

Enhancing Robotic Systems in Healthcare: A Preliminary Analysis of Agent-Based Paradigms and Simulation Environments

Valeria Seidita^{1,2,*}, Antonio Chella^{1,2}

¹Department of Engineering, University of Palermo, Italy

²ICAR-CNR National Research Council, Palermo, Italy

Abstract

The integration of agent-based paradigms and robotic simulation environments has become increasingly important in the design and development of robotic systems. As with any complex software, the development of robotic software requires disciplined processes to ensure efficiency and robustness. This paper presents the preliminary results of an analysis conducted as part of two healthcare projects. In these projects, we explore intelligent patient support and make robotic systems explainable to improve human-robot interaction. The aim is to increase patient confidence in technology, make interactions more efficient, and improve care outcomes. In this context, we propose the adoption of the agent-based paradigm as a systematic approach to cover the notable gap in the literature and practice regarding an engineering discipline that systematically addresses the complexities of designing agent-based robotic systems with the use of simulation. This gap underscores the need for further research to develop methodologies that embrace the technical capabilities of robotic simulators and exploit the benefits of agent-based architectures.

Keywords

Healthcare robotics, Robotic simulation, Robot Operating System (ROS)

1. Introduction

In the ever-evolving field of robotics, the design and development of robust and effective systems require disciplined and adaptable engineering approaches. This is particularly evident in the domain of assistive healthcare robotics [1, 2], where systems must interact complexly with humans and the surrounding environment, doing so with high reliability and flexibility. The growing need for advanced robotic solutions is evident in healthcare sectors, where the aging global population presents unprecedented challenges. This scenario has significantly stimulated interest in innovative approaches to robotic design that can meet these complex needs.

In the context of healthcare robotics, robotic systems must navigate dynamic and unpredictable environments, interact safely and effectively with patients, and perform a wide range of tasks, from monitoring vital signs to assisting with mobility or daily activities. The inherent


WOA 2024: 25th Workshop "From Objects to Agents", July 8-10, 2024, Forte di Bard (AO), Italy

*Corresponding author.

†These authors contributed equally.

✉ valeia.seidita@unipa.it (V. Seidita); antonio.chella@unipa.it (A. Chella)

ORCID 0000-0002-0601-6914 (V. Seidita); 0000-0002-8625-708X (A. Chella)

 © 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

complexity of these tasks, combined with the high standards of safety and usability required, necessitates a design methodology that goes beyond traditional engineering approaches.

The adoption of robotic simulators represents a fundamental solution to address these challenges. ROS (Robot Operating System) [3, 4] and robotic simulators such as Gazebo [5, 6] and Webots [7, 8], among others, offer controlled and flexible environments for testing algorithms, hardware designs, and robot-environment interaction scenarios without the risks or costs associated with real-world experiments. These tools provide an essential foundation for building and testing robotic applications in simulated environments that mimic real-world conditions. *However, despite their widespread adoption, a structured engineering approach for the design and development of these systems is still lacking.*

The work presented here is part of research from two ongoing projects. The first project aims to develop a robotic support system for patients with metabolic deficits, providing guidance for physical exercises and monitoring progress for both patients and their physiotherapists. The second project, also within the healthcare domain, focuses on analyzing the effects of explainability and trustworthiness in supporting patients and their caregivers during the post-hospitalization period.

Our primary research interests include modeling the knowledge robots possess about highly dynamic environments, updating this knowledge during the design phase, and implementing a decision-making process that can explain its decisions and reasoning. In the initial stages of both projects, we conducted a theoretical analysis to thoroughly define the application domains, determine the appropriate technologies to use, and apply the agent paradigm to the implementation of the robotic systems.

In this paper, we present the initial results of the analysis conducted to use agents as a design paradigm for developing complex robotic systems with ROS and robotic simulators, particularly considering Gazebo and Webots. We focused on and used the concept of a metamodel to map the agent paradigm to the simulation environments.

2. From Agent to Developing Robotic Systems with Robotic Simulators

In the context of robotic systems, the use of the agent-based design paradigm [9, 10] has proven effective in managing complexity and scalability. Agents are autonomous computational entities capable of perceiving the surrounding environment, processing information, making decisions, and acting autonomously to achieve predefined goals. The adoption of the agent-based paradigm for the development of robotic systems, particularly those integrated with simulators such as Gazebo and Webot integrated with ROS, represents a strategic and innovative choice to address the challenges inherent in complex and interactive environments.

