אוניברסיטת בן-גוריון בנגב
Ben-Gurion University
of the Negev

Software Engineering Application Requirements Document (ARD) - AOS Middleware

Abstract

This document outlines the requirements for the development of a new middleware for the Autonomous Operating System (AOS). The middleware will feature RESTful user interface, facilitating the management and control of robot capabilities. The operations include receiving configuration files, automatic code generation, listening to public communication in the robot.

The system will operate in two modes:

- Configuration mode involves receiving configuration files and dynamically generating code. This generated code, when executed, enables the activation of robot capabilities, creation of notifications, and more.
- Execution mode, where the middleware will receive commands to execute capabilities, perform them, and update the planning engine.

Contents

Chapter 1

Introduction

1.1 The Problem Domain

The current landscape of Autonomous Operating System (AOS) usage presents multifaceted challenges that demand a comprehensive solution. Users and developers encounter a myriad of technical and operational difficulties in effectively managing autonomous robots. These challenges include:

Fragmented Tooling: Users and developers are compelled to rely on a disparate array of tools for various tasks such as robot activation, skill documentation, and debugging. This fragmented approach not only complicates workflows but also introduces the risk of inconsistencies and errors in documentation and monitoring.

Operational Cohesion: The separation of debugging and monitoring from the broader AOS system leads to inefficiencies and a lack of cohesion in the troubleshooting process. This disjointed approach hampers the ability to effectively debug and monitor the behavior of autonomous robots.

1.2 Context

In response to these pressing challenges, our project aims to revolutionize the AOS ecosystem by introducing a middleware solution that streamlines operations and enhances user experience. By seamlessly integrating with existing planning engines and robot infrastructures, particularly ROS2, our middleware will bridge the gap between disparate components, fostering operational cohesion and simplifying development workflows.

1.3 Vision

Our vision is to significantly enhance the development, management, and usability of autonomous robots within the AOS framework. We aim to achieve this by:

Streamlined Workflow: Developing a robust middleware solution that acts as an intermediary between the AOS planning engine and robot infrastructure, thereby simplifying development processes and fostering operational efficiency.

Enhanced Usability: Providing intuitive user interfaces for monitoring system status, ensuring that users can easily manage and monitor autonomous robots without significant technical expertise.

1.4 Stakeholders

Stakeholders in our project include developers, system administrators, and end-users involved in autonomous robot development. Additionally, our collaborating lab, Prof. Barfman's lab, underscores the importance of delivering a modular, easily maintainable, and flexible product to support future development endeavors.

1.5 Software Context

High-Level Description of the Software System:

Our planned software system is a middleware solution tailored for the Autonomous Operating System (AOS) environment, with a primary focus on enhancing usability, streamlining development processes, and facilitating effective communication between components. Emphasizing modularity, ease of maintenance, and flexibility for future enhancements, the system will feature RESTful user interfaces to cater to diverse user needs and preferences.

Major Inputs:

1. Metadata Configuration: User-provided metadata describing robot capabilities, environment, and goals.
2. User Commands: Commands from users through the RESTful interfaces, specifying actions such as activating robot capabilities or querying system status.

Outputs

1. System Notifications: Notifications and alerts informing users about the status of executed commands, system events, and potential issues.
2. Real-time Reports: Real-time reports and visualizations displaying the current state of robot capabilities, environment variables, and system alerts.

Common Usage Scenarios (Use-Cases):

1. Configuration Mode
    - User Action: Provide metadata about robot capabilities.
    - System Response: Process metadata, generate activation code, and listen to internal robot communication.
2. Activation Command:
    - User Action: Issue a command to activate specific robot capabilities.
    - System Response: Execute the specified capabilities, update the planning engine, and generate real-time reports.
3. User Interface Interaction:
    - System Response: Display real-time reports, visualize robot activities, and provide alerts if necessary.
4. RESTful Command:
    - User Action: Send a command through the RESTful interface to query robot parameters.
    - System Response: Process the command, retrieve relevant data, and provide a response through the interface.

