

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

A Non-Functional Quality Framework for Blockchain Software: ISO/IEC 25010 Revisited	441
<i>Abeer S. Al-Humaimedy, Salma HabibAllah, and Reema Alhamadi</i>	
Remaining Agile with Hybrid Work Models	459
<i>Mierhamza Rahimi and Catherine Stringfellow</i>	
The Role of Provenance Modeling in Tracing and Reproducing Explainable AI Pipelines	470
<i>Zerui Wang and Yan Liu</i>	
Best Practice of Scenario-Based Modeling for Humanoid Software Validation	486
<i>Sangho Lee, Janghwan Kim, and R. Young Chul Kim</i>	
Cumulative Effort Estimation in an Evolving Legacy System	498
<i>Lamees Alhazzaa, Anneliese Amschler Andrews, and Aiman Gannous</i>	
The Use of ChatGPT to Teach Coding Inhibits Software Development or Supplements Coding Education	512
<i>José A. Vega, Juan B. Varela, and Ángel Ojeda</i>	
Poster Research Papers	
Branch-Commit-Validate: A Git-Inspired Workflow for Autonomous Red-Team Agents	525
<i>Chinmay Shringi, Alon Hillel-Tuch, and Sariya Rizwan</i>	
Author Index	531



Best Practice of Scenario-Based Modeling for Humanoid Software Validation

Sangho Lee¹ , Janghwan Kim² , and R. Young Chul Kim²

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.

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

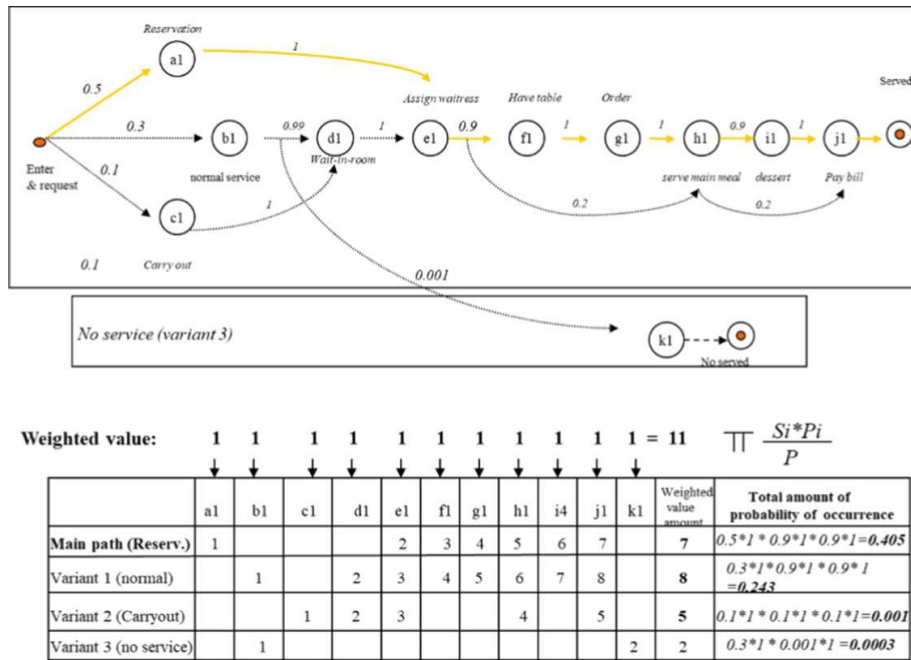


Fig. 1. Action Matrix on Scenario-based Integration Testing

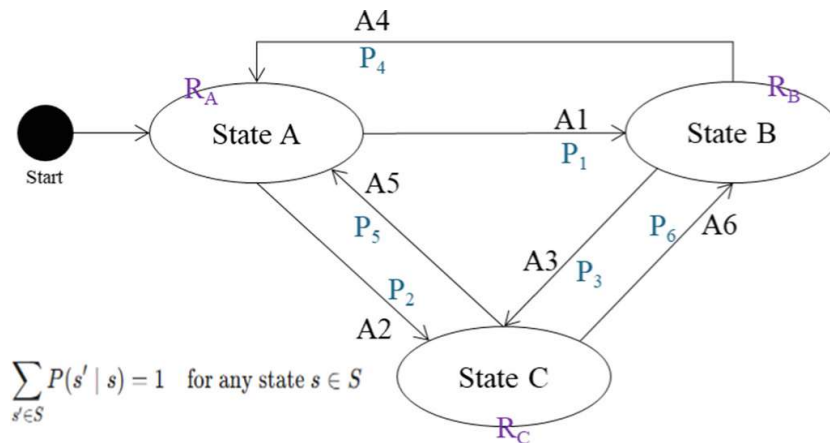


Fig. 2. Scenario Diagram

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)

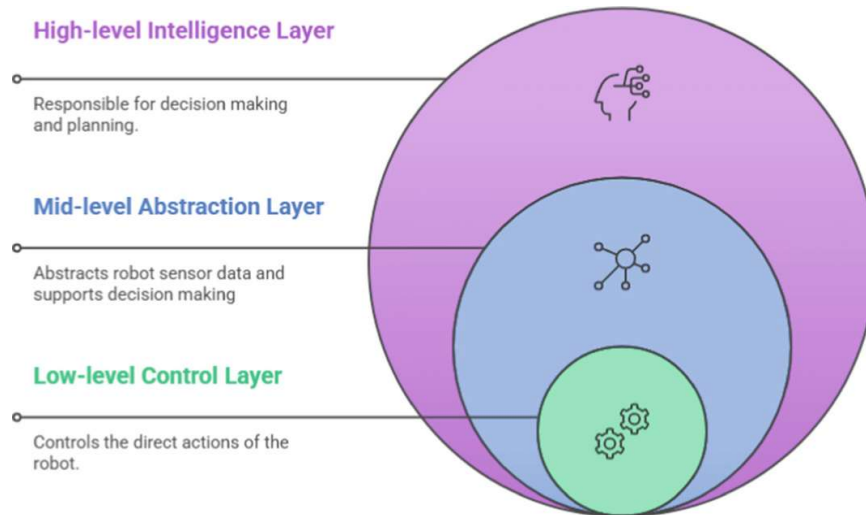


Fig. 3. Hierarchical Control Levels of Robotics System

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.

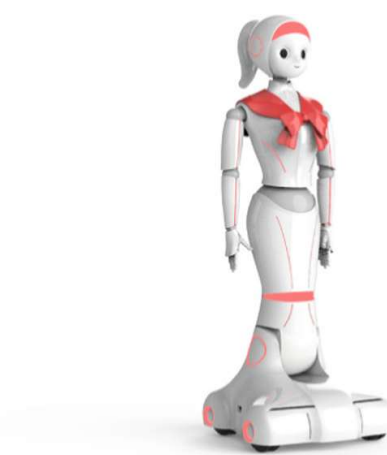
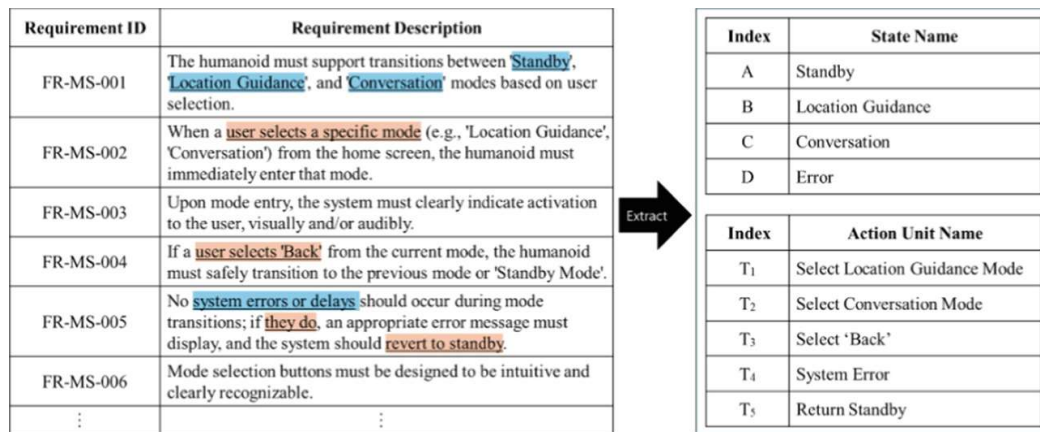


