Kevin Daimi
Hamid R. Arabnia
Leonidas Deligiannidis (Eds.)

Communications in Computer and Information Science 2937

Security and Management, Wireless Networks, Software Engineering Research and Practice

24th International Conference, SAM 2025, 24th International Conference, ICWN 2025, and 23rd International Conference, SERP 2025, Held as Part of the World Congress in Computer Science, Computer Engineering, and Applied Computing, CSCE 2025
Las Vegas, NV, USA, July 21–24, 2025
Revised Selected Papers

Springer

Best Practice of Scenario-Based Modeling for Humanoid Software Validation

Sangho Lee[1], Janghwan Kim[2], and R. Young Chul Kim[2(✉)]

[1] Rastech, Daejeon 34037, Republic of Korea
project@rastech.co.kr
[2] Hongik University, Sejong 30016, Republic of Korea
{lentoconstante,bob}@hongik.ac.kr

Abstract. Recent advances in adapting artificial intelligence technology have revolutionized the field of humanoid robots. The intelligence of robots is based on reinforcement learning and AI models. However, due to the nature of robot software that continuously learns and reacts through interaction with the real environment, it isn't easy to validate safety, reliability, and accuracy using only existing software testing methodologies. To address these issues, we apply scenario-based modeling in software validation to humanoid robot software, incorporating reinforcement learning models. In particular, the 'mode selection' function of humanoid robots is presented as an application case, and a systematic procedure is performed from requirement analysis to scenario creation, refined scenario-based test case creation, and scenario execution and validation using a behavior tree. This is expected to contribute to maintaining the stability of humanoid robot systems by systematically detecting unpredictable behaviors in humanoid software and enhancing efficiency and reliability by validating the decision-making process of AI models using scenario-based modeling based on behavior trees.

Keywords: Scenario-based Modeling · AI Software Validation · Humanoid Software Validation

1 Introduction

Recent advances in artificial intelligence technology have brought about revolutionary changes in the field of robotics, and humanoid robots, in particular, represent a culmination of these technologies. Humanoid robots that mimic human form are utilized in a wide range of fields, including industry, service, and entertainment, due to their potential to perform complex tasks in various environments and interact naturally with humans [1]. The intelligence of these robot systems is mainly driven by artificial intelligence models, including reinforcement learning (RL) [2].

AI models mounted on humanoid robots, especially those using reinforcement learning, are challenging to validate existing software testing methodologies due to their characteristics of continuously learning and responding through interactions with the actual environment.

K. Daimi et al. (Eds.): CSCE 2025, CCIS 2937, pp. 486–497, 2026.
https://doi.org/10.1007/978-3-032-22208-4_34

To solve these problems, this paper proposes a scenario-based modeling (SBM) mechanism to verify the safety, reliability, and accuracy of humanoid robot software equipped with reinforcement learning models. In particular, this paper presents the 'mode selection' function of humanoid robots as an application case and analyzes in detail how the proposed SBM mechanism can be applied to actual robot systems and how it can be effective. In addition, we discuss a method to trace and verify the decision-making process of an AI model by utilizing a behavior tree as a scenario execution (test) method.

In Chap. 2, we discuss validation methods for robot software, drawing on related research, and in Chap. 3, we introduce testing methods for humanoid software. In Chap. 4, we discuss scenario-based testing cases as application cases, and in Chap. 5, we discuss conclusions and future research.

2 Related Studies

2.1 Simulation-Based Testing in Robotics

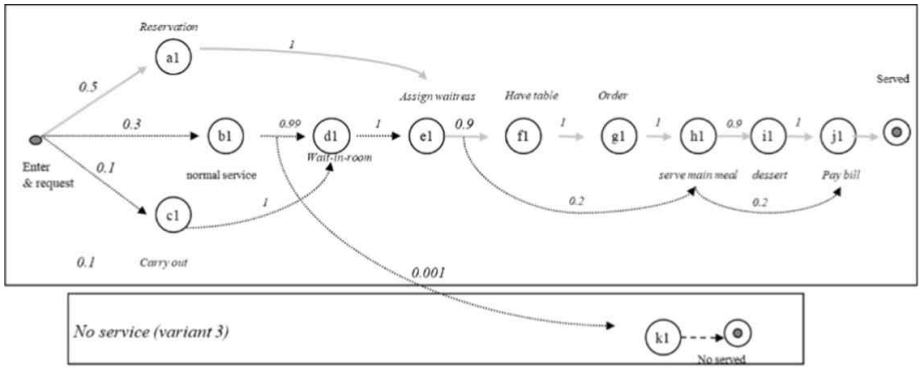
Simulators are widely used to reproduce and test hazardous or repetitive scenarios in virtual environments, such as those involving autonomous vehicles, drones, and humanoid robots. Afzal et al. evaluate simulators as cost-effective and practical in automated testing pipelines but point out that their predictive power in real-world environments is limited due to the Sim-to-Real gap [3]. Huck et al. propose a Monte Carlo-based safety assessment method, but there are difficulties in configuring simulation conditions. Recent research on humanoid robots has presented techniques for reducing the Sim-to-Real gap by integrating dynamic digital twins into reinforcement learning-based walking policies.

