**
  - Load (Maximum payload for the robot)
  - Inclined slope (Max. angle allowed by Mobile robot)

- **Scenario Description:**
  - First scenario: Mobile robot braking at max. speed down the slope
  - Second scenario: Mobile robot braking at max. speed up the slope
  - Third scenario: Mobile robot accelerating to max. down the slope then turning 360°
  - Fourth scenario: Mobile robot accelerating to max. speed across the slope
  - Fifth scenario: Mobile robot turning ten times in an inclined slope $\times 10$
  - Distance covered when stop command issued and when robot actually stops
  - Deviating off of central path given in robots specification

  - Load placed on the robot

- **Acceptance Criteria:**
  - Detecting Mobile robot not falling over
  - Detecting Mobile robot's payload not falling

- **Quantity of tests:**
  - $\times 28$

- **Variation in scenarios:**
  - Yes

- **Approach:**
  1. Scenario implementation
     - Reuse of composite property: *MustHaveOrientation* as well as composite property *MustBeAt*.

D.7 ISO/TR 23482-1: §13.4

- **Objective:** Mobile robot's stability when traveling on steps and gaps.

- **Target Platform:**
  - Mobile robot

- **Related Standards:**
  - ISO 7176 §10

- **Required Components:**
  - Robot
  - Load (Maximum payload for the robot)
  - $20mm$ Steps (Filleted on the edges) (robot specification)
  - $50mm$ Steps (Filleted on the edges) (robot specification)
  - Gapped surface (robot specification)
  - 15 travel patterns

- **Scenario Description:**
  - Mobile robot moves upward on steps max. speed and then comes to a halt
  - Mobile robot moves upward on steps gradually and then comes to a halt

  - Mobile robot moves downward on steps max. speed and then comes to a halt
  - Mobile robot moves downward on steps slowly and then comes to a halt
  - Mobile robot moves across a gap based on allowable specifications of the robot at max. speed
  - Mobile robot moves across a gap based on allowable specifications of the robot at min. speed
- **Acceptance Criteria:**
  - Detecting mobile robot not falling over
  - Detecting mobile robot's payload not falling
- **Quantity of tests:**
  - $\times 12$
- **Variation in scenarios:**
  - Yes
- **Approach:**
  1. Scenario implementation
     - Reuse of composite property: *MustHaveOrientation* as well as composite property *MustBeAt*.

D.8 ISO/TR 23482-1: §15.1

- **Objective:** Mobile robot's halting capability when faced with an obstacle.
- **Target Platform:**
  - Mobile robot
- **Related Standards:**
  - ISO 13482:2014 §5.10.8.1
- **Required Components:**
  - Mobile robot
  - Obstacle detection
  - Obstacle: moving
  - Obstacle: wall
    * Wooden board $90cm \times 90cm$
    * Cloth $90cm \times 90cm$
    * Mirror $90cm \times 90cm$

    - Transparent board $90cm \times 90cm$
    - Fence $90cm - Length, 10mm - Diameter, 100mm - Apart, 90cm - Width$
  - Obstacle: human (Clothing black) or cylindrical piece can be used i.e. ISO 13856-3
  - Obstacle: pole ($90cm - Length, 25mm - Diameter$)

- **Scenario Description:**
  - Mobile robot moves straight [1]
  - Mobile robot moves straight and obstacle aligned left to travel direction
  - Mobile robot moves straight and obstacle aligned right to travel direction
  - Mobile robot moves left
  - Mobile robot moves right
  - Mobile robot moves backward
  - Obstacle perpendicular to Mobile robot (90°)
  - Obstacle angled to Mobile robot (45°)
  - Obstacle angled to Mobile robot (180°)
  - Obstacle angled to Mobile robot (270°)

- **Acceptance Criteria:**
  - Detecting Mobile robot's stopping distance

- **Quantity of tests:**
  - ×128

- **Variation in scenarios:**
  - Yes

- **Approach:**
  1. Wall, dynamic object, board, cloth (lack of support for deformable objects in Gazebo), mirror block, transparent block, fence, pole asset creation
  2. Scenario implementation
     - Design and implementation of composite property: *MustBeNearTo*.

[1] Obstacle beyond sensing range if required.

D.9 ISO/TR 23482-1: §15.2

- **Objective:** Mobile robot's speed on detection of obstacles.
- **Target Platform:**
  - Mobile robot
- **Related Standards:**
  - ISO 13482:2014 §5.10.8.1
- **Required Components:**
  - Mobile robot
  - Obstacle detection
  - Obstacle: moving
  - Obstacle: wall
    - Wooden board $90cm \times 90cm$
    - Cloth $90cm \times 90cm$
    - Mirror $90cm \times 90cm$
    - Transparent board $90cm \times 90cm$
    - Fence $90cm - Length, 10mm - Diameter, 100mm - Apart, 90cm - Width$
  - Obstacle: human (Clothing black) or cylindrical piece can be used i.e. ISO 13856-3
  - Obstacle: pole $(90cm - Length, 25mm - Diameter)$
- **Scenario Description:**
  - Obstacle beyond sensing range
  - Mobile robot moves straight
  - Mobile robot moves left
  - Mobile robot moves right
  - Mobile robot moves backward
  - Obstacle perpendicular to Mobile robot (90°)
  - Obstacle angled to Mobile robot (45°)
  - Obstacle angled to Mobile robot (180°)
  - Obstacle angled to Mobile robot (270°)
- **Acceptance Criteria:**
  - Detecting Mobile robot's reduced speed

- **Quantity of tests:**
  - $\times 128$
- **Variation in scenarios:**
  - Yes
- **Approach:**
  1. Scenario implementation
     - Reuse of composite property: *MustBeNearTo*.

D.10 ISO/TR 23482-1: §15.3

- **Objective:** Mobile robot's halting capability on convex terrain.
- **Target Platform:**
  - Mobile robot
- **Related Standards:**
  - ISO 13482:2014 §5.10.3.3
  - ISO 13482:2014 §6.5.3
- **Required Components:**
  - Mobile robot
  - Obstacle detection
  - Convex terrain (Max. traversable height of robot)
  - Surface sensing capabilities
- **Scenario Description:**
  - Mobile robot moves straight towards convex terrain [2]
  - Mobile robot moves straight towards the edge of the convex terrain
  - Mobile robot moves towards convex terrain at (90°)
  - Mobile robot moves towards convex terrain at (45°)
  - Mobile robot moves towards convex terrain at (10°)
- **Acceptance Criteria:**
  - Detecting Mobile robot's stopping distance

[2] Manual issue of stop command.

  - Detecting Mobile robot's speed reduction

- **Quantity of tests:**
  - $\times 5$
- **Variation in scenarios:**
  - Yes
- **Approach:**
  1. Convex terrain asset creation
  2. Scenario implementation
     - Reuse of composite property: *MustBeNearTo*.

D.11 ISO/TR 23482-1: §15.4

- **Objective:** Mobile robot's halting capability on concave terrain.
- **Target Platform:**
  - Mobile robot
- **Related Standards:**
  - ISO 13482:2014 §5.10.3.3
  - ISO 13482:2014 §6.5.3
- **Required Components:**
  - Mobile robot
  - Obstacle detection
  - Concave terrain (Min. traversable height of robot)
  - Surface sensing capabilities
- **Scenario Description:**
  - Mobile robot moves straight towards concave terrain [3]
  - Mobile robot moves straight towards the edge of the concave terrain
  - Mobile robot moves towards concave terrain at (90°)
  - Mobile robot moves towards concave terrain at (45°)
  - Mobile robot moves towards concave terrain at (10°)

[3] Manual issue of stop command.

–

- **Acceptance Criteria:**
  - Detecting Mobile robot's stopping distance
  - Detecting Mobile robot's speed reduction
- **Quantity of tests:**
  - $\times 5$
- **Variation in scenarios:**
  - Yes
- **Approach:**
  1. Concave terrain asset creation
  2. Scenario implementation
     - Reuse of composite property: *MustBeNearTo*.

D.12 ISO/TR 23482-1: §16

- **Objective:** Mobile robot response in localization and navigation tasks.
- **Target Platform:**
  - Mobile robot
- **Related Standards:**
  - ISO 13482:2014 §5.16.2
- **Required Components:**
  - Safety room $6m \times 3m$ with a non-static rectangular obstacle ($1m \times 1m$) placed from the longer sides of the room
  - Sensors
  - Map of safety-room
- **Scenario Description:**
  - Mobile robot moves between 5 points, 4-times clockwise, 4-times anti-clockwise
  - Mobile robot navigation points are incrementally placed at initially $5cm$, $10cm$, $20cm$, and $50cm$
- **Acceptance Criteria:**

  - Detecting unsafe behavior
  - Detecting consequential maneuvers

- **Quantity of tests:**
  - ×64
- **Variation in scenarios:**
  - Yes
- **Approach:**
  1. Safety room asset creation
  2. Adding various sensors for autonomous localization and navigation
  3. Scenario implementation
     - Design and implementation of composite property: *MustNotCollide*.

