2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

An ISO/IEC 25010-Based Evaluation Methodology for Designing Digital Mock-Up Toward Nuclear Decommissioning

Fumiaki Abe[1], Kuniaki Kawabata[1], Wataru Sato[2] and Tomoki Sakaue[2]

***Abstract*— This paper describes an ISO/IEC 25010-based evaluation methodology for designing Digital Mock-Up (DMU) toward nuclear decommissioning. Tokyo Electric Power Company Holdings, Inc. (TEPCO) is planning to develop a DMU to facilitate safe and reliable decommissioning operations using remote-control robots since the conditions inside the reactor buildings remain many unknown unknowns. To effectively implement the DMU, it is essential to carefully select available software technologies, install them, and adopt a software system integration framework to organize the DMU. Therefore, appropriate evaluations must be conducted. In this paper, we proposed a quantitative method for evaluating the functionality and systemization of the DMU based on a software quality requirements and evaluation model ISO/IEC 25010. Also, we reported a case study in which the proposed evaluation methodology was applied to a fundamental DMU developed based on a typical use case.**

I. INTRODUCTION

The environmental conditions inside the reactor buildings of the Fukushima Daiichi Nuclear Power Station (FDNPS) remain harsh due to high radiation dose rates, and it makes decommissioning tasks difficult. To reduce the risks to human workers, remote-control devices such as robots have been deployed to perform decommissioning works [1]. Remote operations of the robots are conducted by human operators, who monitor camera images and command through a Human Machine Interface (HMI). In general, remote operations for the decommissioning of FDNPS are challenging and time-consuming. This is primarily due to the presence of ”unknown unknowns” at the working space—hazards and issues that cannot be clearly identified in advance. To manage such uncertainties, robust safety measures and meticulous remote operation procedures are essential to ensure both the safety of human workers and the reliability of robotic systems.

Because of this background, TEPCO is planning to develop a DMU to facilitate safe and reliable decommissioning operations using remote-control robots. The development team is assuming that the DMU can be utilized for designing robots and equipment, training robot operators, rehearsing planned operations, and supporting on-site task executions by providing digitalized information of the workspace to the operators. The DMU is a system consisting of various software, hardware including robots, database and middleware for interoperation among these components in order to achieve the aforementioned functions (Fig.1). Since FDNPS decommissioning work is regarded as a long-term project, and the working environments and tasks are expected to be changed depending on the operation progresses, the DMU must install appropriate functions to be utilized in typical operational scenarios. To achieve this, it is important to assess the performance of the DMU through quantitative evaluation. This enables the DMU to mature by facilitating the selection and procurement of appropriate features from the market and by incorporating feedback from stakeholders such as operators and software developers.

ISO/IEC 25010 is an international standard model for evaluating software quality requirements and evaluation [2]. This standard provides a framework for assessing both functional and non-functional requirements of software systems. However, practical evaluation metrics for each requirement are not defined and explained in ISO/IEC 25010.

Therefore, we propose a quantitative evaluation methodology for the DMU based on ISO/IEC 25010 by defining concrete evaluation metrics. Furthermore, we present a case study in which the proposed evaluation methodology is applied to a fundamental DMU developed based on a typical use case.

The paper is organized as follows: Section 2 outlines the requirements of the DMU system that can be utilized in decommissioning work. Section 3 reviews related literature on software evaluation. Section 4 explains the principles of the proposed evaluation methodology. Section 5 presents a case study to demonstrate the application of the methodology. Fi-

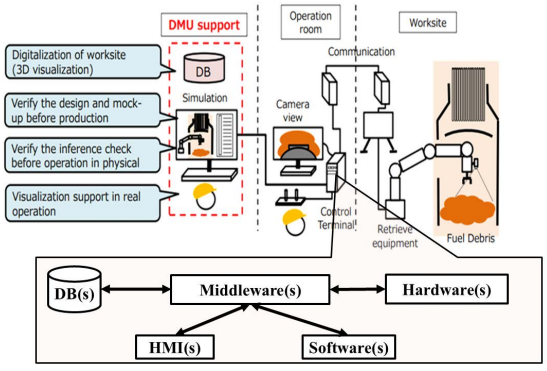

Fig. 1: Concept diagram of the DMU.

nally, Section 6 summarizes the paper.