Our work is based on the concept of a metamodel [11, 12], and after studying ROS, Gazebo, and Webots and how robotic systems are developed with these tools, we hypothesized a metamodel for the design of computation and control in the operation of ROS.

A metamodel is a model that describes other models, providing a higher level of abstraction that defines the structure, rules, and relationships for constructing specific domain models. A metamodel establishes guidelines for executing a design process and spans four different

levels of abstraction. A metamodel for designing agent-based systems includes elements such as Agent, Role, Task, Action, Goal, Environment, Message, and Capability. These elements form the modeling language that can also be used to describe agent-based robotic systems.

2.1. Robot Operating System and simulation environments

In this subsection, we delve into the features of ROS, Gazebo, and Webots, and how they collectively enhance the design, testing, and deployment of robotic systems.

2.1.1. ROS (Robot Operating System)

ROS (Robot Operating System) is a flexible framework designed for developing robotic software. It includes a suite of tools and libraries that assist developers in creating complex and robust robotic applications on various hardware platforms. ROS provides services typical of an operating system across a heterogeneous network of computers, such as low-level device control, implementation of commonly required functionalities, inter-process message passing, and package maintenance. This makes it particularly useful for developing sophisticated robots requiring effective inter-module communication. The primary goal of ROS is to offer capabilities for creating powerful, reusable robotic applications. ROS is composed of several elements.

ROS Nodes are processes that use ROS functionalities to perform data computations. A robot using ROS consists of multiple nodes that communicate with each other, each node performing a specific task and capable of messaging other nodes. For instance, a node might process data from a laser scanner to prevent collisions. Nodes are written with the help of ROS client libraries like `roscpp` or `rospy`, utilizing object-oriented programming languages at a low level.

Messages in ROS are data structures used for communication between nodes via topics or services. Topics are asynchronous communication channels through which nodes exchange messages, while services offer synchronous function calls between nodes, meaning the calling node waits for a response from the service provider node. This type of communication is typically one-to-many, allowing multiple nodes to subscribe to a topic. At the implementation level, the ROS Master acts as a central node that links other nodes together, though this detail is less relevant to our current analysis.

Another significant tool in ROS is the Plugin. Plugins allow developers to use and integrate existing software without modifying the original code, extending the capabilities of simulators by adding components such as sensors or control algorithms. Developers must be aware of existing plugins or able to create new ones to build complex functionalities through integration.

To develop a robotic application with ROS, a developer should:

- Define the system architecture by dividing the robot's functionalities into separate nodes.
- Identify the communications these nodes will exchange.
- Implement processes within the nodes to manage tasks.
- Establish the types of messages exchanged by the topics.
- Define the topics on which nodes will publish or subscribe.

Custom messages may be necessary. After identifying functionalities requiring synchronous communication, services should be implemented. Finally, before testing and debugging, integration with the simulators is carried out. Plugins are used to integrate specific functionalities for

instance into the Gazebo simulator, allowing for the testing of the robot's behavior in a virtual environment.

2.1.2. Gazebo

Gazebo is an advanced robotic simulator that allows the simulation of robots in complex, dynamic environments with realistic physics. It is widely used in robotics research and development because it provides a controlled and flexible environment for testing algorithms, hardware designs, and robot-environment interaction scenarios without the associated risks or costs of real-world experiments. Gazebo integrates seamlessly with ROS, enabling developers to use ROS nodes, messages, and services directly within Gazebo's simulated environment. Specific plugins allow commands and data to flow smoothly between ROS systems and Gazebo simulation. One of Gazebo's strengths is its ability to simulate accurate physical interactions between robotic components and their environment. It provides an extensive library of simulated sensors and ready-to-use robot models, easily integrated and configured in projects. This allows developers to simulate complex scenarios with various types of sensors and actuators without having to build them from scratch. Users can create and modify detailed simulated environments to test their robots in different contexts. Gazebo plugins are specifically designed to extend the capabilities of the simulator. They are mainly used to add specific behaviors directly to models within the simulation, such as controlling a robot's movements, simulating sensors, or interacting with the simulated physical environment.

It's essential to emphasize that ROS nodes are software components performing specific computation, control, or data processing functions within the ROS framework. They handle a wide range of tasks, from processing sensor data to controlling actuators, managing path planning, and handling communication between nodes. While Gazebo plugins operate within the Gazebo simulation environment with direct access to APIs, allowing them to implement detailed and low-level operational behaviors, ROS nodes are more flexible and modular. They communicate through a messaging system on topics or services, making them suitable for implementing and orchestrating high-level logic. ROS nodes can be directly transferred to real robots without relying on Gazebo, while Gazebo plugins can simulate behaviors managed by ROS nodes in the simulation context.