D.13 ISO/TR 23482-1: §17

- **Objective:** Autonomous robots decision assessment.
- **Target Platform:**
  - Mobile robot
  - Robotic manipulator
- **Related Standards:**
  - ISO 13482:2014 §5.12
- **Required Components:**
  - Sensing Capability
  - Autonomous modules (e.g. obstacle avoidance. object grasping)
- **Scenario Description:**
  - Object identification
  - Response to hazardous commands
  - User-defined tests
- **Acceptance Criteria:**
  - Assessing actions performed are within user expectation
  - Detecting anomalous behavior

- **Quantity of tests:**
  - ∞
- **Variation in scenarios:**
  - Yes
- **Approach:**
  1. Property-based testing framework randomized composite property tests.

D.14 ISO/TR 23482-1 Test Parameters

The following parameters are to be assumed when applying the safety case defined by the standard.

- ~~Temperature: 10° — 30°~~ (Simulator does not support the parameter)
- ~~Humidity: 0% — 80%~~ (Simulator does not support the parameter)
- Surface Friction Coefficient: 0.75 — 1.00 (Applied to the simulation)

E

Robot Test Definition Language: Grammar Rules

Listing E.1: Grammar rules for the Robot Test Definition Language used for designing and implementing the three modes of testing i.e. User testing, Randomized testing, and Standard testing.

```
TestInterface:
  'Test_definitions␣(S)'
    test_type+=TestType
  'Test_definitions␣(E)'
;

TestType:
    StandardScenario | UserScenario | RandomizerScenario
;

StandardScenario:
    'standard' standard = STRING
    section_number += StandardSection
;

StandardSection:
    'section' section = FLOAT
    scenario_configuration = ScenarioConfiguration
    custom_scenario = CompositeProperties
;

UserScenario:
    'UserScenario'
    scenario_configuration = ScenarioConfiguration
```

```
    custom_scenario = CompositeProperties
;

RandomizerScenario:
    'RandomizerScenario'
    scenario_configuration = RandomScenarioConfiguration
    custom_scenario = CompositeProperties
;

RandomScenarioConfiguration:
    "scenario_configuration"
    "iterations" iterations = INT
    "worlds" worlds += RandomWorlds
    "obstacles" obstacles += RandomObstacles
;

RandomWorlds:
    "world_type" world_type = STRING
;

RandomObstacles:
    "obstacle_type" obstacle_type = STRING
;

ScenarioConfiguration:
    "scenario_configuration"
    "world_type" world_type = STRING
    'description' description = STRING
    scenario_modifier *= ScenarioModifiers
;

ScenarioModifiers:
    sm_robot_position *= RobotPosition
    sm_robot_goal *= RobotGoal
    sm_payload *= Payload
    sm_robot_velocity *= RobotVelocity
    sm_imparted_forces *= ImpartedForces
    sm_safety_obstacle *= SafetyObstacle
    sm_velocity_percentage *= VelocityPercentage
```

```
    sm_repetition *= Repetition
    sm_randomize_obstacles *= Randomization
    sm_dead_time *= DeadTime
;

RobotPosition:
    "robot_position"
    'x_pos' x_pos = FLOAT
    'y_pos' y_pos = FLOAT
    'z_pos' z_pos = FLOAT
    'r_ori' r_ori = FLOAT
    'p_ori' p_ori = FLOAT
    'y_ori' y_ori = FLOAT
;

RobotGoal:
    "robot_goal"
    'x_pos' x_pos = FLOAT
    'y_pos' y_pos = FLOAT
    'z_pos' z_pos = FLOAT
    'r_ori' r_ori = FLOAT
    'p_ori' p_ori = FLOAT
    'y_ori' y_ori = FLOAT
;

RobotVelocity:
    "manual_speed" manual_speed = BOOL
    "robot_speed" robot_speed = FLOAT
    "speed_duration" speed_duration = FLOAT
;

Payload:
    "payload" payload= STRING
    'x_pos' x_pos = FLOAT
    'y_pos' y_pos = FLOAT
    'z_pos' z_pos = FLOAT
    'r_ori' r_ori = FLOAT
    'p_ori' p_ori = FLOAT
    'y_ori' y_ori = FLOAT
```

```
;

ImpartedForces:
    "imparted_forces" imparted_forces = FLOAT
    "target_entity" target_entity = STRING
;

SafetyObstacle:
    "safety_obstacle" safety_obstacle = STRING
    'x_pos' x_pos = FLOAT
    'y_pos' y_pos = FLOAT
    'z_pos' z_pos = FLOAT
    'r_ori' r_ori = FLOAT
    'p_ori' p_ori = FLOAT
    'y_ori' y_ori = FLOAT
;
VelocityPercentage:
    "velocity_percentage" velocity_percentage = FLOAT
;
Repetition:
    "repetition" repetition = INT
;
Randomization:
    "randomize_obstacles" randomize_obstacles = BOOL
;

DeadTime:
    "dead_time" dead_time = FLOAT
;

CompositeProperties:
    property_check += PropertyCheck
;

PropertyCheck:
    'must_not_collide' must_not_collide = MustNotCollide |
    'must_collide' must_collide = MustCollide |
    'must_be_at' must_be_at = MustBeAt |
    'must_not_be_at' must_not_be_at = MustNotBeAt |
```

```
    'must_have_collision_force_less_than' must_have_collision_force_less_th
    'must_have_orientation' must_have_orientation = MustHaveOrientation |
    'must_be_near_to' must_be_near_to = MustBeNearTo |
    'must_not_be_near_to' must_not_be_near_to = MustNotBeNearTo
;

MustNotCollide:
    'collision_object' collision_object = STRING
;

MustCollide:
    'collision_object' collision_object = STRING
;

MustBeAt:
    'area_start'
    'x1' x1 = FLOAT
    'y1' y1 = FLOAT
    'z1' z1 = FLOAT
    'area_end'
    'x2' x2 = FLOAT
    'y2' y2 = FLOAT
    'z2' z2 = FLOAT
    'tolerance' (tolerance = FLOAT)?
    'time-constraint' (time_constraint = FLOAT)?
;

MustNotBeAt:
    'allowable_region_start'
    'x1' x1 = FLOAT
    'y1' y1 = FLOAT
    'z1' z1 = FLOAT
    'allowable_region_end'
    'x2' x2 = FLOAT
    'y2' y2 = FLOAT
    'z2' z2 = FLOAT
    'tolerance' (tolerance = FLOAT)?
    'time-constraint' (time_constraint = FLOAT)?
;
```

```
MustHaveCollisionForcesLessThan:
    'force_threshhold' force_threshhold = FLOAT
;

MustHaveOrientation:
    'entity' entity = STRING
    'angles'
    'roll' roll = FLOAT
    'pitch' pitch = FLOAT
    'yaw' yaw = FLOAT
    'tolerance' tolerance = FLOAT
;

MustBeNearTo:
    'entity' entity = STRING
    'euclidean_distance' euclidean_distance = FLOAT
    'tolerance' tolerance = FLOAT
;

MustNotBeNearTo:
    'entity' entity = STRING
    'euclidean_distance' euclidean_distance = FLOAT
    'tolerance' tolerance = FLOAT
;
```

F

Usecase: Husky Evaluation

All 245 test definitions defined in AppendixD have been evaluated for completeness for the Husky use case and are explained as follows.

ISO/TR 23482-1: §7.2

In this sub-use case, we evaluate the Husky's highest impact force on the force sensor obstacle. The force threshold corresponds to the human pain index. However, due to lack of availability, $75N$ has been chosen as a dummy value for the threshold of the Husky as the Husky maximum force at full speed that can be applied is around $75N$.

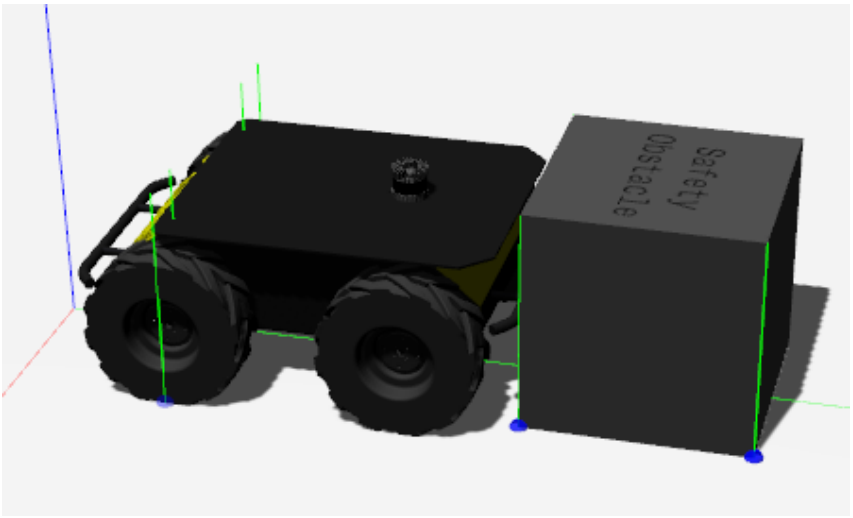


Figure F.1: §7.2 Husky impact force test

Observations

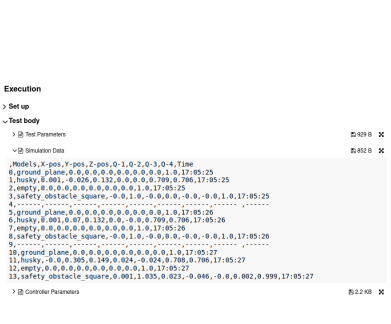
Several of the tests have failed due to the threshold being placed at nearly the exact value of the maximum collision force of the Husky. The reason why tests T3 and T4 have succeeded is due to the simulation