2.2 Scenario-Based Integration Testing for Object-Oriented Software Development

The adaptive use case methodology for improving testing efficiency integrates software design, development, and testing processes through a series of algorithmic transformations and is a process of generating test plans at the design stage [4]. The Use Case Action Matrix describes executable scenarios composed of 'action units', which indicate the execution path by numerically indicating specific action steps in the scenario. Based on the list extracted in Fig. 1, the Action Matrix is created, and the Graph is verified and improved. In addition, applying the concept of Musa's operational profile to optimize the scenario execution order improves test productivity by utilizing software test metrics and effectively provides test priorities at the design stage [5].

2.3 Scenario-Based Modeling in AI Software (Fig. 2)

Scenario-based modeling (SBM) is a methodology designed to address the complexity of reinforcement learning-based AI software validation [6]. This model integrates Markov decision processes (MDPs) and scenario-based software testing approaches to explicitly model state transitions, reward structures, and depreciation rates. In particular, it divides the state of the system into initial, intermediate, goal, and failure states, and systematically expresses possible actions and the probabilistic transitions and rewards that follow



Weighted value: 1 1 1 1 1 1 1 1 1 1 1 = 11 $\prod \frac{Si*Pi}{P}$

	a1	b1	c1	d1	e1	f1	g1	h1	i4	j1	k1	Weighted value amount	Total amount of probability of occurrence
Main path (Reserv.)	1				2	3	4	5	6	7		**7**	*0.5*1 * 0.9*1* 0.9*1=**0.405***
Variant 1 (normal)		1		2	3	4	5	6	7	8		**8**	*0.3*1*0.9*1 * 0.9* 1 =**0.243***
Variant 2 (Carryout)			1	2	3			4		5		**5**	*0.1*1 * 0.1*1 * 0.1*1=**0.001***
Variant 3 (no service)		1									2	2	*0.3*1 * 0.001*1 =**0.0003***

Fig. 1. Action Matrix on Scenario-based Integration Testing

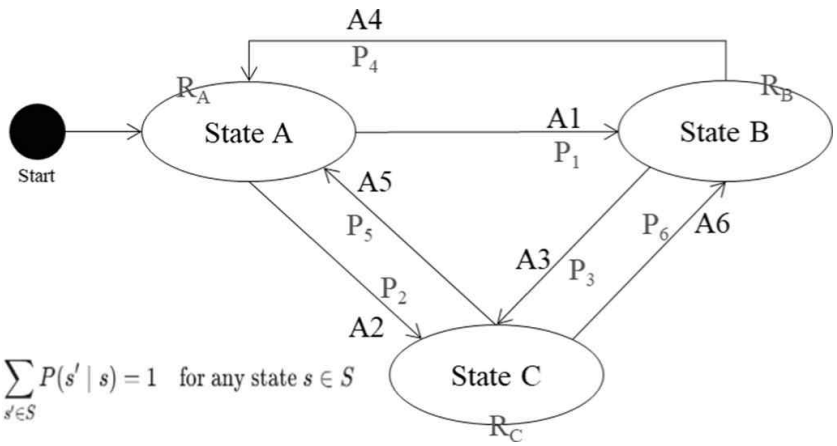


Fig. 2. Scenario Diagram

from each state. SBM defines all possible action paths of reinforcement learning agents as scenarios and reflects uncertainty and dynamic characteristics through modeling that includes probability and reward. This is connected to the concepts of state machine diagrams (SMDs), Markov decision processes (MDPs), and probabilistic scenario decision processes (SSDPs), allowing the system to be expressed visually and mathematically. These scenarios are organized in a tree form, and the priorities of each path can be set and visualized, and a test script is generated based on this, and a procedure is provided to verify the validity of the AI model in a simulation environment. Ultimately, SBM systematically detects unpredictable behaviors of AI software and contributes to increasing the efficiency and reliability of testing.

