Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project Proposal

Rich Descriptive Models for Reusable ROS Software Components

Riddhesh Pradeep More

Supervised by

Prof. Sebastian Houben
Dr. Björn Kahl

June 2025

1 Introduction

1.1 Topic of This R&D Project

The Robot Operating System (ROS) is an open-source framework that forms the backbone of a vast portion of modern robotics development. It provides a modular architecture and a comprehensive ecosystem of reusable software packages used across domains such as industrial automation, mobile robotics, and increasingly in outdoor inspection and maintenance tasks. According to the 2024 ROS Metrics Report, the community now includes over 1,250 companies, more than 8,349 GitHub repositories tagged ROS, and an estimated 531 million package downloads in 2024 alone, with ROS 2 packages accounting for over 71.93% of downloads [10].

Despite this growth, a major bottleneck persists in the form of the lack of standardized semantic descriptions for ROS components. While ROS defines standards for message transfer and node communication, it lacks a formalized semantic layer to describe the function, compatibility, and constraints of software and hardware components. Tools like RosTooling aid in integration, but do not solve the challenge of "functional categorization"—identifying whether a package fulfills a specific task with acceptable quality [6].

One classical technique to deal with such a heterogeneity is Model-Driven Engineering (MDE). In MDE, a model offers an abstract representation of the real system, that abstracts away irrelevant details and focuses on the important information, depending on the purpose of the model. This abstraction allows a better understanding of the problem and the obtained model can be used to perform some reasoning and early validation, by means for instance of formal verification of properties. [1]

Semantic modeling in ROS involves formally describing what a software component is, does, and requires using structured, machine-interpretable models such as ontologies. Instead of relying on ad hoc documentation or naming conventions, semantic models encode explicit metadata about each component's functionality, interfaces, dependencies, and constraints. These models enable automated reasoning, allowing developers or tools to discover, compose, or validate components based on their meaning rather than just their syntax.[12].

This R&D project applies these techniques to define semantic descriptive models specifically for outdoor mobile robots used in inspection and maintenance. These models will capture both functional and non-functional properties such as input/output types, resource usage. The project will produce an ontology, a searchable component database, and a user interface frontend. Altogether, this structured approach aims to support intelligent request matching, enhance integration efficiency, and enable scalable robotic system design.

1.2 Relevance of This R&D Project

Although ROS provides powerful abstractions and a standardized communication protocol, it does not support semantic-level component reuse or discovery. This leads to several challenges:

- **Functional Categorization:** Difficulty in selecting components tailored to specific tasks like inspection and maintenance[6, 7].
- **Integration Complexity:** Ensuring compatibility across hardware and software remains a complex issue [5, 8].
- **Lack of Reusability:** The absence of structured descriptions hinders the scalability of robotic solutions across domains[2, 11].

The project aims to use semantic web technologies to extract, model, and query descriptive properties. This will support scalable and user-friendly integration of ROS packages for complex systems.

2 Related Work

2.1 Survey of Related Work

Several efforts have attempted to formalize aspects of ROS components using Model-Driven Engineering (MDE)—a methodology for using formal models in software development:

- Hammoudeh Garcia et al. [6], Hammoudeh García et al. [7]: The metamodels in this context are used to model ROS nodes, communication interfaces, and systems composed of subsystems . They provide a way to describe ROS systems with minimal models, capturing interaction patterns through external communication interfaces. These models aim to improve understanding of runtime behavior, ease interaction between ROS and other systems, and diffuse best practices for manually written code
- Zander et al. [12]:ReApp employs a model-driven approach for developing robotic components/applications. It uses ontologies as the glue that integrates different model and information types, ensuring data uniformity and allowing the description of data semantics in an application-independent way
- Casalaro et al. [3]: Conducted a systematic mapping of MDE in mobile robotics, revealing gaps in semantic representation.
- Trezzy et al. [11]: Analyzed modeling formalisms like UML and DSLs for ROS systems.
- Neto et al. [8]: Demonstrated how static analysis tools like HAROS improve the safety and maintainability of ROS components in industrial applications. HAROS works by performing static analysis to assess the quality of ROS repositories. It identifies potential issues such as code smells, security vulnerabilities, and performance bottlenecks, providing a general takeaway that static analysis can significantly enhance the reliability of ROS systems. ROSInfer builds upon tools such as HAROS.
- Dürschmid et al. [5]: Developed ROSInfer, a static analysis tool designed to infer behavioral models for ROS-based systems. These models capture

architecturally-relevant component behavior, detailing what causes a component to send messages, including triggers, state variables, and state transitions. ROSInfer addresses the challenge of ensuring correct component composition, as inconsistent compositions can lead to unexpected system behavior 5. The behavioral models inferred by ROSInfer can be used to find bugs that result from incorrect component composition and impact the software architecture's behavior.