Table F.1: ISO 23482-1 Husky §7.2 Evaluation by the property-based testing framework. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Direction	Envir.	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§7.2	Forward	Plane	X	X	✓	✓	X	X	X	X	X	X

(a) Allure automatically generated report for testing of ISO 23482-1 §7.2 Husky

(b) Allure logged test parameters



(c) Allure logged simulation data

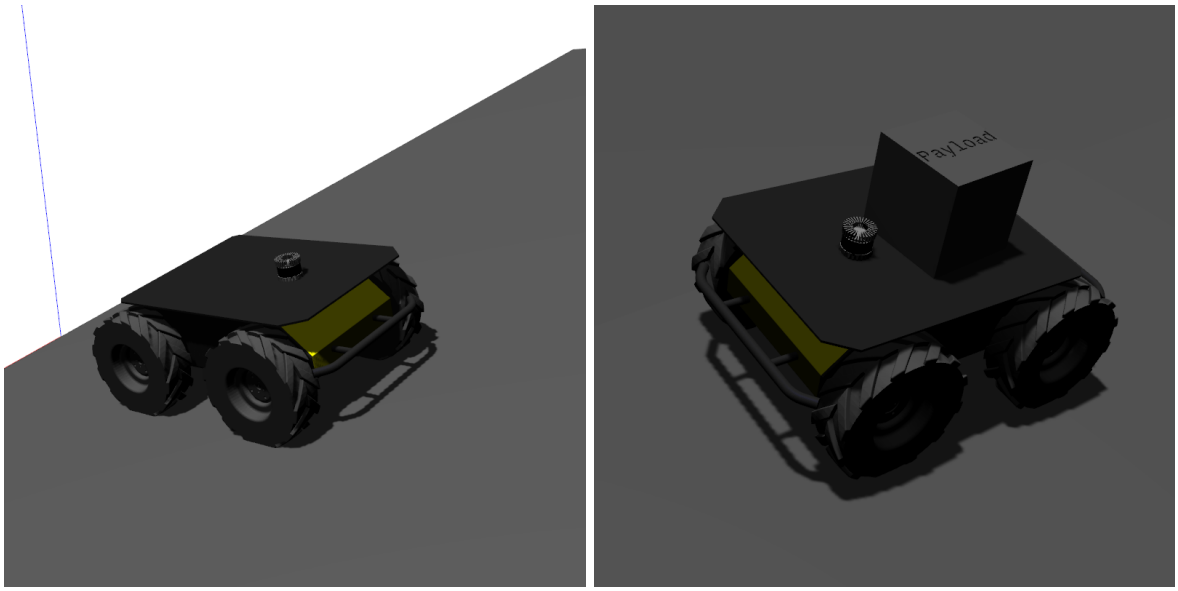
(d) Allure logged controller data

Figure F.2: The automated **Allure** result generation format is similar for each of the tests, however, with more definitions the test parameters increase substantially, due to which we shall not be providing it for other sections' tests.

time synchronization issue of the temporal logging unit. The time step is 1 second and the highest impact force at that time sequence is not recorded during the impact. The solution to this issue is to record at a smaller time step such as 0.5 seconds, but for performance, we opted to keep it at 1 second.

ISO/TR 23482-1: §11.1

In this sub-use case, we evaluate the Husky's dynamic stability by moving through 92 scenarios defined by ISO 23482-1. The scenarios consist of different slopes (6°,9°,15°). Although the tests in the standards have not been numerically provided, based on their information we categorized the test cases into 11.1 and 11.2. 11.1 use case considers the Husky without payload while 11.2 considers the Husky holding a $30kg$ square payload.



(a) §11.1 (b) §11.2

Figure F.3: Sub-Fig. (a) and (b) are Husky static stability tests on 15^o slope. In Sub-Fig. (b) the Husky has an additional $30kg$ payload.

Table F.2: ISO 23482-1 Husky §11.1 Evaluation by the property-based testing framework. Appendix D contains the test definitions. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Motion	Direction	Envir.	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§11.1	Ascent	Forward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Forward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 15^o	O	✓	✓	✓	✓	✓	✓	✓	O	✓
	Descent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Observations

Attributable to its bulky design, the Husky succeeded in 99.1% of the static stability tests. Two tests failed due to physics simulation errors.

ISO/TR 23482-1: §11.2

Table F.3: ISO 23482-1 Husky §11.2 Evaluation by the property-based testing framework. Appendix D contains the test definitions. Each test here contains a $30kg$ payload. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Motion	Direction	Envir.	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§11.2	Ascent	Forward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Forward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 15^o	O	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Observations

Similar to §11.1 the Husky with the added $30kg$ payload did not have an issue in the remaining static stability tests. The only issues that occurred were due to the physics simulation errors.

ISO/TR 23482-1: §12.1

In this sub-use case, we evaluate the Husky's dynamic stability by moving through 92 scenarios defined by ISO 23482-1. The scenarios consist of different slopes (6°,9°,15°). Although the tests in the standards have not been numerically provided, based on their information we categorized the test cases into 12.1, 12.2, 12.3, and 12.4. 12.1 and 12.3 use case considers the Husky without payload while 12.2 and 12.4 consider the Husky holding a $30kg$ square payload along with randomized forces.

Table F.4: ISO 23482-1 Husky §12.1 Evaluation by the property-based testing framework. Appendix D contains the test definitions. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Motion	Direction	Envir.	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§12.1	Ascent	Forward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 6^o	X	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 9^o	X	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Forward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 15^o	✓	✓	X	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 15^o	X	X	✓	✓	✓	X	X	X	✓	✓
	Descent	Forward	Slope 6^o	X	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 6^o	X	✓	X	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 6^o	X	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 6^o	X	X	✓	X	X	X	✓	✓	✓	✓
	Descent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 9^o	X	✓	X	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 9^o	X	X	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 9^o	X	✓	✓	X	X	X	✓	✓	✓	✓
	Descent	Forward	Slope 15^o	O	✓	✓	✓	✓	✓	✓	✓	O	✓
	Descent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 15^o	X	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 15^o	X	✓	✓	X	X	X	✓	✓	✓	✓

Observations

For §12.1 most of the failures occur due to the robot's planner being unable to be at the designated location by the time it was provided. However, it passes all the tests of ascent for the three different slopes. For §12.1.13-§12.1.24 the failures are a mixture of the planner (i.e. Navfn, dwa) not providing an ideal plan as well as orientation failures which were the primary failures that we were looking for. From tests T5 - T10, we increased the test time to allow for more time for the robot to execute its plan allowing for more successful behaviors as can be seen from the table. Additionally, at times the robot would spawn into the terrain causing it to move erratically in the environment. These behaviors are logged as O in tests.

An interesting observation is that in §12.1.12 (Husky turning left on a 15° slope (ascent)), the Husky moves forward on the slope and then attempt to reverse consistently and loses its initial momentum causing it to slip in most of the §12.1.12 scenarios due to which it does not reach to its final position in time failing the test.

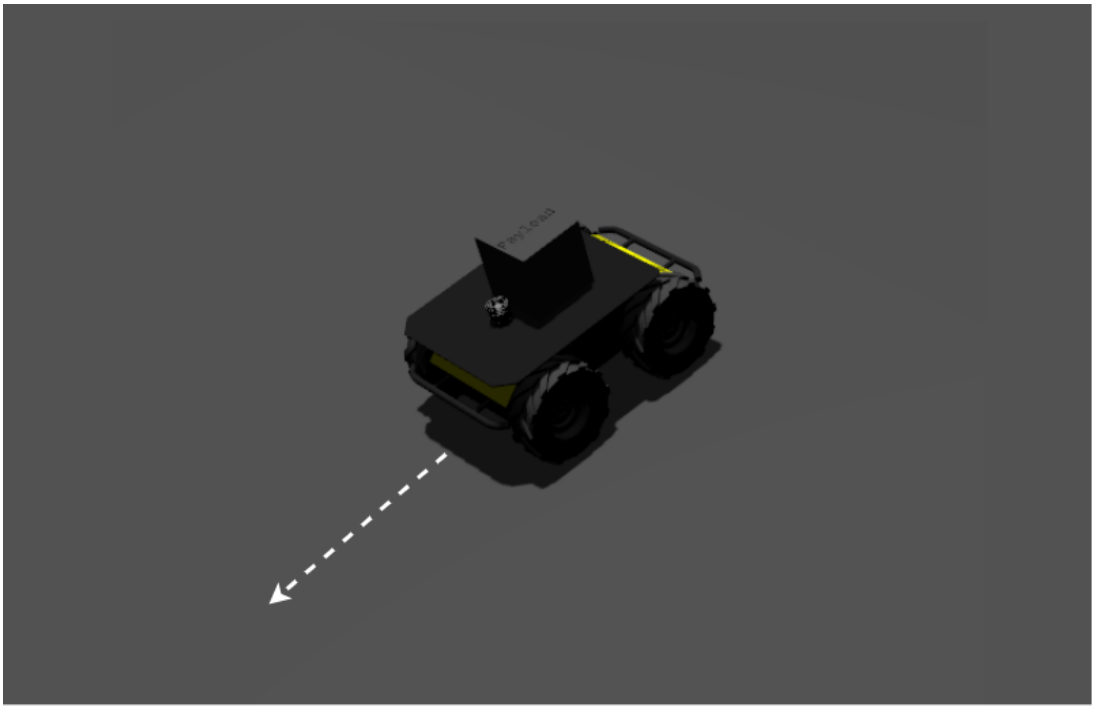


Figure F.4: §12 Husky Dynamic stability test on 9^o slope with a $30kg$ payload. The white arrow indicates the goal position that the Husky in navigating towards.