2.1.3. Webots

Webots is a 3D robot simulator supporting a wide range of commercial and custom robots, providing a realistic simulation environment. A Webots world represents the environment where robots operate, defined by a configuration file specifying objects, terrain, lighting, and other environmental elements. This world is equivalent to the environment in the agent paradigm, representing the physical context in which agents operate. Robots in Webots are mobile entities equipped with various sensors and actuators that can be programmed to interact with the environment and perform specific tasks. These robots are comparable to agents, being autonomous entities that perceive the environment, make decisions, and act to achieve goals. Devices in Webots, including sensors and actuators, extend the capabilities of robots, representing the capabilities of agents in the agent paradigm. These devices model what the

agent can do and the actions it can perform. A supervisor in Webots is a special type of robot that can control and monitor other robots in the simulated world, similar to a high-level agent managing and coordinating other agents' actions. Additionally, a controller in Webots is a program defining a robot's behavior, executing cycles of sensor reading, data processing, and command sending to actuators. This is equivalent to the plan or task of an agent, defining the sequence of actions the agent must perform to achieve its goals. The interaction between ROS and Webots offers a powerful combination that merges ROS's flexibility with Webots' realistic simulation environment. This integration allows robots simulated in Webots to leverage ROS's advanced control and communication capabilities. ROS nodes, representing autonomously executable processes, can be used within Webots to control robots, enabling developers to apply existing ROS packages and libraries to define the robot's behavior.

Robots simulated in Webots can publish and subscribe to messages through ROS topics, allowing efficient communication between various system components. Additionally, it is possible to execute ROS service calls for synchronous operations and use ROS actions to manage complex asynchronous operations. This bidirectional interaction facilitates the realization of complex simulations and allows for rapid development, testing, and iteration of robot behaviors, reducing the time and costs associated with field tests.

2.2. Relationship between the agent paradigm, ROS, and simulators

In this subsection, we briefly present the results of our analysis of ROS, Gazebo, and Webots for the design of robotic systems and their relationship with the metamodel elements of the agent paradigm. The results are summarized in the tables below.

Agent Paradigm Concept	Concept in Webots	Description
Agent	Robot	A mobile entity that can interact with the environment and perform specific tasks. It is equivalent to the agent in the agent paradigm, being an autonomous entity capable of perceiving, deciding, and acting to achieve goals.
Capability	Device (Sensors and Actuators)	Extensions to the robot's capabilities, representing what the agent can do and the actions it can perform.
Environment	World	The physical context in which robots operate, defined by a configuration file specifying objects, terrain, lighting, and other environmental elements.
Task/Plan	Controller	Programs that define the robot's behavior, performing sensor readings, data processing, and sending commands to actuators. Equivalent to the agent's plan or task.
Action	Action performed by controllers and actuators	Specific operations a robot can perform to interact with the environment or achieve its goals.

Table 1

Mapping between agent paradigm concepts and Webots concepts.

Agent Paradigm Concept	Concept in Gazebo	Description
Agent	Robot	An autonomous entity capable of perceiving the environment and acting to achieve goals.
Capability	Sensor and Actuator	Devices that can be mounted on robots to extend their capabilities, representing the actions an agent can perform.
Environment	World	The simulated physical context defined by a configuration file, including terrain, objects, and lighting.
Task/Plan	Plugin	Code that extends robot or environment functionalities, defining specific behaviors and functions of a robot or component.
Action	Actions executed by plugins and actuators	Specific operations that a robot can perform to interact with the environment or other robots.
Communication	Topic, Service, Action	Mechanisms for communication between ROS nodes or Gazebo components, supporting coordination and information exchange.

Table 2
Mapping between agent paradigm concepts and Gazebo concepts.

Agent Paradigm Concept	Concept in ROS	Description
Agent	Node	An autonomous process that performs computations, similar to an agent perceiving the environment, making decisions, and acting.
Capability	Topic, Service, Action, Message	Mechanisms for communication, perception, and interaction with the environment and other nodes.
Environment	ROS Master	The operational context managing node communication and registration.
Task/Plan	Node Scripts	Programs defining specific behaviors and functions of a node, similar to an agent's plan or task.
Action	Service Call, Topic Publishing, Action Execution	Operations that a node can perform to interact with the environment or other nodes.
Communication	Topic, Service, Action	Mechanisms for asynchronous message passing, synchronous service requests, and complex asynchronous interactions, supporting agent cooperation and coordination.