2.2 Limitations and Deficits in the State of the Art

Despite the existence of standards for message transfer and communication in ROS, the following gaps persist:

- **Lack of Semantic Descriptions:** There is no standardized method for semantically describing the purpose, capabilities, and quality of ROS packages [7].
- **Underexplored Use of NLP:** While static analysis tools and metamodeling approaches exist, NLP and semantic search techniques remain underexplored [8, 12].
- **High Entry Barrier for MDE:** MDE is a powerful method, but its adoption is hindered by the complexity of tools and expertise required [3].
- **Lack of Package Safety Information:** Packages lack metadata related to safety compliance, e.g., with standards like IEC 61508.

3 Problem Statement

There is currently no standardized way of describing the purpose(functional properties), interface, or usage constraints of ROS software packages. This absence impedes reuse, makes safety assurance difficult, and hampers the rapid integration of components for new applications. Without semantic descriptions, developers rely heavily on manual exploration and documentation, which is often incomplete or outdated.

3.1 Assumptions and Constraints

Assumptions:

- Outdoor mobile robots used in inspection/maintenance represent typical ROS use cases.
- The developed semantic models will follow ROS naming and structural conventions.

Constraints:

- Focus is on semantic modeling and component description, not runtime optimization.

3.2 Tasks

3.2.1 System Analysis:

- Examine component configurations from outdoor robotic platforms used in inspection and maintenance.
- Identify functional groupings (e.g., SLAM, path planning, actuation) and their usage context.

3.2.2 Ontology and Taxonomy Design:

- Design a taxonomy of component classes (e.g., LocalizationNode, SensorDriver).
- Define functional and non-functional properties, for example message types, update rates, and compatibility etc
- Select an appropriate representation format (e.g., OWL, RDF).

3.2.3 Model Implementation and Integration:

- Build a semantic model repository using Blazegraph or an equivalent RDF store.

- Create a UI with NiceGUI for querying and visualization.
- Ensure models are modular and standalone (ontology, UI, database).

3.2.4 Validation and Documentation:

- Populate the model with 25% (goal: 75%) of components extracted from case studies.
- Validate correctness and expressiveness of semantic descriptions.
- Produce documentation to enable future extension and reuse in other domains.

3.3 Key Challenges

- **Taxonomy Design:** Finding a balance between specificity and generality for component categorization.
- **Scalability:** Ensuring performance and relevance as more components and properties are added.
- **Usability:** Making tools intuitive for developers unfamiliar with semantic web technologies.
- **Safety Integration:** Incorporating safety-critical metadata, referencing standards like IEC 61508.

4 Project Plan

4.1 Work Packages

The project is divided into six Work Packages (WPs) with specific objectives, tasks, and deliverables.

4.1.1 WP1: Literature Study

Objective:
Understand existing frameworks and best practices for descriptive models in ROS/ROS2 software.

Tasks:

- Study Model-Driven Engineering (MDE) and semantic modeling techniques.
- Identify gaps in the current descriptive modeling of ROS components.

Deliverables:

- Comprehensive literature review highlighting challenges and opportunities.
- Preliminary framework outlining semantic modeling approaches.

4.1.2 WP2: Component Analysis and Classification

Objective:
Classify and derive attributes for ROS components.

Tasks:

- Collaborate with robotic teams to gather data about existing ROS/ROS2 packages.
- Develop a taxonomy to classify ROS components and their attributes systematically.
- Apply MDE principles to ensure consistent and reusable component classification.

Deliverables:

- Structured hierarchy and taxonomy of ROS components.
- Defined attributes for ROS components aligned with semantic modeling principles.

4.1.3 WP3: Semantic Model Design

Objective:
Design a semantic model to represent ROS components effectively.

Tasks:

- Design a meta-model for defining component representation.
- Select a representation method suitable for scalability and cross-domain usage.
- Document the model's functional and integration aspects.

Deliverables:

- Detailed meta-model specification for component classes and attributes.
- Documentation of functional and integration aspects of the model.

4.1.4 WP4: Proof-of-Concept Implementation

Objective:
Implement the semantic model and gather feedback for refinement.

Tasks:

- Develop a database schema for ROS component descriptions.
- Build a user interface prototype using a framework like NiceGUI.
- Populate the database with at least 25% of identified components.
- Present the prototype to the team for feedback and refinement.

Deliverables:

- Functional database and user interface prototype.
- