An approach to automatically generate test cases for AI-based autonomous heavy machinery

Iman Sonji, M.Sc.[1,3], Hannes Vietz, M.Sc.[1,3], Prof. Dr.-Ing. Christof Ebert[2,3], Prof. Dr.-Ing. Michael Weyrich[1,3]

[1]Institute of Industrial Automation and Software Engineering, University of Stuttgart
Pfaffenwaldring 47, 70569 Stuttgart, Germany
[2]Vector Consulting Services, Ingersheimer Str. 20, 70499 Stuttgart, Germany
[3]Robo-Test c/o University of Stuttgart, IAS
Iman.sonji@ias.uni-stuttgart.de
Hannes.vietz@ias.uni-stuttgart.de
Christof.ebert@vector.com
Michael.weyrich@ias.uni-stuttgart.de

Abstract: Testing autonomous systems is challenging due to the self-adaptive behavior and unpredictable environments of such systems. With the increase of AI-based functionalities in autonomous systems, release and homologation become difficult with the lack of transparency and software explainability. For instance, maintaining a valid safety case is hardly possible with learning and adaptive systems and software deliveries over the air. Therefore, with frequent software updates and continuous development streams, test cases must be generated automatically while at the same time providing coverage (e.g., indicating progress with KPIs), efficiency (e.g., limiting the amount of regression testing) and transparency (e.g., showing how specific corner cases are tested in case of accidents). In software testing, ontologies have demonstrated that they can assist in generating test cases as they encode domain knowledge in a more structured and machine-readable format. A case study shows how requirements and ontology-based formalized knowledge addresses systematic testing of autonomous heavy equipment. This paper provides a scenario-based testing method for automatically generating test cases for AI-based autonomous systems. Starting from the requirements, the proposed methodology is to map expert domain knowledge into a formal ontology-based model and then generate test cases with the assistance of this model.

Keywords: Autonomous Systems, Testing, Scenarios, Test Cases, AI-based, Complexity, Transparency

1 Introduction

Autonomous construction and heavy machinery have a vast potential since they can operate in hazardous or toxic environments that can be dangerous for humans. More specifically, earth-moving excavators are often used in removing hazardous materials such as barrels or other industrial waste. As contamination poses risk to humans, automation of

such systems and ensuring their safety becomes necessary. Verification and validation of an autonomous excavator become a great challenge since it is hard to collect data in such environments to train and test the system. However, development and testing require data, and without sufficient amount of the "right" data, AI-based automation solutions cannot be developed, and the safe operation of these solutions cannot be proven.

An industrial autonomous system is defined in [8] as a delimited technical system that can operate in an unpredictable environment systematically and without external intervention. A characteristic of many autonomous systems is high system complexity, which stems from uncertain, complex operating environment that necessitates the use of various sensors. An in-depth discussion of definitions and aspects of industrial autonomous systems is given in [8]. Due to the high complexity of such systems, not all functional and safety requirements can be identified from the beginning in the development phase [2, 3].

Another characteristic of autonomous systems is that they often operate in highly dynamic and unknown environments. This increase of environment complexity makes it infeasible to test for all different environment settings. Due to cost and time constraints, only a limited number of test cases can be run which should still provide high coverage of the system requirements. In high-safety systems, any failure that cause harm to the environment or the system itself is not acceptable. Therefore, an effective and efficient test case generation approach is required to verify the safety of such autonomous systems.

In this paper, we dive into the challenges and the systematic methods for testing autonomous systems in section 2. Focusing on scenario-based testing in relation to ontology-based domain knowledge, an approach for generating test cases and simulating synthetic data for an AI-based excavator is proposed in section 3. Finally, research highlights and possible future work are discussed in section 4.

2 Basics and Related Work

2.1 Challenges of Testing Autonomous Systems

The challenges behind testing autonomous systems come from the need for efficiency, safety, and quality [3]. The most common challenges in achieving systematic testing are primarily related to unpredictable environments, estimating coverage, and lack of transparency.

1. *Test efficiency despite high environment complexity:* The unpredictable and high complexity of the system environment leads to an explosion of parameters and scenarios that can be explored. However, exploring and testing all environmental conditions is inefficient and infeasible. This high complexity leads the development engineer to miss out on operating conditions (i.e. influential factors) that might be unknown during design time or fail to foresee a combination of known parameters that might lead to the system's failure. Since we cannot test for all possible combinations, the research question would be: How can we efficiently derive test cases that reveal unknown knowledge or identify faults in the autonomous system?