Fig. 4. Humanoid Nana

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

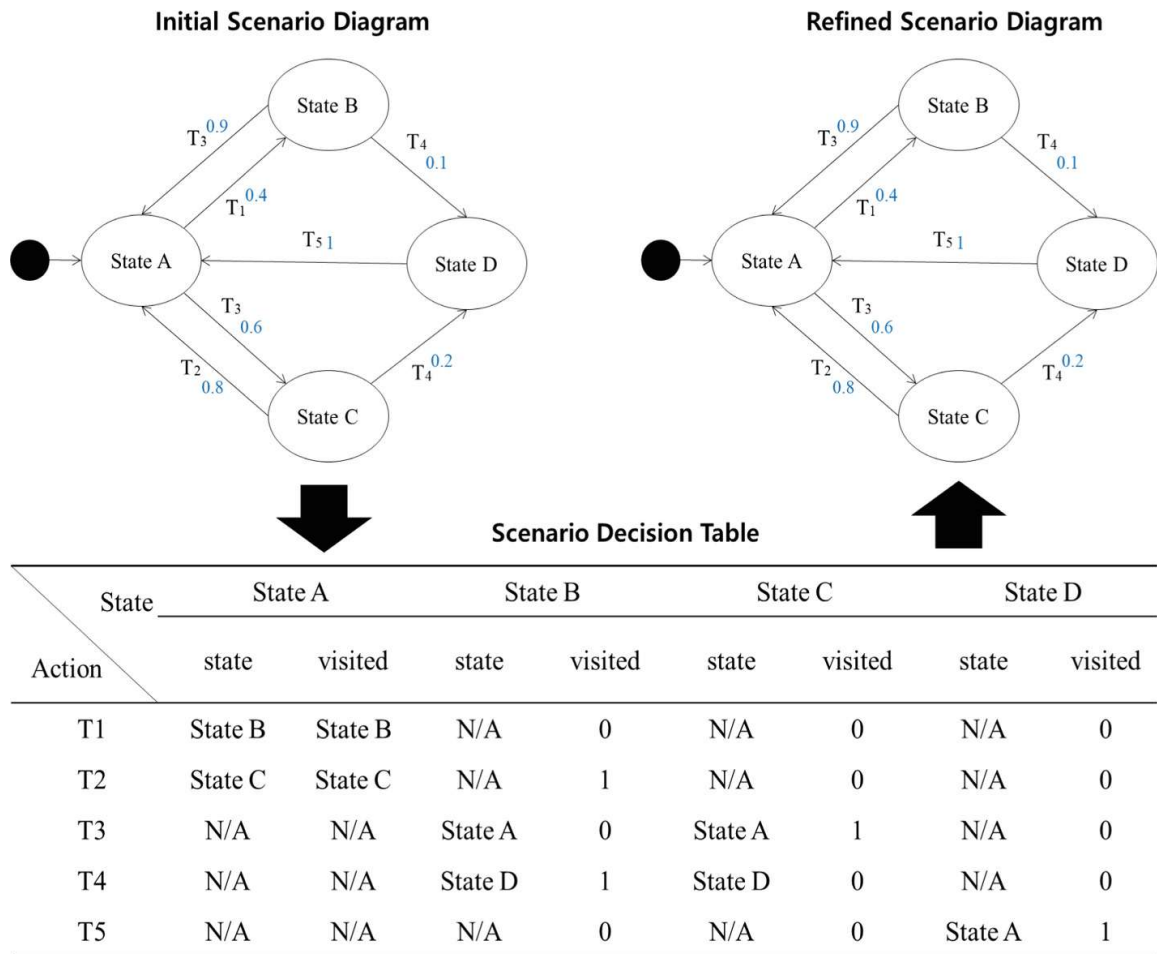
State	Definition
Standby	A default idle state, awaiting user input
Location Guidance	An activate state providing location-related information
Conversation	An activate state engaging in conversation with the user
Error	System error encountered, displaying an error message.

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

**Fig. 6.** Refining Scenario Diagram Process

software that composes the scenario, and the edges represent ‘actions’ between states. As shown in Fig. 6, the probability value based on the frequency of occurrence of ‘actions’ is indicated on the edges, thereby specifying the probability of the agent’s action selection in an uncertain environment.