II. Requirements of DMU system

In this section, the requirements of the DMU system are described. In order to realize that the DMU provides functions such as designing robots and equipment, training robot operators, rehearsing planned operations, and supporting remote task executions by providing digitalized information of the workspace to the operators. We consider that the DMU has to meet following requirements:

(a) Real-time responsibility: enhancing safety and operational efficiency during real task execution, while reducing the operator's workload,
(b) Usability: intuitively and effectively improving the overall operational experience,
(c) Reliability: consistently maintaining the expected functionality in accordance with user intentions,
(d) Security: safeguarding data and assets, as well as ensuring compliance with legal regulations,
(e) Maintainability: responding to system failures and performing updates rapidly and effectively,
(f) Flexibility: rapidly and seamlessly adapting to continuously changing environments and requirements,
(g) Safety: mitigating the risk that unauthorized access or operator error during remote operation could cause serious damage.

To satisfy the requirement (a), the DMU system must incorporate low-latency communication protocols, high-throughput capabilities for sensor data processing, and calculation of control input to the robot that responds in real time. To satisfy the requirement (b), the DMU system must incorporate intuitive user interfaces, consistent design, understandable user manuals, feedback and error message provision functions. To satisfy the requirement (c), the DMU system must incorporate mechanisms for reducing failure rates, ensuring redundancy, and enabling recovery from faults. To satisfy the requirement (d), the DMU system must incorporate data protection, access control, authentication, authorization, monitoring, and penetration prevention functions. To satisfy the requirement (e), the DMU system must incorporate functionalities for automated testing and refactoring to support code modification, extension, bug fixes, and feature enhancements, as well as mechanisms for managing software dependencies. To satisfy the requirement (f), the DMU system must incorporate cross-platform compatibility, scalability, and standard interfaces. To satisfy the requirement (g), the DMU system must incorporate risk detection and alert mechanisms, as well as fail-safe functionalities.

III. Literature review for software evaluation methodologies

There are several evaluation methodologies for assessing software applications. Jan et al. proposed a software maturity model to evaluate digital twin applications [3]. The evaluation model consisted of 7 categories and included 31 ranked characteristics. Users could evaluate them by the predefined metrics for each characteristic. However, this maturity model primarily emphasized functional aspects, while providing limited consideration on flexibility, extensibility, and security.

Robotic middleware is widely adopted in modern robotic systems, as it provides a communication infrastructure that integrates various software modules and libraries [4][5]. Tsardoulias et al. reviewed several robotics frameworks and middlewares, providing a comprehensive comparison based on supported operating systems, programming languages, and system architecture [6]. However, the review was conducted from a subjective perspective and lacked consideration on maintainability, flexibility, and safety.

Rohini et al. proposed a set of quantitative evaluation metrics based on ISO/IEC 25010 for assessing IoT applications and presented a case study demonstrating the use of these metrics[7]. These evaluation metrics were based on the ISO/IEC 25010 standard, which originally consisted of 8 evaluation categories (referred as characteristics) — namely, Functional Suitability, Performance Efficiency, Compatibility, Interaction Capability, Reliability, Security, Maintainability, and Flexibility — and 31 associated subcategories (referred as sub-characteristics), as published in 2011 [2]. In the 2023 revision, ISO/IEC 25010 was revised to include a new characteristic related to safety and its associated sub-characteristics, along with the renaming and reorganization of some existing elements. The revised standard now comprises 9 characteristics and 40 sub-characteristics. Since the DMU is required to be closely integrated with robots that perform decommissioning tasks, safety is the most critical perspective.

In this study, we propose a novel evaluation methodology for the DMU based on the 2023 revision of ISO/IEC 25010.

IV. Proposed Evaluation Metrics

In this section, we describe the evaluation methodology for the DMU based on ISO/IEC 25010. The proposed metrics enables a simple yet comprehensive evaluation of DMU functionalities and performance. The evaluation value of each characteristic is calculated as the simple average of the evaluation values of its sub-characteristics, with each sub-characteristic

normalized to the range [0, 1].

$$E_{c_i} = \frac{1}{n_{c_i}} \sum_{j=1}^{n_{c_i}} X_{s_j}, \quad X_{s_j} \in [0,1] \tag{1}$$

Where, n_{c_i} is the number of sub-characteristics in each characteristic, and E_{s_j} is evaluation value of the j-th sub-characteristic. Evaluation values of sub-characteristics are defined based on quantifiability. For items evaluated directly from DMU functions or features, X_{sj} is set to 0.0 or 1.0. For items evaluated indirectly, we apply $X_{s_j} = A_j/B_j$, where A_j is the measured value or count corresponding to the sub-characteristic j-th that satisfies the required conditions, and B_j is the total possible value or count. Note that the evaluation value for Resource utilization is calculated as $E_{s_j} = \frac{1}{n}\sum_{k=1}^{n}(A_k+B_k+C_k)/3$, where n is the total number of computers in the system, and A_k, B_k, and C_k are the CPU, RAM, and GPU usage of the computer k-th, respectively. Functional appropriateness, availability, and installability are excluded from the scope, as they are similar to functional completeness, faultlessness, and adaptability, respectively. Table I and Table II show the lists for the evaluation metrics.