Chapter 2

Usage Scenarios

2.1 User Profiles — The Actors

Autonomous Robot Developers:

- *Characteristics:*
  - Highly skilled professionals in robotics.
  - Proficient in configuring and programming autonomous robots.
  - Well-versed in metadata specifications and system integration.
- *Role:*
  - Engage in configuring the system by providing metadata about robot capabilities.
  - Activate and manage robot functionalities through the middleware.
  - Monitor real-time reports for performance optimization.

System Administrators:

- *Characteristics:*
  - Expertise in system management and maintenance.
  - Familiarity with middleware architecture and configuration.
  - Skillful in addressing system-related issues.
- *Role:*
  - Manage and maintain the middleware system.
  - Ensure system reliability and performance.
  - Address configuration and maintenance requirements.

External Systems:

- *Characteristics:*
  - Non-human entities, such as monitoring tools.
  - Capable of sending and receiving data programmatically.
  - Automated systems seeking data and control functionalities.

- *Role:*
    - Interact with the middleware through the RESTful interface.
    - Send commands to query robot parameters or initiate specific actions.
    - Receive data and responses from the middleware for external system integration

2.2 Use-Cases

1. Configure Robot Capabilities:

- *Description:* an autonomous robot developer or system administrator configures the system by providing detailed metadata about the robot's capabilities, including tasks and relevant details

  Actors: autonomous robot developer, system administrator
- *Pre-Condition:* middleware should be activated
- *Post-Condition:* robot capabilities are accurately configured and securely stored in the system.
- *Main Success Scenario:*
  1. The user accesses the configuration interface, which provides clear guidance and options.
  2. Inputs detailed metadata describing the robot's capabilities, with prompts and validation to ensure accuracy.
  3. The system efficiently processes and securely stores the configuration data, with real-time feedback to the user.
- The system confirms successful configuration, providing confirmation messages and access toconfiguration logs for verification

2. Activate Robot Capability

- *Description:* developer initiates the activation of a specific robot capability, enabling the robot to perform designated tasks efficiently and effectively. Activation can be performed through the intuitive via RESTful commands for seamless integration.

- *Actors:* AOS developer

- *Pre-Condition:* robot capabilities must be accurately configured and validated.

- *Post-Condition:* the specified robot capability is successfully activated and ready for immediate use.

- **Main Success Scenario:**
  1. The user navigates to the activation section within the interface, which clearly displays available capabilities and their status
  2. Selects the desired robot capability for activation, with visual indicators for clarity.
  3. The system promptly translates the activation request into executable tasks for the robot, with error handling and fallback mechanisms in place.
  4. Confirms successful activation, providing real-time

status updates and integration logs for verification.

3. **Monitor Real-Time Robot Behavior**

- **Description:** Developers monitor and analyze the real-time behavior and status of autonomous robots through the Gazebo simulator. This includes comprehensive insights into sensor data, task execution progress, and immediate alerts for timely intervention.
- **Actors:** AOS Developer
- Pre-Condition:
  - Robot capabilities must be successfully activated and operational.
  - Gazebo simulator is set up and integrated with the robot's ROS2 environment.
- **Post-Condition:** Developers gain actionable insights into the robot's behavior, enabling informed decision-making and optimization.
- Main Success Scenario:

1. **Access Gazebo Simulation:**
   The developer launches the Gazebo simulator tovisualize the robot in its environment.
2. Monitor Real-Time Data:
   - The developer observes real-time sensor data and task execution progress through Gazebo's interface.
   - The system continuously publishes relevant data from the robot to ROS2 topics, which Gazebo subscribes to and visualizes.
3. Receive Immediate Alerts:
   Gazebo provides dynamic visualizations and notifications for critical events, such as the robotreaching its goal or encountering obstacles.
4. Dynamic Updates:
   The system ensures that Gazebo's interface is continuously updated with real-time information, maintaining accuracy and relevance.
5. Gain Insights and Take Action:
   - The developer gains valuable insights into the robot's behavior, with the ability to make informed decisions.

        - Options for immediate action or further analysis are available based on the insights gained.