Table 3
Mapping between agent paradigm concepts and ROS concepts.

Moreover in ROS the Supervisor is a special type of robot that can control and monitor other robots in the simulated world, akin to a high-level agent managing and coordinating the actions of other agents. In Gazebo the Supervisor Plugins is the one that can monitor and control other robots and components within the simulation, similar to a high-level agent managing

other agents and in Webots the Supervisor Node is the node responsible for monitoring and controlling other nodes, coordinating activities and handling anomalies. These elements leave the way open for reasoning about how agent organisations can be implemented, which for now are out of the analysis presented.

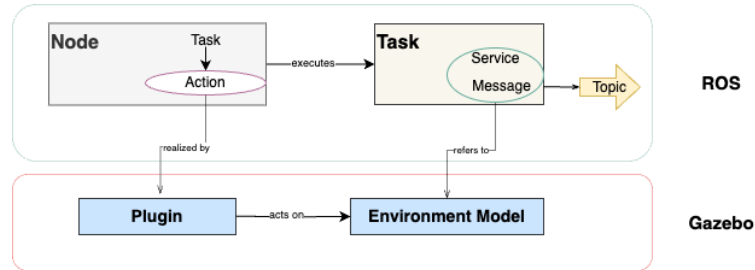


Figure 1: The process of programming with ROS and Gazebo

The results of this initial part of the study are shown in the following two figures. Figure 1 visually illustrates how the key elements of ROS and Gazebo relate to each other in the context of robotic programming. The underlying rationale is the same for Webots. This figure provides a detailed mapping of the fundamental components and concepts of ROS and Gazebo and how they interact to create an integrated robotic system. It outlines the logical design process followed to realize a robotic system. ROS nodes are shown as blocks performing specific functions or controls. Nodes communicate with each other using messages and topics, represented by arrows indicating the flow of information between nodes. Additionally, ROS services, allow for synchronous function calls between nodes, and ROS actions, which handle complex asynchronous interactions. The figure also shows Gazebo plugins, which extend the simulator's capabilities by enabling the implementation of simulated sensors, actuators, and other robotic components.

In Figure 2, we present a first mapping between the fundamental concepts of ROS and the agent metamodel paradigm. This mapping is essential to understand how the abstract concepts of the agent paradigm can be concretely implemented using the functionalities offered by ROS. In the agent metamodel paradigm, an agent is seen as an autonomous entity that can perceive the environment, make decisions, and act to achieve its goals. In ROS, this concept is represented by nodes, which are independent processes performing specific computations and controls. Each node in ROS can be compared to an autonomous agent operating within a broader system, communicating with other nodes via messages, services, and actions.

The concept of an agent's capability, which describes the skills and actions an agent can perform, is mapped in ROS through the use of topics, services, and actions. Topics allow for asynchronous communication between nodes, services offer synchronous function calls, and actions handle more complex asynchronous interactions. These communication mechanisms enable nodes to coordinate and collaborate to achieve common goals, replicating the capabilities of agents in the metamodel. The environment, which in the agent paradigm represents the physical and operational context in which agents interact, is managed in ROS by the ROS Master. The ROS Master acts as a central registry that keeps track of all nodes and facilitates their

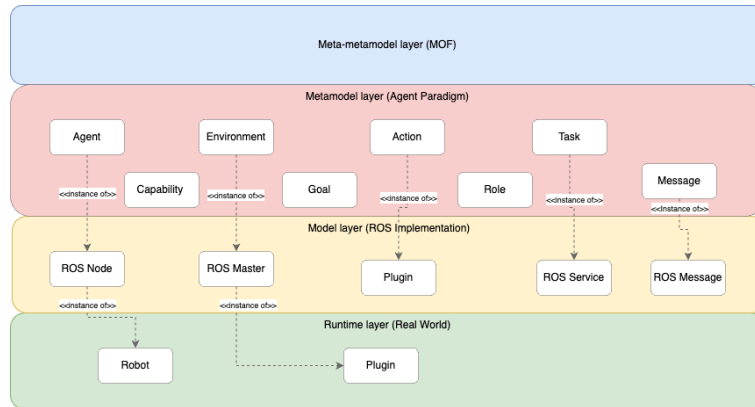


Figure 2: Mapping between ROS and agent paradigm concepts

communication. This allows for the creation of a structured and coordinated operational environment in which nodes can effectively operate. The supervisor in the agent paradigm, which monitors and coordinates the activities of other agents, is represented in ROS by supervisor nodes. These nodes are responsible for monitoring the status and performance of other nodes, intervening in case of anomalies or errors. Finally, the tasks or plans of agents, which define the sequences of actions to be performed to achieve goals, are implemented in ROS through scripts and programs executed within nodes. These scripts can be written in various programming languages supported by ROS, such as Python or C++, and define the specific behavior of each node.