V. Fundamental DMU evaluation case study

The primary aim of this section is to present a case study that verifies the feasibility of quantitative evaluation using the proposed methodology. In this case study, we evaluate a fundamental DMU developed based on the functional requirements identified through a possible use case in FDNPS decommissioning task.

TABLE I: List of evaluation metrics (1/2).

Characteristics	Sub-characteristics	Description	Evaluation metrics
Functional suitability	Functional completeness	Evaluates the number of functions that realize the user's intended purpose compared to the number of functions defined as requirements.	$X_1=A_1/B_1$ A_1: The number of functions that realize the user's intended purpose, B_1: The number of functions defined as requirements.
	Functional correctness	Evaluates the number of functions that satisfy the required level of precision compared to the number of functions included in the evaluation scope.	$X_2=A_2/B_2$ A_2: The number of functions that satisfy the required level of precision, B_2: The number of functions defined included in the evaluation scope.
	Functional appropriateness	Since this sub-characteristics is similar to Functional completeness, it will be out of scope.	-
Performance efficiency	Time behavior	Evaluates whether the system satisfies the required level of respose time and throughput.	X_3 = {0: Not satisfied or 1: Satisfied}
	Resource utilization	Evaluates how much computational resources (CPU, RAM, GPU) are used by the system.	$X_4 = 1/n\sum_{k=1}^{n}(A_k + B_k + C_k)/3$ n: The number of computers utilized across the entire system, A_k: CPU usage of the computer k-th, B_k: RAM usage of the computer k-th, C_k: GPU usage of the computer k-th.
	Capacity	Evaluates the number of functions that satisfy the required level of amount of data compared to the number of functions included in the evaluation scope.	$X_5=A_5/B_5$ A_5: The number of functions that satisfy the required level of amount of data, B_5: The number of functions included in the evaluation scope.
Compatibility	Co-existence	Evaluates the number of hardware devices that can interconnected to the system compared to the number of hardware devices included in the evaluation scope.	$X_6=A_6/B_6$ A_6: The number of hardware devices that can interconnected to the system, B_6: The number of hardware devices included in the evaluation scope.
	Interoperability	Evaluates the number of middlewares that can interconnected to the system compared to the number of middlewares included in the evaluation scope.	$X_7=A_7/B_7$ A_7: The number of middlewares that can interconnected to the system, B_7: The number of middlewares includedn in the evaluation scope.
Interaction capability	Appropriateness recognizability	Evaluates the number of functions that judged by users as appropriate ones compared to the number of functions included in the evaluation scope.	$X_8=A_8/B_8$ A_8: The number of functions that judged by the users as appropriate ones, B_8: The number of functions includedn in the evaluation scope.
	Learnability	Evaluates the number of functions that users understand how to use system within specified time compared to the number of functions included in the evaluation scope.	$X_9=A_9/B_9$ A_9: The number of functions thatusers understand how to use system within specified time, B_9: The number of functions includedn in the evaluation scope.
	Operability	Evaluates whether the system has the functions that provide users to control and operate easily.	X_{10} = {0: Not implemented or 1: Implemented}
	User error protection	Evaluates whether the system has the functions that prevent operator from error operation.	X_{11} = {0: Not implemented or 1: Implemented}
	User engagement	Evaluates the number of user interfaces the present functions and information in an inviting and motivating manner, encouraging continued interaction compared to the number of user interfaces included in the evaluation scope.	$X_{12}=A_{12}/B_{12}$ A_{12}: The number of user interfaces that present functions and information in an inviting and motivating manner, encouraging continued interaction, B_{12}: The number of user interfaces included in the evaluation scope.
	Inclusibity	Evaluates whether the system can be operated by users who have various background.	X_{13} = {0: Not implemented or 1: Implemented}
	User assistance	Evaluates whether the system can operate by users within the company/organization.	X_{14} = {0: Not implemented or 1: Implemented}
	Self-descriptiveness	Evaluates whether the system provides instructions with users without interact with any other resources.	X_{15} = {0: Not implemented or 1: Implemented}

TABLE II: List of evaluation metrics (2/2).