3 Humanoid Software Testing

3.1 Scenario-Based Modeling in AI Software (Fig. 3)

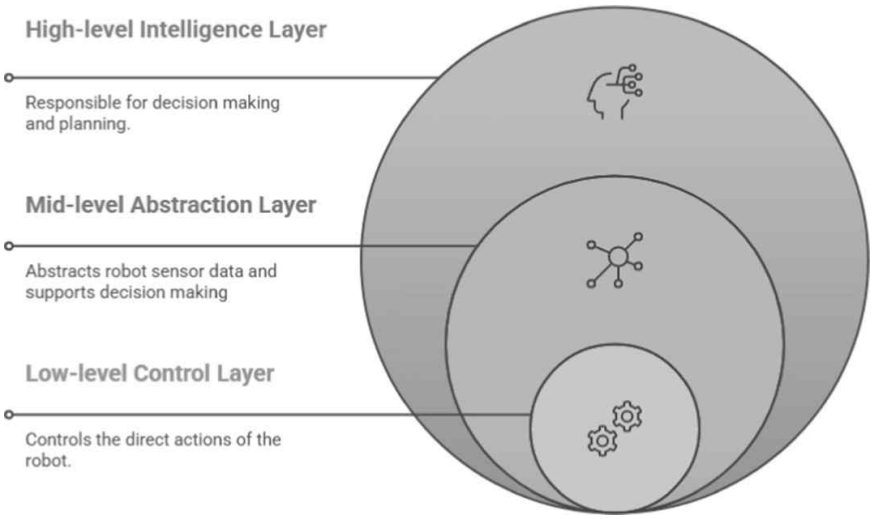


Fig. 3. Hierarchical Control Levels of Robotics System

Humanoid robots are robots with a similar body structure to humans and are designed to operate in various human-friendly environments. The software of these robots can be largely divided into three layers.

The lower control layer is directly responsible for the physical movement of the robot. This layer performs functions such as motor control, sensor data processing, and joint position control. A high level of real-time and precision is required for real-time and precise movement of the robot.

The middle abstraction layer combines commands from the lower control layers to create more complex movement patterns. For example, actions such as walking or moving arms are implemented in this layer. It also processes sensor data to perform environmental recognition of the robot. This layer may include a perception module, a planning module, and an action generation module.

The upper intelligence layer is responsible for the robot's cognitive, decision-making, and learning functions, and artificial intelligence (AI) models play a key role. AI technologies such as reinforcement learning and deep learning are utilized importantly in this layer. Interaction with users, overall task planning, and complex situation judgment are performed in the upper intelligence layer. Mode selection functions are an important part of the decision-making process that takes place in this higher intelligence layer. Humanoid robot software is characterized by complex interactions between these different layers and continuous feedback loops with the physical environment. In particular, the higher intelligence layer equipped with AI models greatly increases the autonomy and adaptability of the robot, but if unpredictable behavior or errors occur, it can have a negative impact on the safety and reliability of the entire system.

We focus on the mode selection requirements among the various functional requirements of humanoid software as shown in Table 1 and apply the Scenario-Based Modeling

Table 1. Humanoid Software Mode Selection Requirements.

Requirement ID	Requirement Description
FR-MS-001	The humanoid must support transitions between ‘Standby’, ‘Location Guidance’, and ‘Conversation’ modes based on user selection.
FR-MS-002	When a user selects a specific mode (e.g., ‘Location Guidance’, ‘Conversation’) from the home screen, the humanoid must immediately enter that mode.
FR-MS-003	Upon mode entry, the system must clearly indicate activation to the user, visually and/or audibly.
FR-MS-004	If a user selects ‘Back’ from the current mode, the humanoid must safely transition to the previous mode or ‘Standby Mode’.
FR-MS-005	No system errors or delays should occur during mode transitions; if they do, an appropriate error message must display, and the system should revert to standby.
FR-MS-006	Mode selection buttons must be designed to be intuitive and clearly recognizable.

(SBM) method as a methodology to analyze and verify them. Humanoid systems provide various services, and each service is operated in a specific ‘mode’. Since switching between these modes has a significant impact on the user experience and the overall usability of the system, the implementation of an accurate and efficient mode selection function is essential. We will use the SBM method to create scenarios and systematically verify them, focusing on core mode switching functions such as ‘location guidance’, ‘conversation’, and ‘back’. Each scenario comprehensively considers all situations that may occur in an actual usage environment by clearly defining the initial state, specific user behavior, expected target state, and possible failure state. In this way, through the scenario-based approach, we aim to contribute to the implementation of a highly reliable humanoid system by closely verifying whether the mode selection function of humanoid software operates accurately according to the user’s intention and whether the system functions stably.