3. Conclusion

In this paper, we explored the integration of agent-based paradigms and robotic simulation environments, specifically focusing on ROS, Gazebo and Webots, in the context of healthcare robotics. Our initial theoretical analysis and the application of the metamodel concept demonstrated the potential benefits of adopting agent-based design for developing complex robotic systems. We highlighted the significance of using robotic simulators like ROS, Gazebo, and Webots to address the inherent challenges in creating reliable and flexible robotic systems for dynamic and unpredictable environments. These simulators provide essential tools for testing and validating robotic behaviors in controlled settings, reducing the risks and costs associated with real-world experimentation.

Our research is part of two ongoing projects aimed at improving patient support through advanced robotic systems. In both projects, the use of agent-based paradigms has shown promise in managing the complexity and scalability of the systems. By mapping the concepts of the agent paradigm to the functionalities of ROS, Gazebo and Webots, we established a structured approach for designing and implementing robotic systems. This mapping provides a clear framework for translating high-level agent-based models into practical robotic applications, enhancing the development process and ensuring robust system performance.

The adoption of agent-based paradigms and the use of advanced robotic simulators represent

a strategic and innovative approach to addressing the challenges in healthcare robotics. Our findings underscore the need for further research to refine these methodologies and explore their application in other domains. Future work will focus on validating our theoretical results through practical implementations and expanding our analysis to include additional simulation environments and robotic platforms.

Acknowledgments

The work has been supported by the PRIN 2022 project I-TROPHYTS - IoT and humanoid Robotics for autonomic PHYsio-Therapeutic monitoring, coaching and supervising in smart Spaces: a feasibility study, P20224TAETP and PRIN-PNRR 2022 ADVISOR - ADaptiVe legIble robotS for trustwORthy health coaching.

References

- [1] I.-H. Kuo, E. Broadbent, B. MacDonald, Designing a robotic assistant for healthcare applications, in: the 7th conference of Health Informatics New Zealand, Rotorua, 2008.
- [2] P. Shubha, M. Meenakshi, Design and implementation of healthcare assistive robot, in: 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), IEEE, 2019, pp. 61–65.
- [3] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al., Ros: an open-source robot operating system, in: ICRA workshop on open source software, volume 3, Kobe, Japan, 2009, p. 5.
- [4] L. Joseph, J. Cacace, Mastering ROS for Robotics Programming: Design, build, and simulate complex robots using the Robot Operating System, Packt Publishing Ltd, 2018.
- [5] N. Koenig, A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, in: 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566), volume 3, Ieee, 2004, pp. 2149–2154.
- [6] M. Marian, F. Stîngă, M.-T. Georgescu, H. Roibu, D. Popescu, F. Manta, A ros-based control application for a robotic platform using the gazebo 3d simulator, in: 2020 21th International Carpathian Control Conference (ICCC), IEEE, 2020, pp. 1–5.
- [7] O. Michel, Webots: Symbiosis between virtual and real mobile robots, in: Virtual Worlds: First International Conference, VW'98 Paris, France, July 1–3, 1998 Proceedings 1, Springer, 1998, pp. 254–263.
- [8] O. Michel, Cyberbotics ltd. webots™: professional mobile robot simulation, International Journal of Advanced Robotic Systems 1 (2004) 5.
- [9] J. Ferber, G. Weiss, Multi-agent systems: an introduction to distributed artificial intelligence, volume 1, Addison-wesley Reading, 1999.
- [10] M. Wooldridge, An introduction to multiagent systems, John wiley & sons, 2009.
- [11] Q. Wang, D. Astruc, State of the art and prospects in metal–organic framework (mof)-based and mof-derived nanocatalysis, Chemical reviews 120 (2019) 1438–1511.
- [12] Model-driven development: a metamodeling foundation, IEEE software 20 (2003) 36–41.