4. Send RESTful Command

- **Description:** External systems or users interact with the middleware through RESTful commands using Postman, enabling seamless integration and interoperability. Commands can include queries for robot parameters, initiation of specific actions, or retrieval of system information for enhanced functionality.
- **Actors:** External Systems
- Pre-Condition:
    - Middleware must be operational and accessible.
    - Secure authentication and authorization mechanisms must be in place.
    - Postman is set up and configured to interact with the middleware.
- **Post-Condition:** The system promptly processes and responds to the RESTful command, providing accurate data or executing requested actions.
- Main Success Scenario:
    - **Construct RESTful Command: Description**: The user can request the AOS server to build the project by integrating the project's skills andenvironment files through a RESTful command sent using Postman. There are several modes of Initialize Project Request: code generation only, inner simulation (without activating the robot), sequence of actions to run, robot activation, and robot activation/inner simulation without rebuilding the solver engine. The user can choose the integration mode and add additional parameters relevant to it (e.g., in robot activation mode, the user can choose the time interval between robot actions).
    - **Actors**: Autonomous Robot Developer, SystemAdministrator, and External Systems.
    - Pre-Condition:
        - The AOS server is operational.

  - At least one project exists in the system.
  - Postman is set up and configured to interact with the AOS server.

- **Post-Condition**: Initialize Project Request endssuccessfully.
- Main Success Scenario:

  1. Construct RESTful Request:

     The user constructs a valid RESTful Initialize Project Request using Postman, following thedocumented API specifications.
  2. Submit Request via Postman:
  3. The user submits the Initialize Project Requestto the AOS server through Postman, including proper authentication credentials and error handling mechanisms. Inquire Integration Mode and Parameters:
     The system prompts the user for the integration mode and any relevant parametersfor the chosen mode.
  4. Process Initialize Project Request:
     - The system sends the Initialize Project Request to the AOS server.
     - The AOS server processes the request and performs the integration to build the project's solver engine code (unless thechosen mode is code generation/start without rebuilding).
  5. Send Response:
     The system notifies the user via Postman thatthe Initialize Project Request has ended successfully.

5. *Initialize Project Request*

- **Description**: The user can request the AOS server to build the project by integrating the project's skills and environment files through a RESTful command sent using Postman. There are several modes of Initialize Project Request: code generation only, inner simulation (without activating the robot), sequence of actions to run, robot activation, and robot activation/inner simulation without rebuilding the solver engine. The user can choose the integration mode and add additional parameters relevant to it (e.g., in robot activation mode, the user can choose the time interval between robot actions).

- **Actors**: Autonomous Robot Developer, SystemAdministrator, and External Systems.

- **Pre-Condition:**
    - The AOS server is operational.
    - At least one project exists in the system.
    - Postman is set up and configured to interact with the AOS server.

- **Post-Condition**: Initialize Project Request endssuccessfully.

- **Main Success Scenario:**

6. Construct RESTful Request:

   The user constructs a valid RESTful Initialize Project Request using Postman, following thedocumented API specifications.
7. Submit Request via Postman:
   The user submits the Initialize Project Requestto the AOS server through Postman, including proper authentication credentials and error handling mechanisms.
8. Inquire Integration Mode and Parameters:
   The system prompts the user for the integration mode and any relevant parametersfor the chosen mode.
9. Process Initialize Project Request:
    - The system sends the Initialize Project Request to the AOS server.
    - The AOS server processes the request and

performs the integration to build the project's solver engine code (unless thechosen mode is code generation/start without rebuilding).

10. Send Response:

    The system notifies the user via Postman thatthe Initialize Project Request has ended successfully.

6. *Configure Solver Parameters*

- **Description:** The user configures solver parametersto optimize the robot's decision-making processes during simulations and operational runs.

- **Actors:** Autonomous Robot Developer, SystemAdministrator

- **Pre-Condition:** The AOS server is operational and theproject exists in the system.

- **Post-Condition:** Solver parameters are configuredand saved, optimizing the robot's performance.

- **Main Success Scenario:**

  1. Access Solver Configuration Interface:
     The user accesses the solver configuration interface or constructs RESTful commandsusing Postman.
  2. Define Solver Parameters:
     The user defines solver parameters such as thenumber of particles, planning time per move, and verbosity level.
  3. Submit Configuration:
     The user submits the configuration to the AOSserver.
  4. Confirm Configuration:
     The system confirms the successful configuration of solver parameters andprovides logs for verification.

7. *Generate Project Code*

- **Description:** The AOS system generates project codebased on the provided skills and environment files, enabling seamless integration with the ROS2 framework.
- **Actors:** Autonomous Robot Developer, System Administrator. **Pre-Condition:** The AOS server is operational and theproject exists in the system.
- **Post-Condition:** The generated project code is readyfor deployment and further development.
  - Main Success Scenario:

1. Construct RESTful Request:
   The user constructs a valid RESTful requestusing Postman to generate project code.
2. Submit Request via Postman:
   The user submits the request to the AOS server with the required parameters, includingPLPs directory path and ROS target configuration.
3. Process Request:
   The AOS server processes the request,generating the necessary project code.
4. Confirm Code Generation:
   The system confirms the successful generation of the project code and provideslogs for verification.

9. Perform Internal Simulation

- **Description:** The user initiates an internal simulation to test the robot's behavior and decision-making processes within a controlled environment.

- **Actors:** Autonomous Robot Developer, SystemAdministrator.

- **Pre-Condition:** The AOS server is operational and theproject exists in the system.

- **Post-Condition:** The internal simulation runs successfully, providing insights into the robot's performance.

  - Main Success Scenario:

1. Construct RESTful Request:
   The user constructs a valid RESTful requestusing Postman to perform an internal simulation.

   Submit Request via Postman: The user submits the request to the AOSserver with the necessary parameters, including the internal simulation flag.
2. Process Request:
   The AOS server processes the request and performs the internal simulation.
3. Analyze Simulation Results:
   The system provides the results of the internal simulation, allowing the user to analyze the robot's behavior and decision-making processes.

2.3 Special Usage Considerations

Special Requirements:

1. Scalability:

   Description: The system should be able to scale effectively to accommodate a growing number of autonomous robots and diverse capabilities. It should handle increased traffic and data exchange without compromising performance.

2. Compatibility with ROS Versions:

   Description: Ensure compatibility with multiple versions of the Robot Operating System (ROS) to facilitate integration with different robot infrastructures.

3. Real-time Data Processing:

   Description: The system must be capable of processing real-time data efficiently, ensuring timely responses and decision-making capabilities.

4. Interoperability with External Systems:

   Description: Ensure seamless interoperability with external systems, allowing the middleware to integrate data from various sources.

5. User Interface Responsiveness:

Description: The RESTful user interfaces should be responsive, providing users with a smooth and efficient interaction experience.

Chapter 3

Functional Requirements

1. System Initialization and Configuration
   - Description: The system must initialize and configure settings to ensure proper functionality
   - Priority: Must-Have
   - Risk: Low
2. Metadata Reception and Processing
   - Description: The system must receive and process metadata for configuring and connecting to robots.
   - Priority: Must-Have
   - Risk: Medium
3. Communication Monitoring
   - Description: The system must continuously monitor communication within the robot environment
   - Priority: Must-Have
   - Risk: Medium
4. Running Mode
   - Description: The system must activate and execute robot capabilities based on configured settings.
   - Priority: Must-Have
   - Risk:High
5. Robot Infrastructure Compatibility
   - Description:The system must ensure compatibility with various robot infrastructures, with ROS2 as the default
   - Priority: Must-Have
   - Risk: Medium.
6. Dynamic Parameter Listening
   - Description: The system must dynamically listen to changes in ROS topics and parameters.
   - Priority: Must-Have
   - Risk: Medium

7. Event-Based Notification System
   - Description: The system should implement a notification system to alert users about significant events.
   - Priority: Should-Have

- Risk: Low

8. Remote User Interface
    - Description: The system must provide a web-based interface for remote monitoring and control.
    - Priority: Must-Have
    - Risk:Medium
9. Integration with AOS Planning Engine
    - Description:The system must seamlessly integrate with the AOS planning engine for coordinated decision-making.
    - Priority: Must-Have
    - Risk: Medium
10. System Interface with External Devices
    - Description:The system could allow interfaces with external devices for extended functionality.
    - Priority: NTH
    - Risk:Low

Chapter 4

Non-functional Requirements

4.1 Implementation Constraints:

- **Performance:**

  Description: The middleware system should ensure fast transaction completion times to enhance overall performance.