Characteristics	Sub-characteristics	Description	Evaluation metrics
Reliability	Faultlessness	Evaluates the number of functions that perform intended behavior without breakdown compared to the number of functions included in the evaluation scope.	$X_{16}=A_{16}/B_{16}$ A_{16}: The number of functions that perform intended behavior without breakdown, B_{16}: The number of user interfaces included in the evaluation scope.
	Availability	Since this sub-characteristics is similar to Faultlessness, it will be out of scope.	-
	Fault tolerance	Evaluates the number of functions that perform intended behavior even if there is any software failure compared to the number of functions included in the evaluation scope.	$X_{17}=A_{17}/B_{17}$ A_{17}: The number of functions that perform intended behavior even if there is any software failure, B_{17}: The number of user interfaces included in the evaluation scope.
	Recoverability	Evaluates whether the system has the function that recover the connection to the robot or data which was affected by the system directly.	X_{18} = {0: Not implemented or 1: Implemented}
Security	Confidentiality	Evaluates whether the system can be accessed only by the authorized users.	X_{19} = {0: Not implemented or 1: Implemented}
	Integrity	Evaluates whether the system can ensure that the state of its system and data are protected from unauthorized modification or deletion either by malicious action or computer error.	X_{20} = {0: Not implemented or 1: Implemented}
	Non-repudation	Evaluates whether the system can be proven to have taken place so that the evants or actions cannot be repudiated later.	X_{21} = {0: Not implemented or 1: Implemented}
	Accountability	Evaluates whether actions of an entity can be traced uniquely to the entity.	X_{22} = {0: Not implemented or 1: Implemented}
	Authenticity	Evaluates whether the identity of a subject or resource can be proved to be the one claimed.	X_{23} = {0: Not implemented or 1: Implemented}
	Resistance	Evaluates whether the system sustains operations while under attack from a malicious actor.	X_{24} = {0: Not implemented or 1: Implemented}
Maintainability	Modularity	Evaluates whether the system is composed of discrete components such that a change to one component has minimal impact on other components.	X_{25} = {0: Not implemented or 1: Implemented}
	Resusability	Evaluates whether the system can be used as an asset in more than one system, or in building other assets.	X_{26} = {0: Not implemented or 1: Implemented}
	Analyzability	Evaluates whether the system can be possible to assess the impact on a system of an intended change to one or more of its parts, to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified.	X_{27} = {0: Not implemented or 1: Implemented}
	Modifiabilty	Evaluates whether the system can be effectively and efficiently modified without introducing defects or degrading existing product quality.	X_{28} = {0: Not implemented or 1: Implemented}
	Testability	Evaluates whether the system has the function that verify the quality of components automatically.	X_{29} = {0: Not implemented or 1: Implemented}
Flexibility	Adaptablity	Evaluates the number of OSs that can work with the system compared to the number of OSs included in the evaluation scope.	$X_{30}=A_{30}/B_{30}$ A_{30}: The number of OSs that can work with the system, B_{30}: number of OSs included in the evaluation scope.
	Scalability	Evaluates the number of the meeting requirements regarding response time and through response time and throughput compared to the maximum number of robots.	$X_{31}=A_{31}/B_{31}$ A_{31}: The number of the meeting requirements regarding response time and through response time and throughput, B_{31}: The maximum number of robots.
	Installablity	Since this sub-characteristics is similar to Adaptability, it will be out of scope.	-
	Replaceablity	Evaluates the number of functions capable of handling system version changes compared to the number of functions included in the evaluation scope.	$X_{32}=A_{32}/B_{32}$ A_{32}: The number of functions that perform intended behavior without breakdown, B_{32}: The number of user interfaces included in the evaluation scope.
Safety	Operational constraint	Evaluates whether the system has the function that constraints its operation to within safe parameters or states when encountering operational hazard.	X_{33} = {0: Not implemented or 1: Implemented}
	Risk identification	Evaluates whether the system has the function that can identify a course of events or operations that can expose life, property or environment to unacceptable risk.	X_{34} = {0: Not implemented or 1: Implemented}
	Fail safe	Evaluates whether the system has the function that can automatically place the system in a safe operating mode.	X_{35} = {0: Not implemented or 1: Implemented}
	Hazard warning	Evaluates whether the system has the function that provide warnings of unacceptable risks to operations.	X_{36} = {0: Not implemented or 1: Implemented}
	Safe integration	Evaluates whether the system has the function that maintain safety during and after integration with one or more components.	X_{37} = {0: Not implemented or 1: Implemented}

A. A use case and a scenario

Here, we selected a use case focusing on the inspection around the Hydraulic Control Unit (HCU) by a remote-control robot. HCU is equipment that supplies and controls high-pressure water to the Control Rod Drive beneath the reactor. At FDNPS Unit 3, TEPCO has identified the HCU as a high-dose source [8]. Currently, an HCU-inclusive water sampling task is under consideration based on the results of pre-investigations, and a plan is underway to reduce the air dose rate around the HCU. In this task, understanding the conditions around the HCU is essential for ensuring the safety of future operations. Here, we designed a scenario simulating the HCU inspection task, in which an operator remotely controlled a mobile robot, navigated it through the environment to inspect a target object, and then returned it to the start position. The workspace was a corridor of 9.0 m long and 1.8 m wide, with a target object placed at the end of the corridor (Fig.2).

