

Ontology-based Testing for Quality Assurance of Robotics Applications

Shahzad Nayab and Franz Wotawa

Graz University of Technology, Institute of Software Engineering and Artificial Intelligence, Graz, Austria
nayab.shahzad@tugraz.at, wotawa@tugraz.at

Abstract—Testing autonomous robots in warehouse logistics is vital for safety especially when they interact with humans. This paper introduces a testing approach that uses formal environmental models, i.e., ontologies, to generate comprehensive test suites. We discuss an ontology for warehouse logistics and some challenges.

Keywords—test automation; automated test case generation; robot testing

1. INTRODUCTION

The Fourth Industrial Revolution, also known as Industry 4.0, requires highly autonomous and flexible robots that can safely interact with humans and other robots, such as automated guided vehicles (AGVs) used in warehouses and manufacturing facilities. Ensuring the safety of these AGVs involves adhering to standards like ISO 3691-4¹, which specify necessary precautions such as system detection, speed regulation, and effective braking mechanisms. Consequently, rigorous testing is crucial before deployment, specifically to identify critical interactions between robots and their environments. It is worth noting that for automated driving functions and autonomous driving, there has been considerable work focusing on system testing, where critical scenarios are of particular importance. For example, Li et al. [1] showed an approach that utilizes formal ontologies for test case generation. It is worth noting that the algorithms for compiling ontologies to input models have been improved recently [2]. The objective of this paper is to demonstrate, through a realistic case study, the effectiveness of ontology-based testing in exposing potential faults or unsafe behaviors. The paper details requirements, and ontology modeling, and discusses the challenges and solutions associated with this approach.

2. USE CASE

The study presents ontology-based testing for robotics through a warehouse navigation scenario, where a mobile robot moves between interconnected positions, encountering various fixed and moving obstacles. The robot must dynamically distinguish between temporary and permanent obstacles, recalculating routes or raising errors when necessary. Testing requires diverse scenarios, though a simplified single-pathway assumption is initially adopted.

Figure 1 depicts a case where a robot is positioned at state A and has to move to state C. Both possible paths have obstacles,

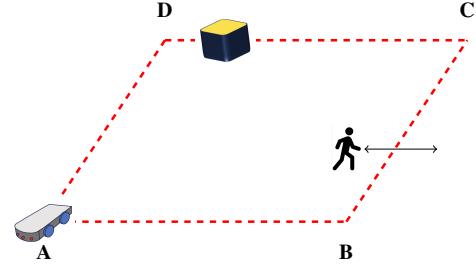


Figure 1. A robot in position A that needs to move to position C, facing a person and a parcel as obstacles.

but the one with the parcel on it is permanently blocked, and the other is by a moving person, where mitigation on the side of the robot under test is possible. For our use case, we know that we have to deal with the robot under test, the obstacles, the pathways, and their conditions, like the degree of light (e.g., dark) or slippery (e.g., because of cleaning the pathway). First, we introduce a concept *RobotExecEnvironment* that forms the top concept of the ontology. This concept is considered to handle all the information required to come up with a scenario. In each *RobotExecEnvironment*, we have the robot under test and obstacles. Therefore, we introduce corresponding concepts *RobotUT* and *Obstacle*. For each scenario, we have exactly one instance of *RobotUT* and 0 to k *Obstacle*. When restricting ourselves to simple pathway structures like the one given in Figure 1, we set k to 4. Both concepts *RobotUT* and *Obstacle* need to be placed on a pathway. Therefore, we further introduce a concept *Edge* that represents a pathway. We can place obstacles and the robot under test in a certain position of the pathway, e.g., from the beginning to the end, and then from left to right. Hence, we introduce attributes *where* and *orientation* to store this information. Let us go back to the concept *Edge*; this concept has one attribute *edge* for holding information on all existing edges. To represent the environmental conditions of pathways, we come up with a concept *EnvParameter* having two parameters *light* and *slippery* for storing information of the current light and slippery condition. This concept is in relation to *Edge*. However, for each *Edge*, we have exactly one set of parameters. The last part of the ontology is for storing different obstacles. We distinguish obstacles that are not mobile from mobile ones. Therefore, we introduce concepts *FixedObstacle* and *MovingObstacle*, respectively. *FixedObstacles* can be smaller or bigger, which is handled using an attribute *type*. *MovingObstacles* can be

¹See <https://www.controleng.com/ensuring-agv-safety-with-standards-compliance>, last accessed April 14th, 2025.