- **Reliability & Stability:**

  Description: The middleware system needs to support data recovery and error-correction mechanisms to ensure reliability and stability.

- **Safety & Security:**

  Description: The middleware system should prioritize confidentiality, implementing encryption for data security and controlling user access based on authority levels.

- **Usability:**
  Description: The middleware system should prioritize a user-friendly interface, accommodating users with varying levels of technical expertise.

- **Availability:**

  Description: Factors affecting system availability, such as maintenance and updates, should be carefully managed to meet the specified availability level.

4.2 Platform Constraints:

The development must be done using Python & C# . The system should be designed to integrate seamlessly with ROS2 (Robot Operating System 2).

The middleware system must be compatible with existing middleware technologies, ensuring smooth communication with ROS services and actions.

The system must be deployable on Linux (Ubuntu 20)-based operating systems to align with ROS2 requirements.

The middleware system should utilize a database system compatible with Python & SQL, MongoDB, for efficient data storage and retrieval.

4.2.1 SE Project Constraints:

The inputs for the system during the demo will be provided based on the information and metadata received from users or developers. The specifics of the inputs and their sources will be coordinated and managed by our project monitor, who will facilitate the necessary data to showcase the system's capabilities effectively.

4.3 Special Restrictions & Limitations:

- **Assumptions:**
  Description:The design assumes access to standard development tools and hardware. Simulations may be required for specific hardware conditions. The middleware system assumes users have basic computer literacy.

Chapter 5

Risk Assessment & Plan for the Proof of Concept

Risk Assessment and Feasibility Study Plan

- **Objective:**
  To define the prototype's scope, identify potential risks, and justify the selection of specific features for the Proof of Concept (PoC).

- **Prototype Scope:**
  The prototype will serve as a demonstration of fundamental functionalities of the middleware system. It will focus on core features essential for autonomous robot development within the AOS framework.

- **Justification for Starting Features:**
  1. Basic System Development:
     ★ Rationale: Starting with a basic version of the system allows us to establish a foundation for further development and testing.
  2. MongoDB Integration:
     ★ Rationale: MongoDB integration provides efficient data storage and management capabilities, crucial for handling metadata and system configurations.
  3. RESTful User Interface Communication:
     ★ Rationale: Enabling RESTful communication enhances system accessibility by providing a standardized interface for interaction.
  4. Robot Capability Demonstration:
     ★ Rationale: Demonstrating the robot's capability to execute specific actions based on commands showcases the system's core functionality and integration with robotic infrastructure.

- **Risk Assessment:**
    1. Integration of New Technologies:
        - ★ Mitigation: Adopt a phased integration approach, conduct thorough compatibility and performance testing at each stage to minimize risks.
    2. Listening to Internal Robot Communications:
        - ★ Mitigation: Develop clear communication protocols and compatibility layers to ensure seamless integration with internal robot communications.
    3. RESTful User Interface:
        - ★ Mitigation: Prioritize effective API design, implement robust security measures, and focus on efficient user interaction management to mitigate risks associated with RESTful interface implementation.

Plan for Proof of Concept:

1. Develop a basic version of our system to showcase fundamental functionalities.

2. Implement MongoDB integration to efficiently store and manage essential data.

3. Enable RESTful user interface communication to enhance system accessibility.

4. Demonstrate the robot's capability to execute specific actions based on received commands.

5. Develop a prototype of the system using ROS2:
Explain the decision to use ROS2 as the framework for prototyping the system.
Outline the specific ROS2 components and functionalities that will be implemented in the prototype.
Discuss how the ROS2 prototype will interact with other components of the system , such as MongoDB integration and RESTful communication.
Detail the expected benefits of using ROS2 for prototyping, such as its robustness in managing robot behavior and its compatibility with various hardware platforms.

- **Feasibility Study:**
  The feasibility study will assess the technical, economic, and operational viability of the proposed middleware system. It will include an analysis of existing technologies, market trends, and user requirements to determine the project's potential for success.