B. Experimental system contiguration

The system and software configurations are illustrated in Fig.3. We used a TurtleBot3 Waffle Pi [9] with a camera, and a laser distance sensor. We also used a remote control PC with the following specifications: Intel Core Ultra 9, 32GB RAM and NVIDIA GeForce RTX 4070 GPU, and Ubuntu 22.04 as the operating system. Both TurtleBot3 and the remote control PC were configured with ROS noetic. To realize the scenario, we utilized the following ROS packages: turtlebot3-core for base control, turtlebot3_slam_gmapping for SLAM, turtlebot3-lds for laser sensor support, turtlebot3-teleop for manual control, robot_state_publisher for broadcasting the state of the robot, usb_camera for image aquisition, and Rviz for 3D visualization.

C. Subjects

Since Appropriateness recognizability, Learnability, and User engagement required evaluation by the operators, 3 participants (3 males; aged 20-40, not familiar with the operating remote-control robot) were involved in this case study.

D. Conditions of each evaluation metric

1) Functional suitability: For Functional completeness, we counted the number of the following functional requirements for the fundamental DMU: (1) the system shall visualize sensory data, (2) the system shall plan routes for the robot, (3) the system shall create environmental maps, (4) the system shall provide HMIs to control the robot. In this case, 4 functional requirements were subject to evaluation. For Functional correctness, we verified the accuracy of the 2D occupancy map generated by the turtlebot3_slam-gmapping system which performs SLAM using the Gmapping algorithm, generating a map in real time from 2D laser scan data while simultaneously estimating its position. In FDNPS remote robot operation, a positional accuracy of at least 0.5 m is required; thus, we evaluated whether the generated map met this threshold. In this case, only this function was subject to evaluation.

2) Performance efficiency: For Time behaviour, we evaluated whether the throughput between the TurtleBot3 and the remote operation PC exceeded the required 1.5 Mbps under typical camera settings (640×480 pixels, 30 FPS, H.264 compression). Throughput was measured by the iperf3 command. For Resource utilization, we measured the average CPU, RAM, and GPU usage for Raspberry Pi and remote control PC by the glance and intel-gpu-top commands. For Capacity, we evaluated whether the DMU could handle a bandwidth of approximately 3.6 MB/s, corresponding to the same camera settings as above Resource utilization. Only the camera bandwidth was subject to evaluation, as it was anticipated to be the primary performance bottleneck.

3) Compatibility: For Co−existence, the Raspberry Pi camera and LDS-01 were selected as target devices, and we evaluated the availability of these hardware interfaces. For Interoperability, ROS2 [12]

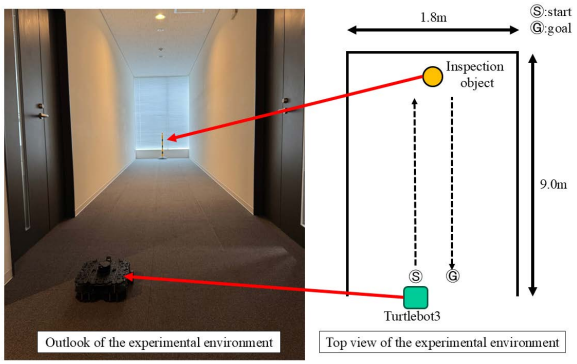

Fig. 2: Overview of the experimental environment.

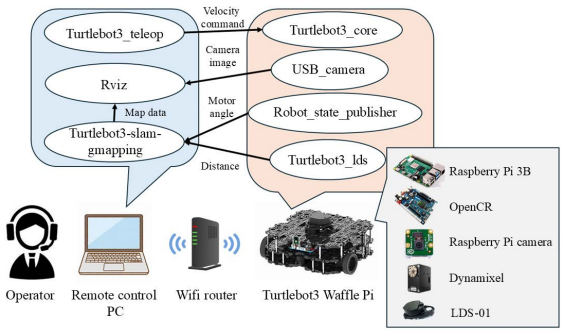

Fig. 3: The fundamental DMU and hardware configuration.

and OpenRTM-aist [13] were selected as target middlewares, and we evaluated the availability of these interfaces.