4 A Case Study on Humanoid Software Validation

In this chapter, we focus on the mode selection requirement among the various functional requirements of the Humanoid Robot Nana’s system in Fig. 4 and analyze it and establish a scenario-based validation procedure. Humanoid provides various services through interaction with the user, and each service is defined as a unique ‘mode’. Therefore, the function of accurately and efficiently selecting and entering the desired mode by the user has a critical impact on the overall usability and user experience of the humanoid system. We create scenarios focusing on the three mode switching functions of the humanoid: ‘location guidance’, ‘conversation’, and ‘back’, and verify them. Through this, we verify whether the system accurately switches the mode according to the user’s

intention. We clearly define the initial state, user behavior, expected target state, and failure state for each mode selection scenario. This scenario-based approach contributes to systematically analyzing the functional requirements of a complex humanoid system and establishing a validation procedure by considering all situations that can occur in an actual usage environment.

Fig. 4. Humanoid Nana

Step 1. Creation of Humanoid Software Scenario for Testing. In this step, a humanoid software validation scenario is created. The functional target environment, action unit, initial target state, and failure state of the humanoid robot to be verified are defined. These elements are the core foundation for creating an effective scenario. Information such as the environment, state, and action of the humanoid software is extracted from the functional requirements mentioned in Chap. 3. Fig. 5 shows the state table, action unit table, and scenario diagram created based on the information extracted from the requirements. The scenario-diagram diagrammatically expresses the transition process from the initial state ($S_{initial}$) to each state. Each transition branch describes the probability of the corresponding action occurring and the reward value. In this paper, the reward value is set to 1 for the convenience of calculation.

Tables 2 and 3 show the description of the state and action list extracted from Fig. 4. Organizing it in a table like this can improve the readability of the Scenario Diagram.

Step 2. Refine Scenarios with Decision Table. In this step, the initial scenario diagram generated in STEP 1 is refined through the scenario decision table as shown in Fig. 6. The decision table systematically checks the transition rules and conditions between each state to identify all possible cases and prevent unnecessary duplication. This reduces the error rate of the scenario diagram and increases reliability. The decision table defines what action the robot should take and what next state it should transition to depending on specific conditions (input, internal state of the robot, etc.). This process helps the scenario accurately reflect the complex logic of the actual robot system and consider all exceptional situations that may occur during the mode selection process. The refined scenario plays an essential role in verifying that the mode selection function of the

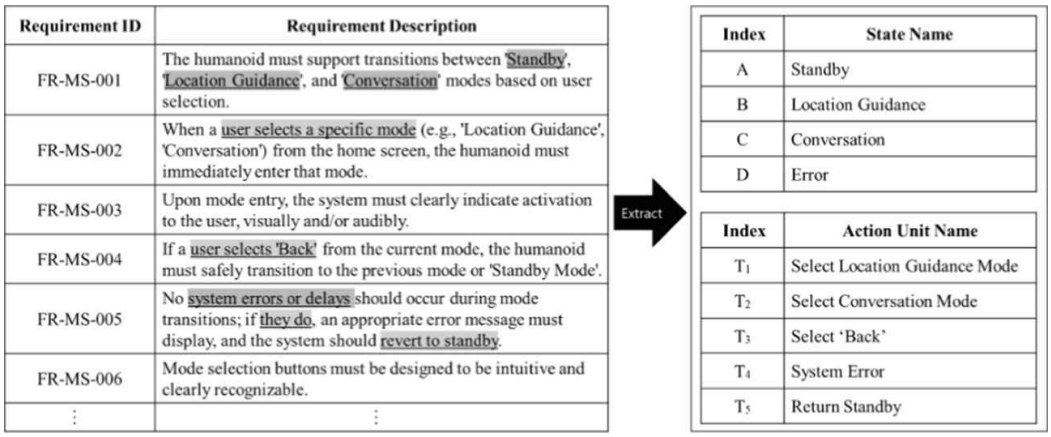


Fig. 5. State and Action Unit Extraction from Requirements

Table 2. State Description of Humanoid Software Mode Selection

State	Definition
Standby	A default idle state, awaiting user input
Location Guidance	An activate state providing location-related information
Conversation	An activate state engaging in conversation with the user
Error	System error encountered, displaying an error message.

Table 3. Action Description of Humanoid Software Mode Selection

Action	Definition
Select Location Guidance Button	User selects a 'location guidance button', initiating immediate transition.
Select Conversation Button	User selects a 'Conversation button', initiating immediate transition.
Select 'Back'	User selects 'back' from an active mode, transitioning to Standby mode.
System Error	System errors or delays during mode transitions
Return Standby	Represents the system's automatic recovery from the Error State, transitioning back to Standby Mode after displaying an error message.

humanoid robot operates as expected and that there are no potential defects or abnormal behaviors.

Step 3. Convert Scenario Diagram into Scenario-Based Tree. In this step, the refined scenarios from STEP 2 are converted into a tree structure that includes a common starting point and branching points. Each node of this tree represents a 'state' of the humanoid