2. *Estimating degree of safety based on test cases:* Unknowns and uncertainties added to the testing process make it challenging to determine how safe the system is. High coverage is usually achieved with a high number of test cases. Since this easily becomes inefficient, the research question would be: How to derive sound conclusions about safety and quality based on the selected test cases?

3. *Achieving transparency:* The black box nature of AI-based systems and the lack of traceability of requirements make it challenging to explain failures or unpredictable behaviors of the selected test cases. Because of the high system complexity, the influential factors and which parameters to tune might be unknown to the test engineer. Since transparency is important for system verification, the research question would be: How to explain the results of the selected test if something goes wrong?

The development and testing of autonomous systems is complex, and the corresponding processes must be systematic and coupled with continuous improvement character. Scenario-based testing and ontology models are explored in the framework of systematic testing.

2.2 Scenario-Based Testing (SBT)

Due to the increasing complexity of autonomous systems, conventional methods such as the function-based approach or the distance-based approach are no longer sufficient. In the classic function-based approach, system functions are described in detail, forming the basis for testing. But an unambiguous definition of system functions is often too time-consuming for autonomous systems. In the distance-based approach, a vehicle should cover a certain distance without accidents to be considered safe. However, Klara et al. [6] suggest that the autonomous system should drive 11 billion miles to demonstrate that their failure rate is 20% better than that of a human driver, making this approach inefficient.

SBT can be seen as an evolution of the function-based and distance-based approaches [2, 11]. The primary artifact in SBT is the scenario which is defined according to Ulbrich et al. [10] as the temporal sequence of scenes linked by respective actions and events. Subsequently, Menzel et al. [7] represented scenarios in three abstraction layers: functional (semantic level), logical (state space level), and concrete scenarios (distinct values). A test case can be defined as a concrete scenario enriched by suitable evaluation criteria. For validation, scenarios that describe environmental characteristics and specify the goals of the autonomous system are defined. Moreover, operational design domain (ODD) has become an important topic in scenario-based testing to define the environmental operating conditions in which your autonomous system can safely operate in [4]. ODD is used as a systematic measure to verify that the test scenarios cover the system requirements.

There are two different approaches to identify and derive scenarios: a data-driven and a knowledge-based approach [4]. The data-driven approach deploys AI techniques on real-world data to generate scenarios. However, due to data scarcity, similar to our case, a knowledge-based approach can be adopted where scenarios are identified from experts' knowledge. The automation of this process requires this knowledge to be represented in a formalized format to represent the functional and logical scenarios. Ontologies are

usually used to formalize this knowledge. A base setup for SBT is proposed in Figure 1 which will be explained in detail in section 3.