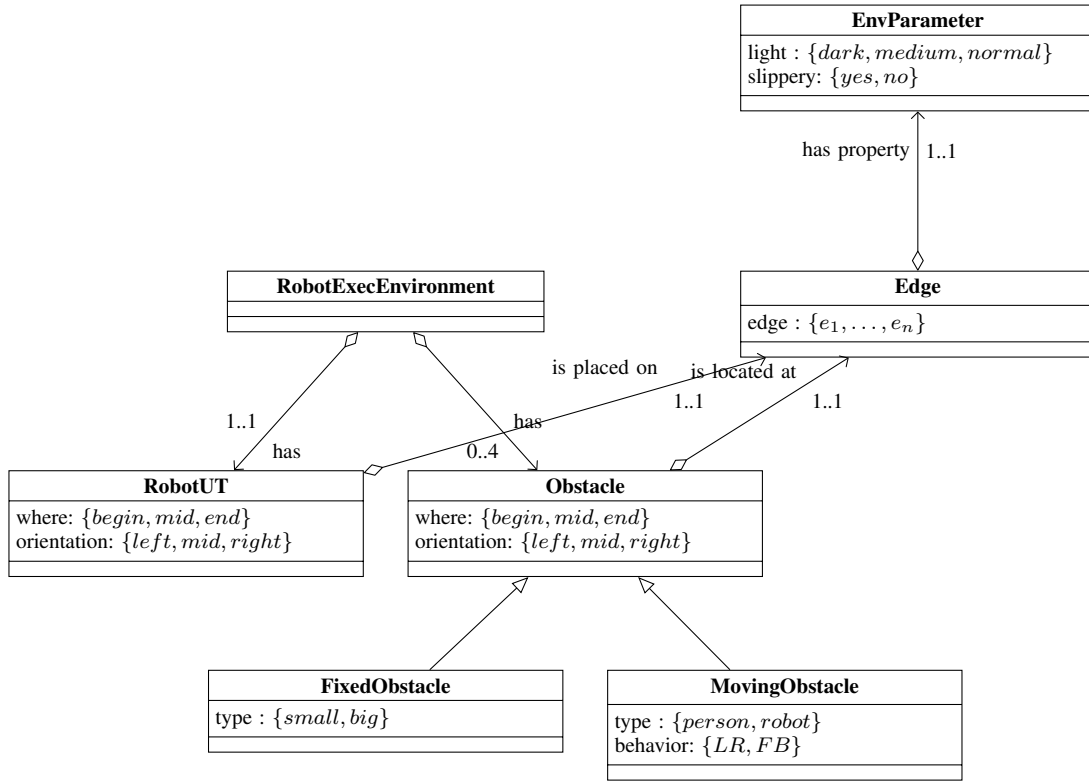


Figure 2. A simple ontology of an execution environment where a robot under test is assumed to find a way to a pre-defined end state, considering a graph-like path structure.

persons or other robots, either moving from one side of the pathway to the other (e.g., left to right) or from the beginning of the pathway to the end (e.g., forward and then to the back, etc). This information is stored in attributes *type* and *behavior*. In Figure 2, we depict this ontology using the UML class diagram notation.

Despite the ontology’s effectiveness, certain structural constraints arise. The developed ontology is inherently non-tree-structured. Hence, we need to develop a tree-structured variant before applying ontology-based testing. This is solved simply by duplicating key concepts like *Edge* and *EnvParameter* together with their relationship, and attaching the original with the *RobotUT* and *Obstacle* respectively. With these changes, we can use the algorithms provided (e.g., in [2]) to generate test cases. We compiled the tree-structured model into a combinatorial testing input model comprising 37 parameters with a domain size of $2 \dots 4$, and 10 constraints. Using ACTS [3] we were able to generate 49 test cases of strength 2 in 0.612 seconds. Hence, the presented approach is capable of developing test suites for robotics considering concepts and their relationships.

3. CONCLUSION

This paper explores ontology-based testing for robotic systems by outlining a structured approach that includes tools for ontology compilation and fault detection. A practical robotics use case demonstrates feasibility, highlighting the need for

comprehensive ontologies similar to those in autonomous driving². Future work involves enhancing the ontology tool and releasing it as an open-source platform.

ACKNOWLEDGMENT

This paper is part of the A-IQ Ready project that has received funding within the CHIPS JU in collaboration with the European Union’s Horizon Framework Programme and National Authorities under grant agreement No. 101096658. The work was partially funded by the Austrian Federal Ministry of Climate Action, Environment, Energy, Mobility, Innovation, and Technology under the FFG project FO999896574.

REFERENCES

- [1] Y. Li, J. Tao, and F. Wotawa, “Ontology-based test generation for automated and autonomous driving functions,” *Information and Software Technology*, vol. 117, Jan. 2020.
- [2] F. Wotawa, *On the Quest for Criticality Searching for Interactions that Matter When Hunting for Bugs*. Cham: Springer Nature Switzerland, 2025, pp. 356–373. [Online]. Available: https://doi.org/10.1007/978-3-031-92196-4_17
- [3] L. Yu, Y. Lei, R. Kacker, and D. Kuhn, “Acts: A combinatorial test generation tool,” in *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, 2013, pp. 370–375.

²See <https://www.asam.net/standards/asam-openxontology/>, last accessed 14th April 2025.