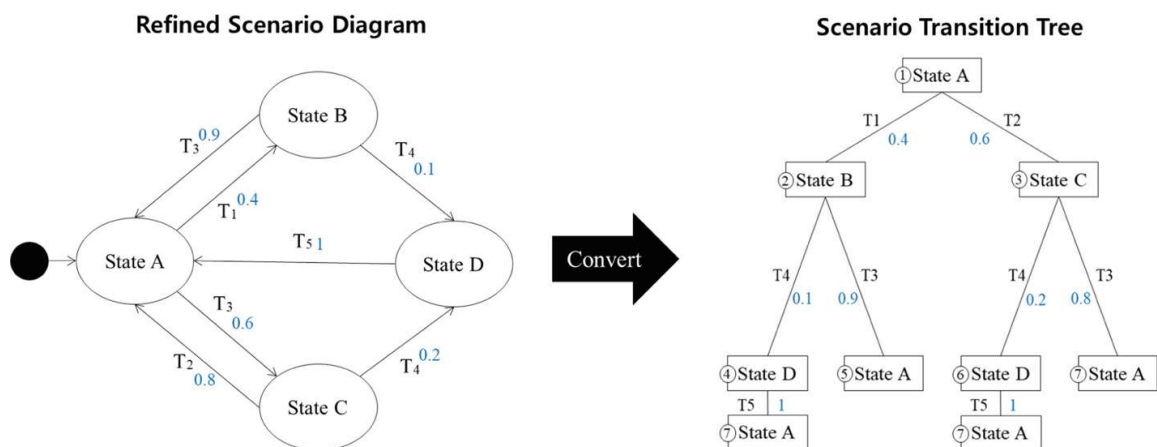
**Fig. 7.** Convert Scenario diagram into Scenario Transition Tree

Figure 7 shows the final form of this scenario transition tree. The refined scenario is converted into a tree structure through the decision table, and the probability value that occurs on the edge between each state is described to complete the scenario transition tree as in Fig. 8. This tree structure clearly and visually shows all possible paths that occur during the mode selection process of the humanoid robot and provides a basis for more quantitative analysis of the robustness and reliability of the system by considering the probabilistic characteristics of each path.

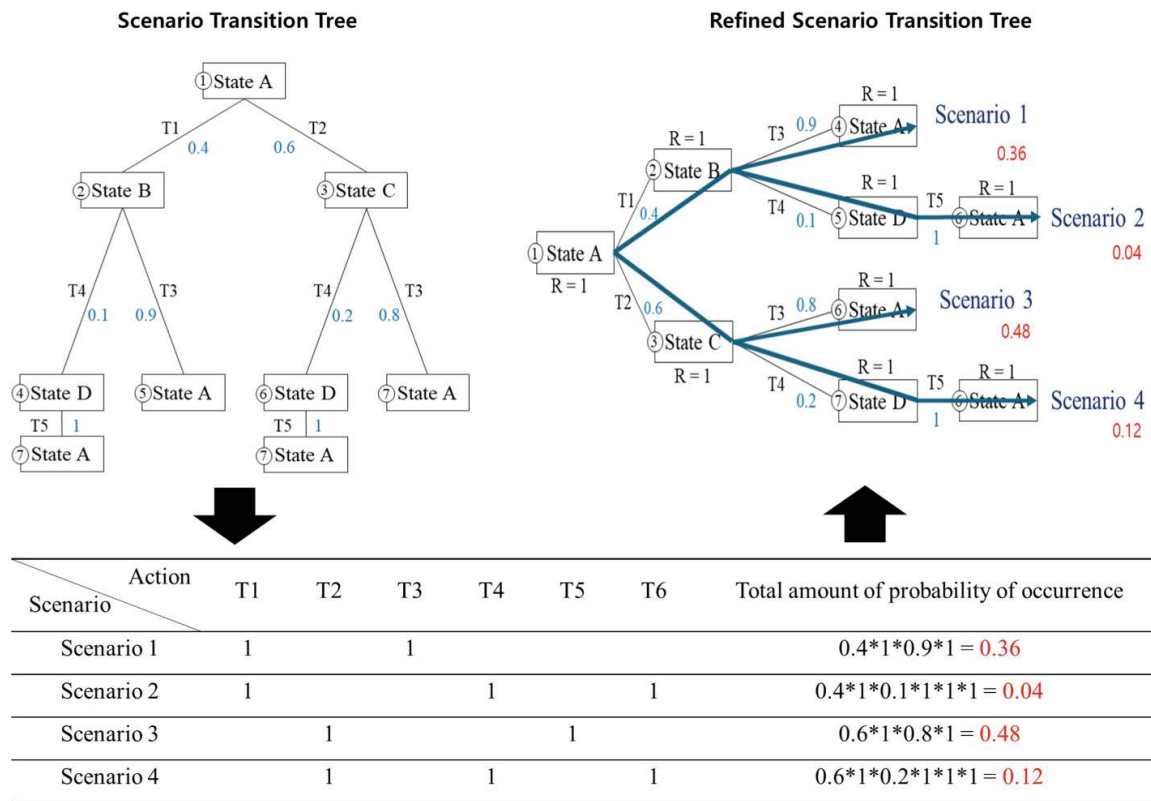


Fig. 8. Refining Scenario Transition Tree with Occurrence Probability

Step 4. Generate Scenario-Based Test Case. In this step, scenario-based test cases are generated based on the scenario transition tree generated in STEP 3. Each test case defines a specific motion sequence of the robot along the path of the tree, which is used to verify that the mode selection function of the humanoid system operates correctly in various situations. As shown in Table 4, this test case consists of Test Case ID, Initial State, Event Index, Action, and Next State. Each field clearly defines the flow of a specific test scenario and provides detailed information to verify that the robot responds as expected to a given input. This process enables systematic and comprehensive validation of the mode transition function of the humanoid robot and contributes to identifying and improving potential errors in advance in an actual operating environment.

Step 5. Generate a Behavior Tree for Performing Test Cases. In this step, a behavior tree is generated to execute the script according to each situational scenario. A behavior tree is a mathematical model of planning and execution used in computer science,

Table 4. Scenario-based Test Case

	Initial State	Action	State	Action	State	Action	Final State
TC 1	State A	T1	State B	T3	–	–	State A
TC 2	State A	T1	State B	T4	State D	T5	State A
TC 3	State A	T2	State C	T3	–	–	State A
TC 4	State A	T2	State C	T4	State D	T5	State A

robotics, control systems, and video games, and is used to effectively structure and manage complex robot behavior. The behavior of the humanoid robot operates through this behavior tree, which helps the robot to flexibly respond to various environments and user inputs. In Fig. 9, the behavior tree is generated based on the state list and action units extracted in STEP 1, as well as the scenario-based test cases generated in STEP 4. The behavior tree hierarchically defines the tasks that the robot must perform, and expresses complex decision-making logic through conditional nodes, sequence nodes, and selector nodes. The behavior tree generated through this process controls the humanoid robot to accurately and efficiently perform the scenarios specified in the test cases and makes the system's behavior predictable and stable.

Step 6. Validate Humanoid Software with Scenario-Based Behavior Tree. In this step, the mode selection function of the humanoid software is validated using the behavior tree generated in STEP 5. This step is performed by executing the scenario defined in the behavior tree in an actual robot or simulation environment. The validation process starts from the initial state specified in each test case and checks whether the robot transitions to the next state exactly according to the defined action sequence and reaches the final target state. In particular, the stability and reliability of the system are evaluated by closely monitoring cases where the robot transitions to an unintended state or enters an error state. This validation process focuses on verifying that the mode switching function of the robot operates exactly according to the user's intention and responds appropriately in exceptional situations. Additionally, the accuracy of sensor data processing and operation execution in an actual environment is evaluated to assess the system's performance comprehensively. This proves that the humanoid robot provides a stable and reliable mode selection function.

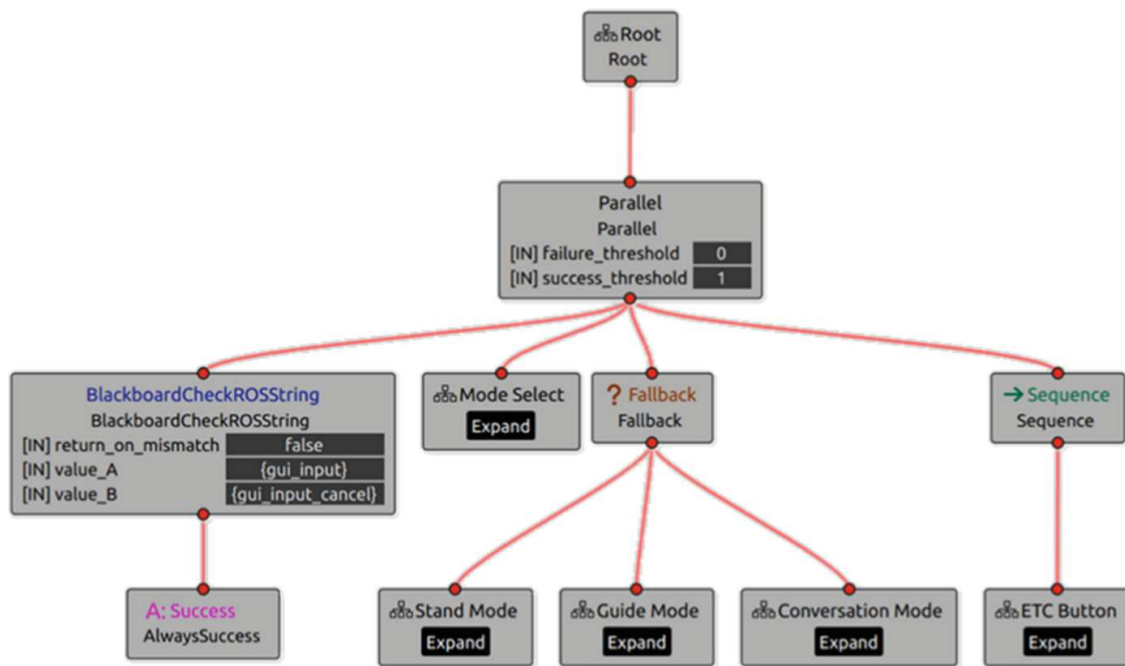


Fig. 9. The Simple case of Behavior Tree for Humanoid Software Validation

5 Conclusion

This paper is a case study on a systematic validation procedure to secure reliability and stability within the complex software system of a humanoid robot. We focus on the mode selection function in the upper intelligence layer where the AI model plays a key role. We demonstrate the efficiency of the validation method for AI-based software operating in the upper intelligence layer through relatively intuitive cases such as ‘location guidance’, ‘conversation’, and ‘go back’. In future studies, we will further extend the scenario-based validation procedure proposed in this study to apply the currently fixed reward value to various reward systems in real environments, thereby performing more realistic scenario validation. Additionally, we will conduct validation directly on an actual humanoid robot, beyond simulation, to address physical interaction and real-time control issues. We will also develop and apply a scenario-based validation procedure for core robot functions (such as path planning and object recognition) other than mode selection, to enhance the reliability of AI software.

Acknowledgments. This research was conducted with the support of the Korea Creative Content Agency (Project Name: Artificial Intelligence-Based Interactive Multimodal Interactive Storytelling 3D Scene Authoring Technology Development, Project Number: RS-2023-00227917, Contribution Rate: 100%) and the Korea Research Foundation’s four, Brain Korea 21 (Project Name: Ultra-Distributed Autonomous Computing Service Technology Research Team, Project Number: 202003520005).

Disclosure of Interests The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Li, Y., Yang, Z., Guo, Y., Chen, X.: Humanoid: a deep learning-based approach to automated black-box android app testing. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 1070–1073. IEEE (2019)
2. Gao, J., Tao, C., Jie, D., Lu, S.: Invited Paper: What is AI Software Testing? and Why. In: 2019 IEEE international conference on service-oriented system engineering (SOSE), San Francisco, CA, USA, pp. 27–2709 (2019). <https://doi.org/10.1109/SOSE.2019.00015>
3. Afzal, A., Katz, D.S., Le Goues, C., Timperley, C.S.: Simulation for robotics test automation: developer perspectives. In: 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST), pp. 263–274. IEEE (2021)
4. Kim, Y., Carlson, C.R.: Scenario-based integration testing for object-oriented software development. In: Proceedings Eighth Asian Test Symposium (ATS'99), pp. 283–288 (1999) IEEE.CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016)
5. Musa, J.D.: Operational profiles in software-reliability engineering. IEEE Softw. **10**(2), 14–32 (1993)
6. Kim, J., Kim, K., Son, H., Kim, R.: Scenario-based modeling in AI software validation. In: The 20th Asia Pacific International Conference on Information Science and Technology, KSII, Phuket, Thailand (2025)