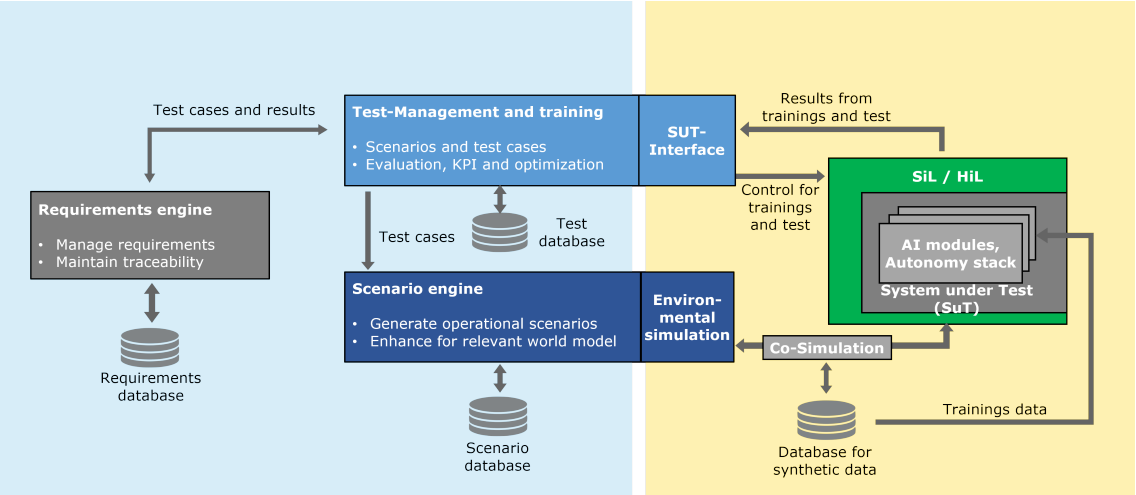

Figure 1: Proposed Framework for Scenario-Based Testing

Scenario-based testing can therefore be used to meet the challenges posed by the high complexity of autonomous systems. By reusing scenarios during further system developments, they also form a good basis for regression testing. However, a complete set of scenarios is still impossible during design time due to many unforeseeable conditions (i.e. unknown unknowns). Moreover, some additional challenges arise when implementing SBT such as:

- Providing the right data: Concrete test data that precisely represent a specific test case of a certain scenario are required.
- Generating scenarios systematically, meaningfully, and comprehensibly [3].

2.3 Ontologies

2.3.1 Basics and Definitions

According to Guarino et. al., "Ontology is a formal, explicit specification of a shared conceptualization" [5, p. 5]. Ontologies are used to represent knowledge in a formalized and machine-readable format. They contain standardized definitions of concepts that are used in a specific domain of knowledge. For example, in the field of heavy machinery, concepts such as landfill, barrel, excavator, or anything that exists in time and space in this domain can be defined in an ontology. Unlike taxonomies which define a hierarchical classification schema on concepts, ontologies define how these concepts relate or differ from each other. A formalized ontology uses first-order logic to describe these relations. Therefore, logical reasoning can be used to infer further knowledge from them.

Ontologies define concepts as *classes*, and the members of these classes are called *individuals*. A class consists of a set of actual things in the domain that share common

characteristics. For example, humans and vehicles can belong to the same class "DynamicObject". The relationships between members of classes are defined as *properties*. The relationships are usually directed, and they point from subject to object, resulting in *triples* in subject-predicate-object. According to Studer et. al. [9], ontologies can be structured into two parts, terminological boxes (known as TBox) which describe the concepts of a specific domain, and assertional boxes (known as ABox) which describe instances of the classes and facts observed from the situational knowledge. For example, defining the triple "RoadVechiles-driveOn-Road" is part of the TBox (terminologies). However, defining the triple "CarA-driveOn-Bahnhofstrasse" is part of the ABox (assertions). Reasoners can then be used to identify missing or additional knowledge or conflicts in concepts from terminological and assertional boxes. Automated reasoning can be applied easily to find these associations between classes and their individuals.

2.3.2 Related Work

Ontologies are increasingly used in the validation and verification process of autonomous systems, especially in automated driving [4] due to their ability to formalize domain knowledge of a specific domain. In [1], Bagschik et. al. propose a knowledge-based approach for scene generation in natural language for testing automated vehicles using ontologies. According to [1], assertional boxes can be used to describe real-world scenes extracted from a set of world terminologies (entities and relations). Functional scenarios were represented on a semantic level using Web Ontology Language (OWL) which was used as a basis for generating concrete scenarios. An ontology-based model was created based on expert knowledge, guidelines that describe infrastructure in Germany, and road traffic guidelines.

Xiong in [12] presented an orchestration of scenarios based on an ontology-based scenario representation. A framework was built to simulate test cases for automated vehicles. The ontology depicted in the framework consisted of concepts and relations between the driving context, actions and events, and temporal relations between different entities during simulation. Another common use case of ontologies is scene understanding which is described in [13]. The paper describes how ontologies can be used to model spatio-temporal relations between different participants in a scene. The ontologies are then applied to infer how these participants would interact with the vehicle under test in real-time.

Ontologies have a potential to provide a suitable framework to support achieving systematic scenario-based testing as they provide a common understanding and standardization of domain knowledge. In addition, unambiguous definitions of terms regarding individual functionalities and environment dimensions allow sharing and reuse of scenarios. However, as most of the research on ontologies was applied in the on-road automated driving domain, different terminologies and concepts need to be identified to create a basis for generating off-road scenarios.