ISO/TR 23482-1: §12.2

Table F.5: ISO 23482-1 Husky §12.2 Evaluation by the property-based testing framework. Appendix D contains the test definitions. Each test here contains a $30kg$ payload. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Motion	Direction	Environ.	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§12.2	Ascent	Forward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Forward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 15^o	O	✓	✓	✓	X	✓	✓	✓	✓	✓
	Descent	Forward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 9^o	✓	✓	✓	X	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Observations

Due to the increase in time for the move_base in section 12, almost all the tests succeeded with the occasional failure due to incorrect pathing or physics simulation errors.

ISO/TR 23482-1: §12.3

The scenarios for §12.3 are the exact same as §12.1 only with the addition of a randomized force up to $100N$ being applied to the base_link of the Husky at x and y coordinates. The upper limit of force i.e. $100N$ should be selected according to the manufacturer's specification manual, however, in this case, the specification manual did not specify the maximum amount of force that could be applied on the Husky.

Table F.6: ISO 23482-1 Husky §12.3 Evaluation by the property-based testing framework. Appendix D contains the test definitions. Each test here has a randomized force of up to $100N$ applied. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Motion	Direction	Environ.	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§12.3	Ascent	Forward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 6^o	✓	✓	X	✓	✓	✓	X	✓	✓	✓
	Ascent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 9^o	✓	X	✓	✓	X	✓	X	✓	✓	✓
	Ascent	Forward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 15^o	✓	✓	X	✓	X	✓	X	✓	X	✓
	Descent	Forward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	X	✓	✓
	Descent	Right	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Observations

In the standard, the acceptance criteria of instability for when a UGV ascends or descends a terrain is not clearly explained, a tight tolerance of $X^o \pm 1$ was given to the UGVs where X corresponds to the angle of the terrain. For the cases such as §12.1 and §11.2 where no forces are applied, the majority of the tests passed and the robot was able to successfully reach its destination, however, in §12.3 and §12.4 where a randomized force that is applied to the base of the robot both on the X and Y directions with a varying force of up to $100N$, it would be expected that majority of the tests would fail, however, due to the bulky and stable design of the Husky, it maintained the correct orientation throughout the test and passed the majority of the tests.

If additional composite property tests were applied that were not in the standard such as the **MustBeAt**, then it would have failed plenty of times as it has been navigating according to its odometry and believing itself to be in the correct position but due to the force that was being applied it would end up not at the designated position and hence would fail the test.

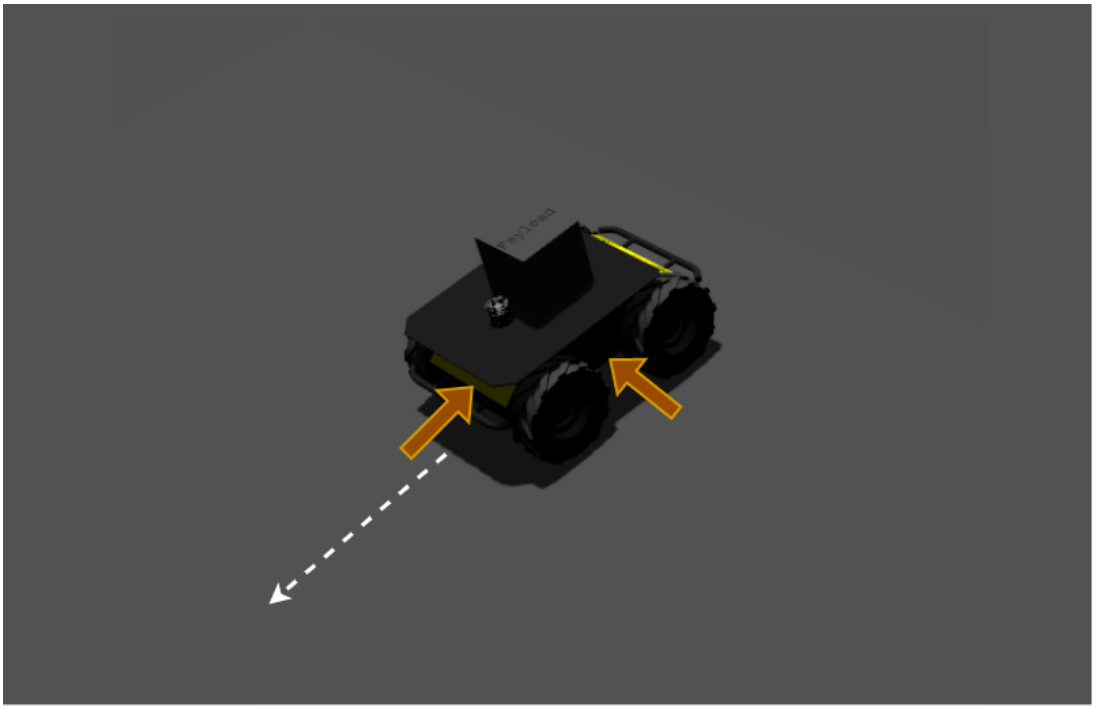


Figure F.5: §12 Husky Dynamic stability test on 9^o slope with a $30kg$ payload. The white arrow indicates the goal position that the Husky in navigating towards. The orange arrows indicate the randomized forces being applied on the x and y axis on the robot to de-stabilize it.

The scenarios for §12.4 are the exact same as §12.2 only with the addition of a randomized force up to $100N$ being applied to the base_link of the Husky at x and y coordinates. However, no force is applied to the payload.

ISO/TR 23482-1: §12.4

Table F.7: ISO 23482-1 Husky §12.4 Evaluation by the property-based testing framework. Appendix D contains the test definitions. Each test here has a randomized force of up to $100N$ applied. Additionally, it contains a $30kg$ payload. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Motion	Direction	Envir.	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§12.4	Ascent	Forward	Slope 6^o	O	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 6^o	✓	X	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	O	✓
	Ascent	Forward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	Slope 15^o	✓	✓	X	✓	X	✓	✓	✓	X	X
	Descent	Forward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 6^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 9^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Backward	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Left	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Right	Slope 15^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Observations

Despite adding an additional payload and payload orientation test for the robot while it was navigating up and down the terrain, no spike in failure cases occurred. Similar results to §12.1 were produced.

ISO/TR 23482-1: §13.2

Table F.8: ISO 23482-1 Husky §13.2 Evaluation by the property-based testing framework. Appendix D contains the test definitions. Each scenario contains a surface plane with a 35μ friction coefficient that tests the slippage of either the left wheel or the right wheel of the Husky. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Brake	Robot Side	Payload	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§13.2	Gradual	Left	n/a	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Gradual	Right	n/a	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Quick	Left	n/a	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Quick	Right	n/a	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Gradual	Left	$30kg$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Gradual	Right	$30kg$	✓	O	✓	✓	✓	✓	✓	✓	✓	✓
	Quick	Left	$30kg$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Quick	Right	$30kg$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Observations

Noted errors were due to path planners taking too much time.

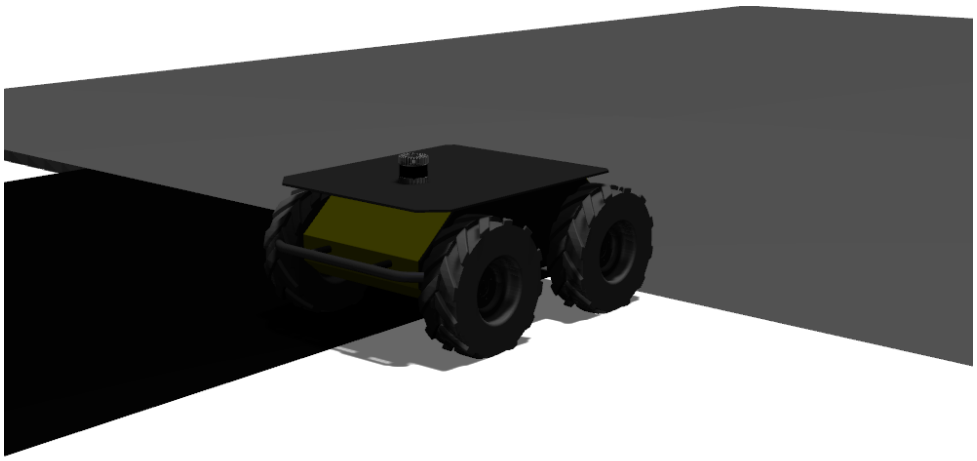


Figure F.6: §13.2 Husky Slippage test. Black surface has 75μ coefficient and white surface has 35μ coefficient.

ISO/TR 23482-1: §13.3

Table F.9: ISO 23482-1 Husky §13.3 Evaluation by the property-based testing framework. Appendix D contains the test definitions. Each scenario contains a 6^o slope platform that tests the quick braking capability of the Husky. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Motion	Direc.	Payload	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§13.3	Ascent	Forward	n/a	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Cross	n/a	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	n/a	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	360^o Turn	n/a	X	X	✓	X	✓	✓	✓	X	X	X
	Ascent	360^o Turn	n/a	✓	✓	✓	✓	✓	X	X	X	X	X
	Descent	Cross	n/a	✓	X	✓	X	✓	✓	✓	✓	✓	✓
	Ascent	Left	n/a	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	n/a	✓	X	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Left	$30kg$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Right	$30kg$	X	✓	✓	X	X	✓	✓	✓	✓	✓
	Descent	Left	$30kg$	✓	X	X	X	✓	✓	✓	✓	✓	✓
	Descent	Right	$30kg$	X	X	✓	X	✓	✓	✓	✓	✓	✓

Observations

The Husky successfully braked in most of the cases, however, as the braking was part of its autonomous behavior, at times its path planner took too much time due to which some cases failed.

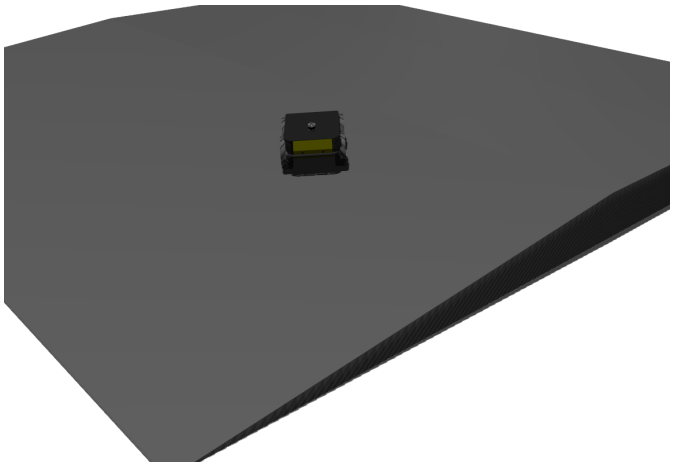


Figure F.7: §13.3 Husky Braking test on 6^o slope.

ISO/TR 23482-1: §13.4

Table F.10: ISO 23482-1 Husky §13.4 Evaluation by the property-based testing framework. Appendix D contains the test definitions. Tests are for the braking capability of the Husky in different environments with different speeds. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Motion	Direc.	Speed	Envir.	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§13.4	Ascent	Forward	Max.	Steps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Cross	Max.	Steps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ascent	Forward	Min.	Steps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Max.	Steps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Cross	Max.	Steps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Descent	Forward	Min.	Steps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	-	Forward	Max.	Gaps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	-	Cross	Max.	Gaps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	-	Forward	Min.	Gaps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Observations

The Husky successfully braked and stopped in the designated time in all of §13.4 scenarios.

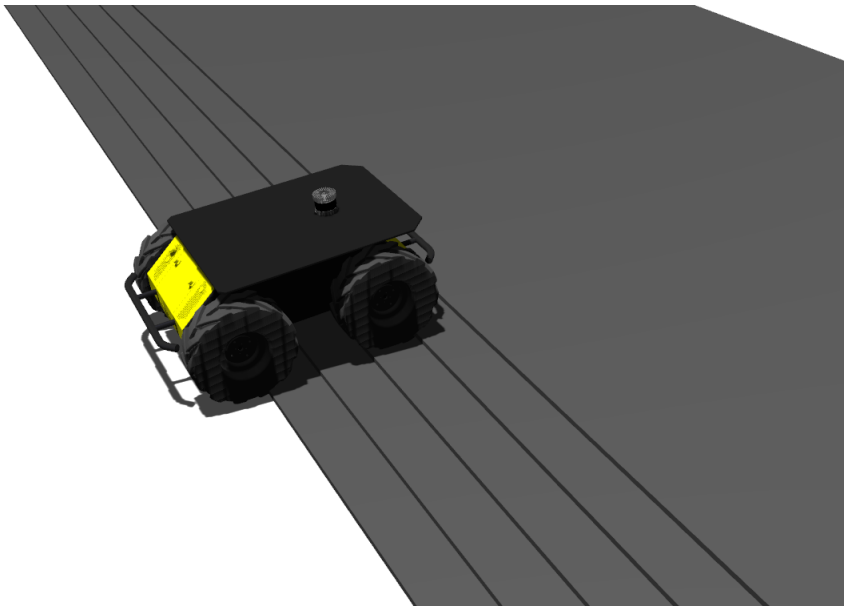


Figure F.8: §13.4 Husky Braking test on steps

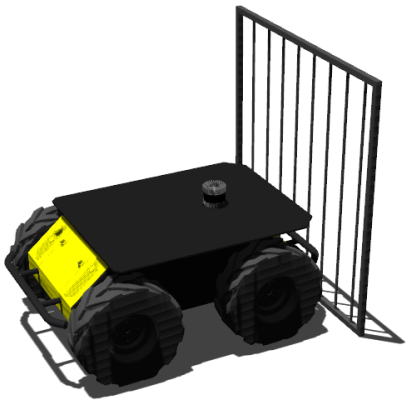
ISO/TR 23482-1: §15.1

Table F.11: ISO 23482-1 Husky §15.1 Evaluation by the property-based testing framework. Appendix D contains the test definitions. The tests verify the halting capability of the Husky when faced with an obstacle. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Direction	Obstacle	Obst. θ	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§15.1	Forward	Square block	0^o	X	X	X	X	X	X	X	X	X	X
	Backward	Square block	0^o	X	X	X	X	X	X	X	X	X	X
	Left	Square block	0^o	X	X	X	X	X	X	X	X	X	X
	Right	Square block	0^o	X	X	X	X	X	X	X	X	X	X
	Forward	Square block	90^o	X	X	X	X	X	X	X	X	X	X
	Forward	Square block	45^o	X	X	X	X	X	X	X	X	X	X
	Forward	Square block	180^o	X	X	X	X	X	X	X	X	X	X
	Forward	Square block	270^o	X	X	X	X	X	X	X	X	X	X
	Forward	Clear block	0^o	X	X	X	X	X	X	X	X	X	X
	Backward	Clear block	0^o	X	X	O	X	X	X	X	X	X	X
	Left	Clear block	0^o	X	X	X	X	X	X	X	X	X	X
	Right	Clear block	0^o	X	X	X	X	X	X	X	X	X	X
	Forward	Clear block	90^o	X	X	X	X	X	X	X	X	X	X
	Forward	Clear block	45^o	X	X	X	X	X	X	X	X	X	X
	Forward	Clear block	180^o	X	X	X	X	X	X	X	X	X	X
	Forward	Clear block	270^o	X	X	X	X	X	X	X	X	X	X
	Forward	Pole	0^o	X	X	X	X	X	X	X	X	X	X
	Backward	Pole	0^o	X	X	X	X	X	X	X	X	X	X
	Left	Pole	0^o	X	X	X	X	X	O	X	X	X	X
	Right	Pole	0^o	X	X	X	X	X	X	X	X	X	X
	Forward	Pole	90^o	X	X	X	X	X	X	X	X	X	X
	Forward	Pole	45^o	X	X	X	X	X	X	X	X	X	X
	Forward	Pole	180^o	X	X	X	X	X	X	X	X	X	X
	Forward	Pole	270^o	X	X	X	X	X	X	X	X	X	X
	Forward	Fence	0^o	X	X	X	X	X	X	X	X	X	X
	Backward	Fence	0^o	X	X	X	X	X	X	X	X	X	X
	Left	Fence	0^o	X	X	X	X	X	X	X	X	X	X
	Right	Fence	0^o	X	X	X	X	X	X	X	X	X	X
	Forward	Fence	90^o	X	X	X	X	X	X	X	X	X	X
	Forward	Fence	45^o	X	X	X	X	X	X	X	X	X	X
	Forward	Fence	180^o	X	X	X	X	X	O	X	X	X	X
	Forward	Fence	270^o	X	X	X	X	X	X	X	X	X	X

Observations

No obstacle-halting capabilities are available in the Husky's base version. Instead it path plans around the obstacles.



(a) §15.2: Obstacle fence

(b) §15.1: Obstacle pole

Figure F.9: In Sub-Fig. (a) §15.1, the Husky is expected to brake when close to an obstacle. In Sub-Fig. (b) §15.2 the Husky is expected to reduce its speed when near to an obstacle.

Observations

In §15.2, the default planner of the Husky plans too close to the obstacles and commonly collides with the object due to which MustNotCollide and MustNotBeNearTo have consistently failed.

ISO/TR 23482-1: §15.2

Table F.12: ISO 23482-1 Husky §15.2 Evaluation by the property-based testing framework. Appendix D contains the test definitions. The tests verify the speed reduction capability of the Husky when faced with an obstacle. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Direction	Obstacle	Obst. θ	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§15.2	Forward	Square block	0^o	✓	✓	✓	✓	✓	X	✓	✓	✓	✓
	Backward	Square block	0^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Left	Square block	0^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Right	Square block	0^o	✓	X	✓	✓	✓	✓	✓	✓	✓	✓
	Forward	Square block	90^o	✓	X	X	X	✓	✓	✓	✓	✓	✓
	Forward	Square block	45^o	✓	✓	✓	✓	✓	✓	✓	X	✓	✓
	Forward	Square block	180^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Forward	Square block	270^o	✓	✓	✓	✓	✓	X	✓	✓	X	✓
	Forward	Clear block	0^o	✓	✓	X	✓	X	✓	✓	✓	✓	✓
	Backward	Clear block	0^o	✓	✓	✓	✓	✓	X	✓	✓	✓	✓
	Left	Clear block	0^o	✓	✓	✓	✓	✓	✓	X	✓	✓	✓
	Right	Clear block	0^o	✓	✓	✓	✓	X	✓	✓	X	✓	✓
	Forward	Clear block	90^o	✓	✓	✓	X	X	✓	✓	✓	✓	✓
	Forward	Clear block	45^o	✓	✓	✓	✓	X	✓	✓	X	✓	✓
	Forward	Clear block	180^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Forward	Clear block	270^o	✓	✓	✓	✓	X	✓	✓	✓	✓	✓
	Forward	Pole	0^o	✓	✓	X	✓	✓	✓	✓	✓	✓	✓
	Backward	Pole	0^o	✓	✓	X	X	✓	✓	✓	✓	✓	✓
	Left	Pole	0^o	✓	✓	✓	✓	O	✓	✓	✓	✓	✓
	Right	Pole	0^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Forward	Pole	90^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Forward	Pole	45^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Forward	Pole	180^o	✓	✓	✓	X	✓	✓	✓	✓	✓	✓
	Forward	Pole	270^o	✓	✓	✓	X	✓	✓	✓	✓	✓	✓
	Forward	Fence	0^o	✓	✓	✓	O	✓	✓	✓	✓	✓	✓
	Backward	Fence	0^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Left	Fence	0^o	✓	✓	✓	✓	✓	✓	✓	O	✓	✓
	Right	Fence	0^o	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Forward	Fence	90^o	O	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Forward	Fence	45^o	✓	✓	✓	X	✓	✓	✓	✓	✓	✓
	Forward	Fence	180^o	✓	✓	✓	✓	O	✓	✓	✓	✓	✓
	Forward	Fence	270^o	✓	✓	✓	✓	✓	✓	✓	✓	X	✓

ISO/TR 23482-1: §15.3

Table F.13: ISO 23482-1 Husky §15.3 Evaluation by the property-based testing framework. Appendix D contains the test definitions. Each scenario contains a convex $5cm$ surface as defined in the standard ISO 23482-1 and tests the stopping or speed reduction of either the left wheel or the right wheel of the Husky. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Driving θ	Robot Side	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§15.3	90^o	-	X	X	X	X	X	X	X	X	X	X
	45^o	-	X	X	X	X	X	X	X	X	X	X
	10^o	-	X	X	X	X	X	X	X	X	X	X
	90^o	Left	X	X	X	X	X	X	X	X	X	X
	90^o	Right	X	X	O	X	X	X	X	X	X	X

Observations

All of the tests have failed as the stopping or reduction in speed has not been defined for the robot but instead deferred to the manufacturer's technical specification. However, in this case, the manufacturers have not provided any such information. It is expected that a methodology or formulation is provided by the standard in case of this situation, however, it is not provided.

ISO/TR 23482-1: §15.4

Table F.14: ISO 23482-1 Husky §15.4 Evaluation by the property-based testing framework. Appendix D contains the test definitions. Each scenario contains a concave $5cm$ surface as defined in the standard ISO 23482-1 and tests the stopping or speed reduction of either the left wheel or the right wheel of the Husky. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Driving θ	Robot Side	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§15.4	90^o	-	X	X	X	X	X	X	X	X	X	X
	45^o	-	X	X	X	X	X	X	X	X	X	X
	10^o	-	X	X	X	X	X	X	X	X	X	X
	90^o	Left	X	O	X	X	X	X	X	X	X	X
	90^o	Right	X	X	X	X	O	X	X	X	X	X

Observations

Similar to §15.3 all of the tests have passed as the stopping or reduction in speed has not been defined for the robot.

ISO/TR 23482-1: §16

Table F.15: ISO 23482-1 Husky §16 Evaluation by the property-based testing framework. Appendix D contains the test definitions. The test is for checking the autonomous behaviors of the Husky in an indoor environment traveling to four points. ✓ indicates that the tests have passed, X indicates that the tests have failed, and O indicates the test in which the simulation failed.

Section	Behavior	Goals	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
§16	Autonomous	4	X	X	X	X	X	X	✓	X	✓	X

Observations

The default planner provided by the Husky package is not tuned, and it generates plans close to obstacles due to which several tests have failed.

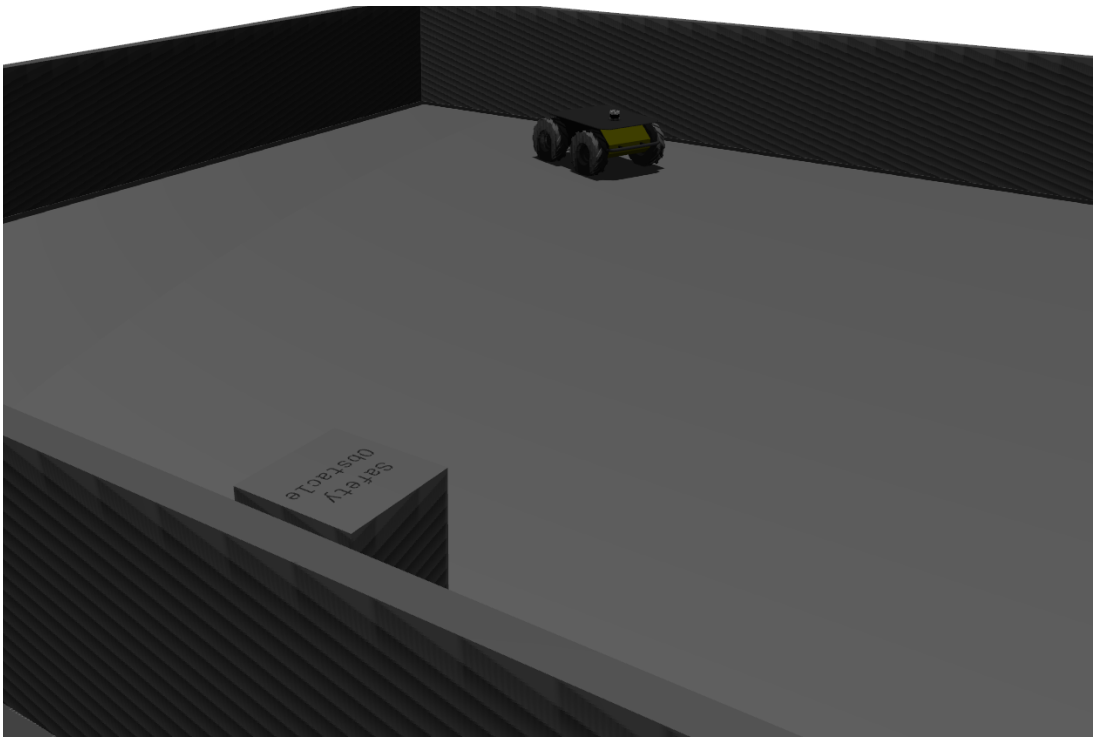
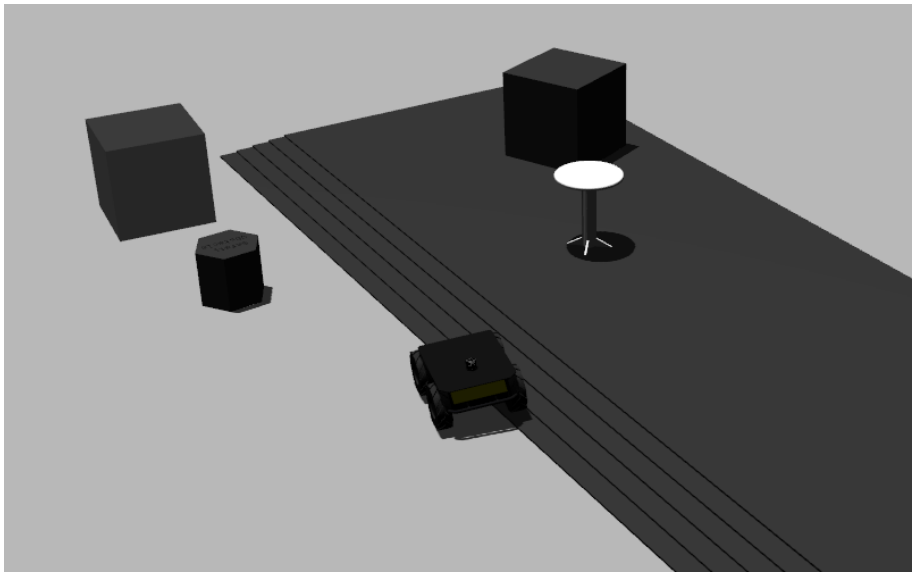


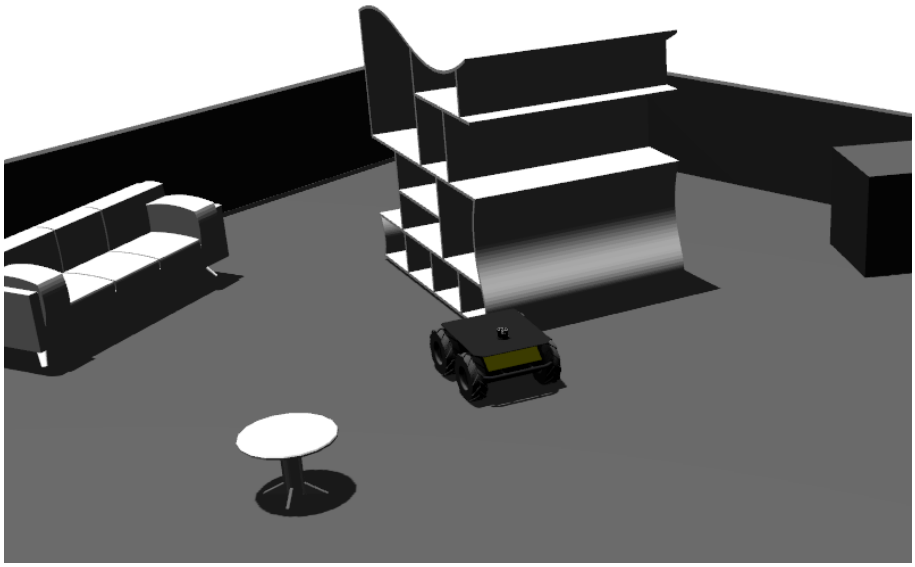
Figure F.10: §16 Husky autonomous navigation test.

ISO/TR 23482-1: §17

§17 is user-defined tests for verifying autonomous behavior. For evaluation, several test cases for were performed using randomized scenario generation in which the robot was given random goals in random environments, and the acceptance criteria was that the robot avoids obstacles. The results for the tests were similar to that of §16.



(a) Randomized scenario in which the Husky is tasked to navigate to the back of the cube that is next to the coffee table.



(b) Randomized scenario in which the Husky is tasked to navigate next to the sofa.

Figure F.11: §17 User tests. Randomized Scenario testing with random goals.

References

[1] ISO, "Application of ISO 13482—part 1: safety-related test methods," International Organization for Standardization, Geneva, CH, Standard, Mar. 2020.

[2] S. O. Sohail, A. Mitrevski, P. G. Plöger, and N. Hochgeschwender, "Property-Based Testing in Simulation for Verifying Robot Action Execution in Tabletop Manipulation," in 2021 European Conference on Mobile Robots (ECMR), 2021, pp. 1–7.

[3] G. Michalos, S. Makris, P. Tsarouchi, T. Guasch, D. Kontovrakis, and G. Chryssolouris, "Design Considerations for Safe Human-robot Collaborative Workplace," Procedia CIRP, vol. 37, pp. 248–253, 12 2015.

[4] M. Valori, A. Scibilia, I. Fassi, J. Saenz, R. Behrens, S. Herbster, C. Bidard, E. Lucet, A. Magisson, L. Schaake, J. Bessler, G. B. Prange-Lasonder, M. Kühnrich, A. B. Lassen, and K. Nielsen, "Validating safety in human–robot collaboration: Standards and new perspectives," Robotics, vol. 10, no. 2, 2021.

[5] ISO, "Robots and robotic devices – Collaborative robots," International Organization for Standardization, Geneva, CH, Standard, Mar. 2016.

[6] ——, "Industrial trucks—safety requirements and verification—part 4: driverless industrial trucks and their systems," International Organization for Standardization, Geneva, CH, Standard, Mar. 2020.

[7] IEC, "Medical Electrical Equipment—Part 2-78: Particular Requirements for Basic Safety and Essential Performance of Medical Robots for Rehabilitation, Assessment, Compensation or Alleviation," International Electrotechnical Commission, Geneva, CH, Standard, Mar. 2020.

[8] ISO, "Robots and robotic devices – Safety requirements for industrial robots - Part 1: Robots," International Organization for Standardization, Geneva, CH, Standard, Mar. 2011.

[9] P. Chemweno, L. Pintelon, and W. Decré, "Orienting safety assurance with outcomes of hazard analysis and risk assessment: A review of the ISO 15066 standard for collaborative robot systems," Safety Science, vol. 129, p. 104832, 09 2020.

[10] S. O. Sohail, A. Mitrevski, P. G. Plöger, and N. Hochgeschwender, "Automated Test Generation for Robot Self-Examination," H-BRS, 2020.

[11] N. Hochgeschwender, G. Cornelius, and H. Voos, "Arguing security of autonomous robots," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 11 2019, pp. 7791–7797.

[12] 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, vol. 3, 2004, pp. 2149–2154 vol.3.

[13] Nvidia. (2022) Nvidia Issac Sim. [Online]. Available: https://developer.nvidia.com/isaac-sim

[14] D. Kottlarz. (2022) MYBOTSHOP. [Online]. Available: https://mybotshop.de/

[15] D. EN, "Electrically powered wheelchairs, scooters and their chargers - Requirements and test methods; German version EN 12184:2014," European Standard, Sweden, Standard, Mar. 2014.

[16] Husky Specification Manual, Clearpath, 09 2022, rev. 1.1.3. Available at https://levelfivesupplies.com/wp-content/uploads/2020/08/Husky-UGV-User-Manual.pdf.

[17] ISO, "Robots and robotic devices – Safety requirements for industrial robots - Part 2: Robot systems and integration," International Organization for Standardization, Geneva, CH, Standard, Mar. 2011.

[18] C. Heer, Industrial Robots: Robot Investment Reaches Record 16.5 billion USD, ser. The new World Robotics. International Federation of Robotics, 2019. [Online]. Available: https://ifr.org/ifr-press-releases/news/robot-investment-reaches-record-16.5-billion-usd

[19] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" Transportation Research Part A: Policy and Practice, vol. 94, pp. 182–193, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965856416302129

[20] W. Wachenfeld and H. Winner, The Release of Autonomous Vehicles. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 425–449. [Online]. Available: https://doi.org/10.1007/978-3-662-48847-8_21

[21] L. Kunze, M. E. Dolha, E. Guzman, and M. Beetz, "Simulation-Based Temporal Projection of Everyday Robot Object Manipulation," in The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, ser. AAMAS '11. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011, p. 107–114.

[22] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, "Simulation environment for mobile robots testing using ROS and Gazebo," in 2016 20th International Conference on System Theory, Control and Computing (ICSTCC), 2016, pp. 96–101.

[23] L. Mösenlechner and M. Beetz, "Fast temporal projection using accurate physics-based geometric reasoning," in 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 1821–1827.

[24] D. Araiza-Illan, A. G. Pipe, and K. Eder, "Model-based Test Generation for Robotic Software: Automata versus Belief-Desire-Intention Agents," CoRR, vol. abs/1609.08439, 2016. [Online]. Available: http://arxiv.org/abs/1609.08439

[25] C. Harper and G. Virk, "Towards the Development of International Safety Standards for Human Robot Interaction," I. J. Social Robotics, vol. 2, pp. 229–234, 09 2010.

[26] ISO. (2022) Developing standards. [Online]. Available: https://www.iso.org/developing-standards.html

[27] 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 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST), 2018, pp. 331–342.

[28] A. Bihlmaier and H. Wörn, "Robot unit testing*," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8810, 2014, pp. 255–266.

[29] A. Santos, A. Cunha, and N. Macedo, "Property-based testing for the robot operating system," in A-TEST 2018 - Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation, Co-located with FSE 2018, ser. A-TEST 2018. New York, NY, USA: Association for Computing Machinery, 2018, pp. 56–62. [Online]. Available: https://doi.org/10.1145/3278186.3278195

[30] K.-T. Xie, J.-J. Bai, Y.-H. Zou, and Y.-P. Wang, "ROZZ: Property-based Fuzzing for Robotic Programs in ROS," in 2022 International Conference on Robotics and Automation (ICRA), 2022, pp. 6786–6792.

[31] V. D. Cosmo, A. Giusti, R. Vidoni, M. Riedl, and D. T. Matt, "Collaborative Robotics Safety Control Application Using Dynamic Safety Zones Based on the ISO/TS 15066:2016," in Advances in Service and Industrial Robotics. Cham: Springer International Publishing, 2020, pp. 430–437.

[32] J. Guiochet, Q. A. Do Hoang, M. Kaâniche, and D. Powell, "Applying Existing Standards to a Medical Rehabilitation Robot: Limits and Challenges," in Safety in Human-Robot Coexistence & Interaction, Vilamoura, Portugal, Oct. 2012. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01282195

[33] R. M. J. and K. J., "Safe human-robot-collaboration-introduction and experiment using ISO/TS 15066," 2017 3rd International Conference on Control, Automation and Robotics (ICCAR), pp. 740–744, 2017.

[34] G. Herrmann and C. Melhuish, "Towards Safety in Human Robot Interaction," I. J. Social Robotics, vol. 2, pp. 217–219, 09 2010.

[35] Y. H. Weng, G. Virk, and S. Yang, "The Safety for Human-Robot Co-Existing: On New ISO 13482 Safety Standard for Service Robotn," Internet Law Review, vol. 17, 2015.

[36] ISO, "Robots and robotic devices — Safety requirements for personal care robots," International Organization for Standardization, Geneva, CH, Standard, Mar. 2014.

[37] V. E. F. and J. G. A., "Robots, standards and the law: Rivalries between private standards and public policymaking for robot governance," Computer Law & Security Review, vol. 35, no. 2, pp. 129–144, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0267364918302863

[38] V. Estivill-Castro, R. Hexel, and C. Lusty, "Continuous integration for testing full robotic behaviours in a GUI-stripped simulation," in CEUR Workshop Proceedings, vol. 2245, 2018, pp. 453–464.

[39] D. MacIver, Z. Hatfield-Dodds, and M. Contributors, "Hypothesis: A new approach to property-based testing," Journal of Open Source Software, vol. 4, p. 1891, November 2019.

[40] D. Baev, "Allure Framework," 2014. [Online]. Available: https://docs.qameta.io/allure/

[41] L. . Pitonakova, M. Giuliani, A. Pipe, and A. Winfield, "Feature and Performance Comparison of the V-REP, Gazebo and ARGoS Robot Simulators," in Part of the Lecture Notes in Computer Science book series (LNAI,volume 10965), February 2018.

[42] T. Parr. (2022) ANTLR. [Online]. Available: https://www.antlr.org/

[43] Eclipse. (2022) Eclipse Xtext. [Online]. Available: https://www.eclipse.org/Xtext/

[44] I. Dejanović, R. Vaderna, G. Milosavljević, and v. Vuković, "TextX: A Python tool for Domain-Specific Languages implementation," Knowledge-Based Systems, vol. 115, pp. 1–4, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705116304178

[45] T. Parr. (2022) ANTLR. [Online]. Available: https://github.com/westes/flex

[46] I. Dejanovic, G. Milosavljevic, and R. Vaderna, "Arpeggio: A flexible PEG parser for Python," Knowledge-Based Systems, vol. 95, pp. 71–74, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705115004761

[47] F. Vaz, D. Portugal, A. Araujo, M. Couceiro, and R. Rocha, "A localization approach for autonomous underwater vehicles: A ROS-Gazebo framework," arXiv, 11 2018.

[48] Autodesk, "Fusion 360 cad modeling software." [Online]. Available: https://www.autodesk.de

[49] CHAMP, "Quadruped controller." [Online]. Available: https://github.com/chvmp/champ

[50] Clearpath, "Clearpath jackal repository." [Online]. Available: https://github.com/jackal/jackal

[51] E. Parliament, "Directive 2006/42/EC on Machinery; European Parliament: Brussels, Belgium," European Parliment, Brussels, Belgium, Standard, Mar. 2006.

[52] D. Kottlarz. (2022) MYBOTSHOP Husky. [Online]. Available: https://www.mybotshop.de/Clearpath-Husky_1

[53] S. O. Sohail, T. Mehmood. (2022) MYBOTSHOP Documentation — Mobile Manipulation Platform Husky. [Online]. Available: https://www.docs.mybotshop.de/projects/robot_mmp_husky/html/index.html

[54] D. Kottlarz. (2022) MYBOTSHOP Jackal. [Online]. Available: https://www.mybotshop.de/Clearpath-Jackal-J100_1

[55] ——. (2022) MYBOTSHOP ROVO2. [Online]. Available: https://www.mybotshop.de/MBS-ROVO2_1

[56] ——. (2022) MYBOTSHOP xARM6. [Online]. Available: https://www.mybotshop.de/UFactory-xArm6_1

[57] ISO, "Safety aspects — Guidelines for their inclusion in standards," International Organization for Standardization, Geneva, CH, Standard, Mar. 2014.

[58] ——, "Robotics - Vocabulary," International Organization for Standardization, Geneva, CH, Standard, Mar. 2021.

[59] ——, "Manipulating industrial robots - Performance criteria and related test methods," International Organization for Standardization, Geneva, CH, Standard, Mar. 1998.

[60] ——, "Manipulating industrial robots — Mechanical interfaces — Part 1: Plates," International Organization for Standardization, Geneva, CH, Standard, Mar. 2004.

[61] ——, "Manipulating industrial robots — Mechanical interfaces — Part 2: Shafts," International Organization for Standardization, Geneva, CH, Standard, Mar. 2002.

[62] ——, "Robots and robotic devices - Coordinate systems and motion nomenclatures," International Organization for Standardization, Geneva, CH, Standard, Mar. 2013.

[63] ——, "Manipulating industrial robots - Presentation of characteristics," International Organization for Standardization, Geneva, CH, Standard, Mar. 1999.

[64] ——, "Robots for industrial environments - Automatic end effector exchange systems - Vocabulary," International Organization for Standardization, Geneva, CH, Standard, Mar. 2022.

[65] ——, "Safety of machinery — General principles for design — Risk assessment and risk reduction," International Organization for Standardization, Geneva, CH, Standard, Mar. 2010.

[66] ——, "Manipulating industrial robots - Informative guide on test equipment and metrology methods of operation for robot performance evaluation in accordance with ISO 9283," International Organization for Standardization, Geneva, CH, Standard, Mar. 1995.

[67] ——, "Safety of machinery - Safety-related parts of control systems - Part 1: General principles for design," International Organization for Standardization, Geneva, CH, Standard, Mar. 2015.

[68] ——, "Safety of machinery - Safety-related parts of control systems - Part 2: Validation," International Organization for Standardization, Geneva, CH, Standard, Mar. 2012.

[69] ——, “Safety of machinery—positioning of safeguards with respect to the approach speeds of parts of the human body,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2010.

[70] ——, “Safety of machinery—two-hand control devices—principles for design and selection,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2019.

[71] ——, “Manipulating industrial robots - Object handling with grasp-type grippers - Vocabulary and presentation of characteristics,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2000.

[72] ——, “Medical devices - Application of risk management to medical devices,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2019.

[73] ——, “Agricultural machinery and tractors—safety of highly automated machinery,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2018.

[74] ——, “Robotics - Performance criteria and related test methods for service robots - Part 1: Locomotion for wheeled robots,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2016.

[75] ——, “Robotics - Performance criteria and related test methods for service robots - Part 2: Navigation,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2019.

[76] ——, “Robotics - Performance criteria and related test methods for service robots - Part 3: Manipulation,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2021.

[77] ——, “Robotics - Performance criteria and related test methods for service robots - Part 4: Lower-back support robots,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2021.

[78] ——, “Mobile robots - Vocabulary,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2017.

[79] ——, “Robotics—safety design for industrial robot systems—part 1: end-effectors,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2018.

[80] ——, “Robotics—safety requirements for industrial robots—part 2: manual load/unload stations,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2017.

[81] ——, “Robotics - Modularity for service robots - Part 201: Common information model for modules,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2022.

[82] ——, “Robotics - Application of ISO 13482 - Part 2: Application guidelines,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2019.

[83] ——, “Risk management — Guidelines,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2018.

[84] ——, “Robotics - Application services provided by service robots - Safety management systems requirements,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2022.

[85] E. Parliament, “Use of Work Equipment Directive,” European Parliment, Brussels, Belgium, Standard, Mar. 2009.

[86] ——, “Workplace Directive,” European Parliment, Brussels, Belgium, Standard, Mar. 1989.

[87] ——, “Product Safety Directive,” European Parliment, Brussels, Belgium, Standard, Mar. 2001.

[88] ——, “Low Voltage Directive (LVD),” European Parliment, Brussels, Belgium, Standard, Mar. 2006.

[89] ——, “Electromagnetic Compatibility (EMC) Directive,” European Parliment, Brussels, Belgium, Standard, Mar. 2004.