DOI: 10.12700/AIS.2025.031

Simulation and Semantic Description of a Mobile Manipulator

Csaba Hajdu
Departement of Informatics
Sze´chenyi University
Gyo˝r, Hungary
hajdu.csaba@sze.hu

Zolta´n Szila´gyi
Doctoral School of Applied Informatics
and Applied Mathematics, Óbuda University
Budapest, Hungary
szilagyi.zoltan@amk.uni-obuda.hu

Ka´roly Sze´ll
Technical Institute
Alba Regia Faculty, Óbuda University
Sze´kesfehe´rva´r, Hungary
szell.karoly@amk.uni-obuda.hu

Pe´ter Galambos
Antal Bejczy Center for Intelligent Robotics,
Óbuda University
Budapest, Hungary
peter.galambos@irob.uni-obuda.hu

Abstract—The design and implementation of robotic systems have increasingly shifted toward software-centric development, where simulation plays a central role in the validation and integration of the system. State-of-the-art simulators such as Gazebo and NVIDIA Isaac Sim support rapid prototyping, but primarily emphasize geometric, physical, and visual modeling. Meanwhile, modern robotic systems have become highly interdisciplinary—requiring the consistent integration of mechanical, control, communication, and software layers. This paper introduces a hypergraph-based ontological description framework that unifies these heterogeneous aspects within a single, modular representation. The proposed model leverages the mathematical formalism of directed hypergraphs to capture many-to-many relationships among components, supporting scalable and interoperable robot design. A case study demonstrates the construction of a mobile-manipulator robot, including its corresponding Gazebo simulation and ROS configuration files. Future work will extend the framework toward multi-robot systems and deeper software-level abstractions, enabling greater reusability and semantic consistency across simulation, deployment, and control environments.

Index Terms—mobile manipulator, hypergraph, domain modeling, simulation

I. Introduction

Modern robotic design relies heavily on accurate kinematic and physical descriptions, which together define the structural and dynamic characteristics of robotic systems. These descriptions serve multiple purposes throughout the design and implementation workflow:

- verifying the kinematic structure and visual appearance of the robot,
- loading the model into simulation environments for visualization and performance evaluation,
- providing a mock-up for software development and system integration, and
- supporting control experiments, such as reinforcement learning or behavior optimization.

Simulation toolkits typically require a well-defined physical description for visualization and basic analysis. However, deeper system integration also depends on detailed representations of sensors, actuators, and their interfaces, including communication links, parameter configurations, and calibration data. Existing description formats—such as URDF [1], SDF, MuJoCo [2], and OpenUSD—often handle interface information separately from the kinematic and physical structure. While this separation is practical for inclusion in simulators like Gazebo [3], it can lead to fragmentation when deploying or synchronizing configurations within complex systems such as those built on the Robot Operating System (ROS).

To address this issue, the present paper introduces a domain modeling methodology based on directed hypergraphs, capable of representing the intricate many-to-many relationships between interface and physical components within robotic systems. Building on prior work in hypergraph-based kinematic modeling [4], this study extends the framework to support the automatic generation of configuration files for ROS 2 and Gazebo, providing a more unified and semantically consistent approach to robotic system description.

II. Related work

This study focuses on the automated generation of configuration and description files for widely used robotic simulators, with particular emphasis on Gazebo [3]. Gazebo is among the most prevalent simulation environments in robotics and is tightly integrated with the Robot Operating System (ROS) [5]. It enables the creation of comprehensive mock-ups of

robots—including sensors, actuators, and control interfaces—thereby facilitating software development, testing, and system configuration. Other notable simulation frameworks, such as MuJoCo [2] and PyBullet [6], focus primarily on the accurate modeling of the robot's physical structure and its surrounding environment.