3 Approach

Existing testing methods either focus on deriving critical tests for corner cases, or focus on testing one particular aspect of the autonomous system (e.g., collision avoidance). Moreover, in common scenario-based testing methods, scenarios always appear as an ideal starting point for the derivation of test cases and suitable safety criteria. However, adding requirements in the automated testing framework and ensuring traceability can bring light in identifying inconsistencies or missing safety requirements unknown at design time. The process of identifying requirements as a starting point and ensuring their correctness and consistency could play an important role in deriving scenarios and test cases.

In this paper, we propose a scenario-based approach based on requirements, knowledge-based ontology models, and synthetically generated data for the development and testing of autonomous systems. The use case is the development and testing of autonomy functions of an autonomous 24-ton excavator. The proposed approach is shown in Figure 1. As discussed, the systematic and meaningful generation of scenarios is crucial for successfully developing and testing autonomous systems. The following sections describe the training- and testing loop, focusing on the blue shaded region of the architecture.

3.1 Identifying requirements and functional scenarios

This phase includes gathering knowledge about the use cases of an excavator and the infrastructure of different types of landfills. Based on expert knowledge, ISO 17757, and in compliance with SOTIF methods, we identify requirements, base scenarios, and scenario dimensions for a 24-ton excavator.

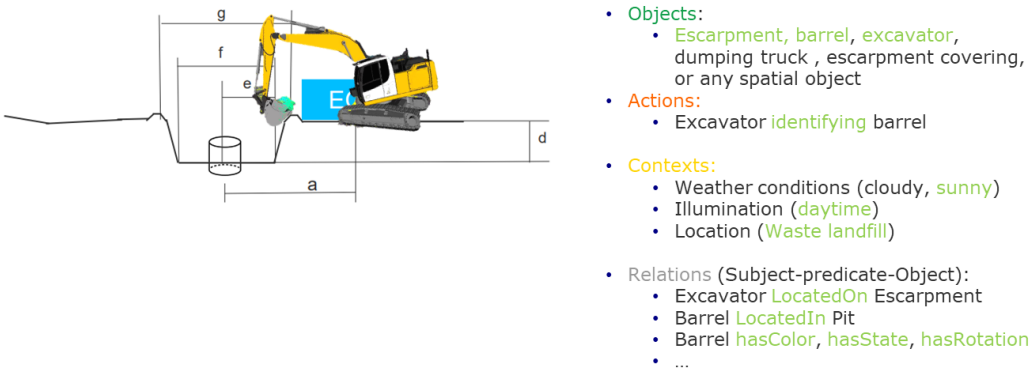

Figure 2: Semantic Description of a Functional Scenario

The functional scenarios are sketched and semantically described based on four main elements: Objects, Actions, Context, and Relations. The semantic description of scenarios represents the base for building an ontology model by modeling the terminologies/concepts as entities and modeling the relationships and dependencies between the scenario elements. A sketch of a functional scenario and its description are shown in Figure 2 based on the use case “Barrel Identification”.

3.2 Generating test cases

This phase is responsible for enriching the abstract scenarios with semantic data to form meaningful and useful scenarios. The functional scenarios are therefore converted to logical scenarios by adding the scenario parameters and their ranges. The ontology defines the parameters identified in the functional scenario and their relations to other parameters in the state space (TBox). Therefore, the ontology would form the basis for scenario generation. For example, the ontology would always automatically require defining the parameters with the dimensions relevant for the scenario. To ensure that the scenario remains valid when certain parameter values are varied, the dependencies between parameters should be also considered. Therefore, the ontology also defines dependencies between parameters and whether one can influence another parameter. For example, the weather (rain or sunny) influences the wetness of the landfill soil. These constraints should also be specified in the TBox or described using other forms of semantic rules.