4) Interaction capability: For Appropriateness recognizability, the functions to be evaluated were turtlebot3_gmapping and turtlebot3_teleop. The subjects were asked to evaluate whether these functions were appropriate. For Learnability, we evaluated turtlebot3_gmapping and turtlebot3_teleop as targets. Subjects had 60 minutes to learn these functions based on the specified pages, and we counted the number they successfully learned within this time. For User engagement, we evaluated turtlebot3_gmapping and turtlebot3_teleop. The subjects were asked to evaluate whether these functions incorporated user engagement aspects. For Operability, User error protection, Inclusivity, User assistant, and Self-descriptiveness, we evaluated whether the fundamental DMU included functions as described in Table I.

5) Reliability: For Faultlessness, the functions under evaluation were turtlebot3_gmapping and turtlebot3_teleop. And we evaluated whether these functions would stop working once the fault occurred. For Fault tolerance, we evaluated usb_camera and turtlebot3_gmapping by simulating a pseudo-failure—unplugging the communication cable to the Raspberry Pi camera and LDS-01 during software execution—to determin whether the functions would stop working when a software failure occurred. For Recoveraiblity, we evaluated whether the fundamental DMU included functions as described in Table II.

6) Security: For the Confidentiality, Integrity, Non-repudiation, Accountability, Authenticity, and Resistance, we evaluated whether the fundamental DMU included functions as described in Table II.

7) Maintainablity: For the Modularity, Resusability, Analysability, Modifiability, and Testability, we evaluated whether the fundamental DMU included functions as described in Table II.

8) Flexibility: For Adaptability, we evaluated Windows, Ubuntu, and macOS as target operating systems (OS) and we counted OSs that could work with the fundamental DMU. For Scalability, we evaluated the maximum number of simultaneous connections to the remote-control robots while meeting the required throughput as defined in Time behaviour. In this case, we limited maximum number of robots to 1. For Replaceability, we evaluated the number of functions compatible with the ROS version change from Noetic to Melodic. The functions evaluated were turtlebot3 slam gmapping, turtlebot3 core, turtlebot3 lds, turtlebot3 teleop, and robot state publisher.

9) Safety: For Operational constraint, Risk identification, Fail safe, Hazard warning, and Safe integration, we evaluated whether the fundamental DMU included those functions as described in Table II.

E. Evaluation results

Table III summarizes the evaluation result of the fundamental DMU. The details of each evaluation item are explained below.

1) Functional suitability: The evaluation value of Functional suitability was set to 1.0 (A:4.0, B:4.0) since all requirements were satisfied. The evaluation value of Functional suitability was set to 0.0 because the vertical error was 5.0 m (=9.0-4.0) and the horizontal error was 0.3m (=1.8-1.5), resulting in an average error of 2.65 m, which exceeded the threshold (Fig.4). As a result, the evaluation value for Functional suitability was calculated as 0.5 (=1.0/2.0).

2) Performance efficiency: The evaluation value of Time behaviour was 1.0 because the iperf result was 3.7 Mbps, which exceeded the threshold of 3.6 Mbps. The evaluation value of Resource utilization was set to 0.51 (A:1.034, B:2.0) because the Raspberry Pi's average usage of CPU was 35.8%, RAM was 84.4%, and GPU was 0.0% (as it was a GPU-equipped model), while the remote operation PC's average usage of CPU was 15.2%, RAM was 22.3%, and GPU was 52.85%. The evaluation value of Capacity was set to 1.0 as the fundamental DMU was capable of handling bandwidth above 3.6 Mbps based on the Time behaviour result. As a result, the evaluation value of Performance efficiency was calculated as 0.78 (=2.35/3.0).

3) Compatibility: The evaluation value of Co-existence was set to 1.0 since ROS provided interfaces for target devices [10][11]. The evaluation value of Interoperability was set to 1.0 (A:2.0, B:2.0) since ROS provided interfaces for target middlewares [14][15]. As a result, the evaluation value for Compatibility was calculated as 1.0 (=2.0/2.0).

4) Interaction capability: The evaluation value of Appropriateness recognizability was set to 1.0 (A:2.0, B:2.0) as subjects judged these functions were appropriated. The evaluation value of Learnability was set to 1.0 (A:2.0, B:2.0) as subjects learnt these functions within the specified time. The evaluation value of Operability was set to 1.0 since ROS provided tools and libraries such as Rviz and Rqt for easy robot control. The evaluation value of User error protection was set to 0.0 since ROS did not provide any functionality to detect operational errors made by human operators. The evaluation value of User engagement was set to 0.5 (A:1.0, B:2.0) because the subjects considered turtlebot3_gmapping to exhibit engagement, as the gradual construction of the map was perceived positively. In contrast, turtlebot3_teleop was not regarded as engaging, as it involved a repetitive task of pressing keyboard keys and buttons. The evaluation value of Inclusivity was set to 1.0 since the ROS documentation was primarily written in English and the tutorial was partially available in Japanese translation. The evaluation value of User

TABLE III: Evaluation results of the fundamental DMU.

Characteristics	Characteristics value	Sub-characteristics	Sub-characteristics value
Functional suitability	0.5	Functional completeness	1.0
		Functional correctness	0.0
Performance efficiency	0.83	Time behaviour	1.0
		Resource utilization	0.51
		Capacity	1.0
Compatibility	1.0	Co-existence	1.0
		Interoperability	1.0
Interaction capability	0.81	Appropriateness recognizability	1.0
		Learnability	1.0
		Operability	1.0
		User error protection	0.0
		User engagement	0.5
		Inclusivity	1.0
		User assistant	1.0
		Self-descriptiveness	1.0
Reliability	0.16	Faultlessness	0.0
		Fault tolerance	0.5
		Recoverability	0.0
Security	0.0	Confidentiality	0.0
		Integrity	0.0
		Non-repudiation	0.0
		Accountability	0.0
		Authenticity	0.0
		Resistance	0.0
Maintainability	1.0	Modularity	1.0
		Reusability	1.0
		Analyzability	1.0
		Modifiability	1.0
		Testability	1.0
Flexibility	0.62	Adaptability	0.67
		Scalability	1.0
		Replaceability	0.2
Safety	0.6	Operational constraint	1.0
		Risk identification	1.0
		Fail safe	0.0
		Hazard warning	1.0
		Safe integration	0.0

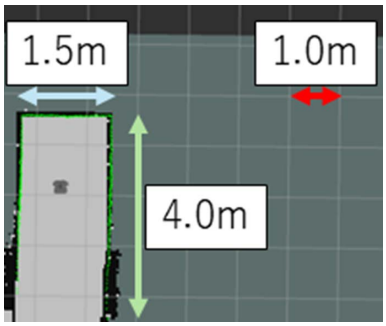

Fig. 4: The accuracy verification of the generated map.

assistance was set to 1.0 because the official ROS documentation and TurtleBot3 tutorials were available by employee within TEPCO. The evaluation value of Self-descriptiveness was set to 1.0 since ROS provided some tools with self-descriptive to users, such as Rviz, Rqt, and Gazebo. As a result, the evaluation value of Interaction capability was calculated as 0.81 (=6.5/8.0).

5) Reliability: The evaluation value of Faultlessness was set to 0.5 (A:1.0, B:2.0) because it was observed that the turtlebot3_gmapping node stopped updating the map when 2D laser scan data matching failed or communication errors occurred. In contrast, the turtlebot3_teleop node functioned stably and continuously during the experiment. The evaluation value of Fault tolerance was set to 0.0 (A:0.0, B:2.0) because the turtlebot3_gmapping stopped working when laser scan data was no longer delivered by the LDS-01 and the usb_camera stopped working if the camera device was not recognized or if it failed to deliver data. The evaluation value of recoverability was set to 0.0 since ROS was known to have difficulty recovering once they have gone down [16]. As a result, the evaluation value for Reliability was calculated as 0.16 (=0.5/3.0).

6) Security: The fundamental DMU did not implement functions such as Confidentiality, Integrity, Non-repudiation, Accountability, Authenticity, and Resistance, as these capabilities were not natively supported by ROS. Accordingly, the evaluation value for each of these sub-characteristics, as well as for security, was set to 0.0 (=0.0/6.0).

7) Maintainability: The fundamental DMU implements functions such as Modularity, Resusability, Analysability, Modifiability, and Testability, as it reflects the modular architecture of ROS, includes an anomaly evaluation tool (roswtf [17]), and provides a

test framework (rostest [18]). Accordingly, the evaluation value for each of these sub-characteristics, as well as for Maintainability, was set to 1.0 (=5.0/5.0).

8) Flexibility: The evaluation value of Adaptability was set to 0.67 (A:2.0, B:3.0) because ROS installation was supported on Windows and Ubuntu, according to the the official ROS website [19]. The evaluation value of Scalability was set to 1.0 since the evaluation result for Time behaviour satisfied the required throughput. The evaluation value of Replaceability was set to 0.2 (A:1.0, B:5.0) because melodic was not supported for TurtleBot3, according to the github of TurtleBot3 [20]. On the other hand, joint_state_publisher was supported, according to the official web page [21]. As a result, the evaluation value for Replaceability was calculated as 0.62 (=1.87/3.0).

9) Safety: The fundamental DMU implements functions such as Operational constraint, Risk identification, and Hazard warning, since ROS provided mechanisms for preventing hazardous inputs through tools such as ros_control [22], move_base [23] and diagnostics [24], which enabled periodic monitoring of the status of nodes and sensors, as well as anomaly detection. Accordingly, the evaluation value for each of these sub-characteristics was set to 1.0. On the other hand, the fundamental DMU does not implement functions such as Fail safe and Safe integration. Accordingly, the evaluation value for each of these sub-characteristics was set to 0.0. As a result, the evaluation value of Safety was calculated as 0.6 (=3.0/5.0).

10) Discussions: The results of the case study showed that the proposed methodology enabled quantitative evaluation of the fundamental DMU and allows relative comparisons with results obtained under different setups or requirement conditions. An insight gained through this case study was the evaluation value of Security was 0.0, indicating that ROS lacks security-related functions. ROS2 is a possible alternative to ROS, as it adopts the Data Distribution Service protocol, providing enhanced security while retaining the functionalities and features of ROS.

VI. CONCLUSIONS

In this paper, we proposed an evaluation methodology for DMU design based on the 2023 revision of ISO/IEC 25010, incorporating both functional and non-functional quality characteristics relevant to nuclear decommissioning. The methodology was applied to a fundamental DMU developed from a defined use case, tasks, and user requirements, enabling quantitative and relative comparisons with alternative setups and requirement conditions. In the future, we will select software and frameworks for DMU design using the proposed methodology.

REFERENCES

[1] N. Okuzumi, K. Matsuzaki, and S. Okada, Development and Application of Robotics for Decommissioning of Fukushima Daiichi NPS by IRID, J.Robot. Mechatron., Vol.36, No.1, pp.9-20, 2024.

[2] ISO: ISO/IEC 25010:2023, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.

[3] J. Uhlenkamp, J. Hauge, B. Jannicke E. Broda, M. Lütjen, M. Freitag, Michael, K. Thoben, Digital Twins: A Maturity Model for Their Classification and Evaluation, IEEE Access, vol.10, pp. 69605-69635, 2022.

[4] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng ROS: An open-source robot operating system, Proc. Open-Source Software Workshop Int. Conf. Robotics and Automation, 2009.

[5] H. Bruyninckx, Open robot control software: the OROCOS project, Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, Vol.3, 2001.

[6] E. Tsardoulias, P. Mitkas, Robotic frameworks, architectures and middleware comparison, 10.48550/arXiv.1711.06842, 2017.

[7] R. Temkar, A. Bhaskar, Quality Assurance of IoT based Home Automation Application using Modified ISO/IEC 25010, International Journal of Engineering Trends and Technology, vol.69, no.2, pp.92-101, 2021.

[8] Outline of Decommissioning, Contaminated Water and Treated Water Management, https://www.tepco.co.jp/en/hd/decommission/information/committee/pdf/2024/roadmap_20240627_01-e.pdf, 2024.

[9] Turtlebot3 Waffle Pi, https://e-shop.robotis.co.jp/product.php?id=351, Accessed: Aug. 11, 2025.

[10] usb_cam, https://wiki.ros.org/usb_cam, Accessed: Jul. 18, 2025.

[11] hls_lfcd_lds_driver, https://github.com/ROBOTIS-GIT/hls_lfcd_lds_driver, Accessed: Jul. 18, 2025.

[12] ROS2 Documentation, https://dock.ros.org/en/jazzy/index.html, Accessed: Jul. 18, 2025.

[13] OpenRTM-aist, https://openrtm.org, Accessed: Jul. 18, 2025.

[14] ros1_bridge, https://github.com/ros2/ros1_bridge, Accessed: Jul. 18, 2025.

[15] rtmros_common, https://start-jsk/rtmros_rtmros_common, Accessed: Jul. 18, 2025.

[16] P. Kaveti, H. Singh, ROS Rescue: Fault Tolerance System for Robot Operating System, Robot Operating System (ROS): The Complete Reference (Volume 5), Springer International Publishing, pp.381-397, 2021.

[17] roswtf, https://wiki.ros.org/roswtf, Accessed: Jul. 17, 2025.

[18] rostest, https://wiki.ros.org/rostest, Accessed: Jul. 17, 2025.

[19] ROS Noetic installation instructions, https://wiki.ros.org/noetic/installation, Accessed: Aug. 2, 2025.

[20] turtlebot3, https://github.com/ROBOTIS-GIT/turtlebot3, Accessed: Aug. 3, 2025.

[21] joint_state_publisher, https://wiki.ros.org/roswtf, Accessed: Aug. 3, 2025.

[22] ros_contorl, https://wiki.ros.org/ros_control, Accessed: Jul. 17, 2025.

[23] move_base, https://wiki.ros.org/move_base, Accessed: Jul. 17, 2025.

[24] diagnostics, https://wiki.ros.org/diagnostics, Accessed: Jul. 17, 2025.