Regarding robot description formats, most simulators rely on URDF [1] or SDF, while MuJoCo employs its own XML-based schema. These formats are primarily designed to represent:

- the kinematic structure of the robot using a hierarchical tree-based approach,
- the visual appearance and collision geometry,
- inertial and dynamic parameters, and
- the spatial placement of sensors and actuators.

However, these schemas do not explicitly capture the interfaces and configurations of sensors and actuators, despite their strong coupling with the physical and control structures of the robot. Such details typically require additional configuration layers or external files for complete system specification.

The approach presented in this paper builds upon the HyMeKo framework [4], [7]—a hypergraph-based description language developed by the authors for modeling complex, multi-relational systems such as kinematic structures and communication networks. By extending HyMeKo, this work aims to provide a unified, semantically rich representation that bridges the gap between physical modeling, interface description, and simulator configuration.

III. Methodology

The proposed framework provides a unified representation for describing kinematic structures together with sensor–actuator interfaces and their associated control parameters. This work continues the project introduced in [8], extending it with a formal domain model built on the HyMeKo framework. The final purpose is to describe a two arm four-wheeled mobile manipulator, together with a proper communication graph definition and associated sensors (e.g., cameras, laser scanners) and flexibly generate to the target frameworks using simple programmatic queries (e.g., based in Python). Using HyMeKo's directed hypergraph formalism, the proposed description can express several relationships within a robotic configuration that traditional formats such as URDF cannot explicitly capture:

- connections between sensors and their corresponding information interfaces (e.g., ROS topics), including parametric attributes such as image resolution and their association with physical nodes (kinematic links),
- mappings between control interfaces, control parameters, and actuators,
- associations between communication-graph elements and their physical counterparts, and
- specification of crane-like or branched kinematic structures through fixation points defined on shared rigid bodies.

Overall, the HyMeKo-based description extends conventional robotic models by introducing the following additional elements:

- sensor configurations and their attachment to physical components,
- controller and actuator constraints, including control methods (e.g., position or velocity feedback),
- control-specific configuration parameters (e.g., controller tuning coefficients, refresh rate, monitor rate), and
- communication-graph associations (e.g., ROS 2 topics linked to sensors or controllers).

An excerpt of a six-degree-of-freedom (6-DOF) robotic arm description is shown in Listing 1. This code section defines the state publisher, control method (position or velocity), and control type (trajectory planner) together with the necessary configuration parameters. A complementary example in Listing 2 demonstrates the description of the sensor setup of a four-wheel robot, including camera parameters, spatial placement on the corresponding kinematic link, and the associated communication-graph topics.

Listing 1. Code snippet describing the robot control parameters and state publishers in a 6-DOF robotic arm

```
...
@sim_control_plugin: kinematics.sim_plugin {
    plugin
      "gz_ros2_control::GazeboSimROS2ControlPlugin",
    filename "gz_ros2_control-system",
    parameters "control.yaml"
}
@joint_state_broadcaster:
  kinematics.control_definition {
    - robot.anthropomorphic_arm_control,
    + sensors.joint_state_broadcaster
}
joint_trajectory_controller:
  controllers.joint_trajectory_controller {
   state_publish_rate 100.0
   action_monitor_rate 20.0
   @interfaces {
       + control_attributes.position,
       - control_attributes.position,
       - control_attributes.velocity
   }
   @joints {
        ->j0, ->j1, ->j2, ->j3,
        ->j4, ->jtool
   }
}
...
```

Listing 2. Code snippet describing a camera sensor assocaited with a 4-wheel robot

```
@camera_topic: communication.topic {
    - camera_image_topic_definition,
    - camera_info_topic_definition,
    + camera
}
@camera_connection:
  kinematics.sensors.sensor_connection {
    - camera,
    + camera_link
}
```

A. Generation of Description Files

The general process for generating configuration and description files from a single hypergraph-based model is illustrated in Figure 1. The framework transforms a unified robot description into multiple target formats (e.g., URDF, ROS 2, and Gazebo configuration files) through a sequence of well-defined steps:

1) Robot description: the base model contains the kinematic structure and the communication-graph mappings of the middleware system.
2) Design parameters: geometric and configuration parameters (e.g., dimensions, mass properties) that can be injected into the model during generation.
3) Hypergraph queries: a set of mapping rules that link elements of the hypergraph-based description to textual representations in the output files. For example, a query may map kinematic links to XML nodes in the generated URDF. These queries are typically implemented as Python code snippets executed in a visitor- or parse-tree fashion over the input model.
4) Hypergraph transformation: the queries and textual templates are combined to produce outputs conforming to the target schema:
   - URDF files provide the kinematic and physical description of the robot.
   - The Gazebo configuration and launch files define the setup for simulation and visualization.
   - The ROS 2 configuration files specify the controller parameters and initialize the communication graph for the initialization of the node.

This approach draws inspiration from the Model-Driven Software Development (MDSD) paradigm [9], where system specifications are defined at a higher level of abstraction and transformed into concrete implementation artifacts through automated model transformations.

IV. Results

Using the proposed framework, the individual components of an autonomous mobile-manipulator were successfully modeled and generated:

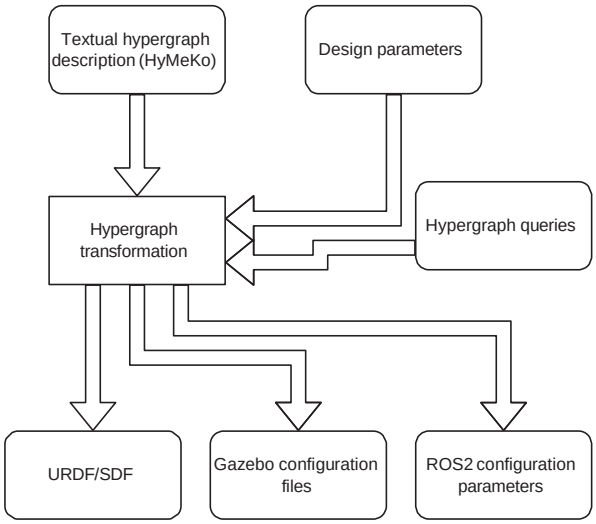

Fig. 1. Overview of the generation process. The framework produces simulation and configuration files from a unified hypergraph-based robot description and associated query templates.

- Robotic arms: two independent 6-DOF manipulators were defined (see Figure 2), each equipped with properly configured control interfaces based on the ROS 2 Control framework [10].
- Mobile base: a four-wheeled differential-drive platform was described, including sensor definitions and control interfaces required for navigation and motion control (see Figure 3).

All components were generated with complete configuration files, including:

- kinematic and transformation trees (for joint state publishers and TF management),
- controller interfaces (with update and monitor rates), and
- sensor configurations (e.g., refresh rates and ROS topic definitions).

Both systems are fully operational within the Gazebo environment and can be controlled via ROS 2 topics and custom control programs. The HyMeKo-based robot descriptions and source code of the language framework are publicly available at [1] . Due to space limitations, the full robot descriptions are not included in this paper.

V. Conclusion

This paper introduced a hypergraph-based methodology for generating unified robot description and configuration files compatible with Gazebo and ROS 2. The proposed framework integrates kinematic, control, and sensor–actuator interface descriptions within a single