The concrete scenarios are then derived from the logical scenario. Using logical reasoning, the validity and the meaningfulness of the scenario can be verified by identifying any logical inconsistencies or missing information. The concrete scenarios are then supported with evaluation criteria (pass/fail) to form proper test cases. The role of the ontology in this phase is to describe the environment and formalize expert knowledge about landfills' topology, its elements, and their relations between them. The primary use cases are summarized as follows:

- Support in describing functional scenarios by providing definitions for scenario elements.
- Provide correlations between dimensions (e.g., rain implies cloudy sky).
- Verify concrete scenarios and add missing information (e.g., infer certain parameter value based on chosen values)

3.3 Executing test cases and analyzing results

The concrete scenarios are then simulated to generate synthetic data. The development of autonomy functions for a 24-ton excavator focuses on using the camera and lidar sensors, for instance in the safe identification of half-buried barrels. Unreal Engine was selected as a simulation tool since photorealistic camera data is needed. As shown in Figure 3, the Unreal Engine is a powerful simulation tool compatible with various CAD 3D modeling formats and can render photorealistic camera data. Unreal Engine is controlled via the standard Universal Scene Description interface.

The test results achieved in the executed scenarios are analyzed to close the training and test loop. The first step is to assess whether test cases are classified as pass or fail, and which requirements have already been met since all the associated tests have been passed. Further analysis enables intelligent cognitive testing; for example, the identification of challenging scenarios and the execution of additional loop runs that focus on corner cases and possibly bring to light errors in or missing requirements.

Figure 3: Model of excavator used in Unreal engine in URDF format

A new loop run is also necessary if functions are refined, or new functions are developed. The training- and testing loop provides support here, particularly in regression tests. If necessary, further scenarios can be added as well. By distinguishing and describing logical scenarios separately from concrete scenarios, the maintenance effort is minimized since the logical scenarios remain independent of simulation tools.

4 Conclusion and Future Work

The constant improvement of autonomous systems challenges the development of such systems. As a result of vulnerabilities becoming known, the need for continually learning artificial intelligence arises. This results in continuous development, and corresponding processes must be adapted to this. For this purpose, a training and test loop that supports iterative development and realizes scenario-based testing was discussed. Ontologies are suggested as a knowledge-based system to aid in the process of systematically generating test cases.

One of the most important factors for the development of autonomous systems is data, which is needed on the one hand for the training of artificial intelligence and on the other hand for the testing of the system. To provide the right data for scenario-based development or testing, the example of the development of an autonomous 24-ton excavator was used to show how synthetically generated data can be used. By using synthetically generated data, the challenges of continuous development can be solved. Future work includes the development of a web application for managing scenarios and test cases.

References

[1] G. Bagschik, T. Menzel, and M. Maurer. Ontology based scene creation for the development of automated vehicles. IEEE, 2018.

[2] C. Ebert and R. Ray. Test-driven requirements engineering. *IEEE Software*, 38(1): 16–24, 2021.

[3] C. Ebert, M. Weyrich, B. Lindemann, and S. P. Chandrasekar. Systematic testing for autonomous driving. *ATZelectronics worldwide*, 16(3):18–23, 2021.

[4] J. Erz, B. Schütt, T. Braun, H. Guissouma, and E. Sax. Towards an ontology that reconciles the operational design domain, scenario-based testing, and automated vehicle architectures. In *2022 IEEE International Systems Conference (SysCon)*, pages 1–8, 2022.

[5] N. Guarino, D. Oberle, and S. Staab. What is an ontology? pages 1–17. Scholars Portal, 2009.

[6] N. Kalra and S. M. Paddock. *Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?* RAND Corporation, 2016.

[7] T. Menzel, G. Bagschik, and M. Maurer. Scenarios for development, test and validation of automated vehicles, 2018.

[8] M. Müller, T. Müller, B. Ashtari Talkhestani, P. Marks, N. Jazdi, and M. Weyrich. Industrial autonomous systems: a survey on definitions, characteristics and abilities. *at - Automatisierungstechnik*, 69(1):3–13, 2021.

[9] R. Studer, V. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1-2):161–197, 1998.

[10] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer. Defining and substantiating the terms scene, situation, and scenario for automated driving. pages 982–988. IEEE, 2015.

[11] H. Winner, K. Lemmer, T. Form, and J. Mazzega. Pegasus—first steps for the safe introduction of automated driving. Lecture Notes in Mobility, pages 185–195. Springer, 2019.

[12] Z. Xiong. Creating a computing environment in a driving simulator to orchestrate scenarios with autonomous vehicles. 2013.

[13] L. Zhao, R. Ichise, Y. Sasaki, Z. Liu, and T. Yoshikawa. Fast decision making using ontology-based knowledge base. pages 173–178. IEEE, 2016.