[1]https://github.com/kyberszittya/himeko_lang

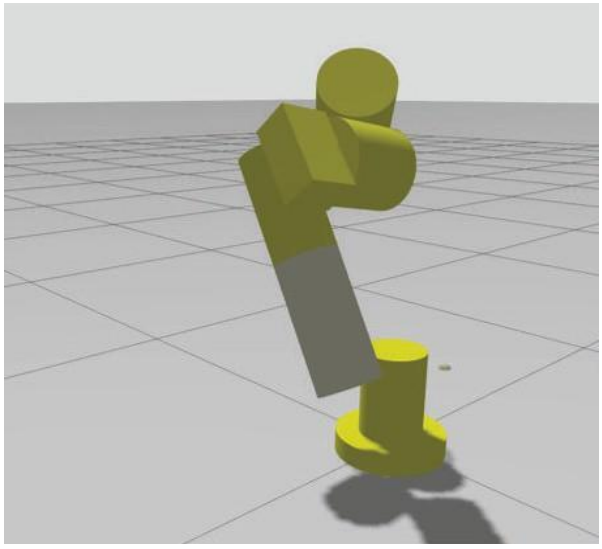

Fig. 2. 6-DOF robotic arm generated from the HyMeKo framework, running in Gazebo with ROS 2 control interfaces and defined communication graph.

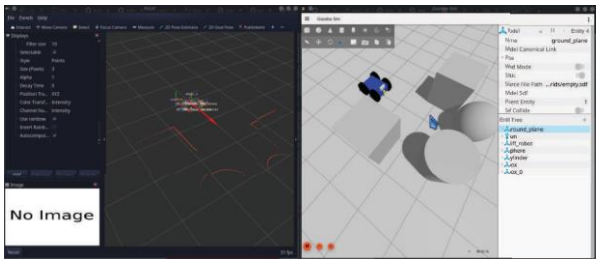

Fig. 3. Four-wheeled mobile robot in Gazebo and RViz, visualizing the configured sensors and real-time environment representation.

formal model, enabling the automatic generation of simulator-ready artifacts.

The approach was validated through the implementation of an autonomous mobile-manipulator, whose major subsystems—robotic arms, mobile base, and sensor configurations—were successfully simulated in Gazebo using ROS 2 control interfaces. While the current results demonstrate the feasibility and consistency of the framework, the complete robot integration and full simulation workflow remain under development.

Future work will focus on quantitative evaluation of the generated configurations, including timing performance, communication latency, and controller stability. In the long term, the framework aims to support the construction of a physical prototype, establishing a seamless bridge between simulation and real-world deployment and further advancing model-driven robot design.

References

[1] D. Tola and P. Corke, "Understanding URDF: A Dataset and Analysis," 2023, _eprint: 2308.00514. [Online]. Available: https://arxiv.org/abs/2308.00514

[2] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 5026–5033.

[3] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," in *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 2149–2154.

[4] C. Hajdu and N. Hegyi, "Modeling Kinematic and Dynamic Structures with Hypergraph-Based Formalism," *Applied Mechanics*, vol. 6, no. 4, 2025. [Online]. Available: https://www.mdpi.com/2673-3161/6/4/74

[5] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[6] E. Coumans and Y. Bai, *PyBullet, a Python module for physics simulation for games, robotics and machine learning*, 2016. [Online]. Available: http://pybullet.org

[7] C. Hajdu and B. Csapo´, "Hypergraph-based Modeling of Cognitive Dataflow Systems," in *IEEE CogInfoCom 2024: 2024 15th IEEE International Conference on Cognitive Infocommunications*, 2024, p. p. 37. [Online]. Available: https://m2.mtmt.hu/api/publication/35653972

[8] Z. Szila´gyi and C. Hajdu, "Design and Requirement Analysis of an Outdoor Autonomous Mobile Manipulator," in *AIS 2022 - 17th International Symposium on Applied Informatics and Related Areas - Proceedings*, 2022, pp. 44–49. [Online]. Available: https://m2.mtmt.hu/api/publication/33261175

[9] D. Varro´ and A. Balogh, "The Model Transformation Language of the VIATRA2 Framework," *Sci. Comput. Program.*, vol. 68, no. 3, pp. 214–234, Oct. 2007. [Online]. Available: https://doi.org/10.1016/j.scico.2007.05.004

[10] B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, "Timed-Elastic Bands for Manipulation Motion Planning," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 2019.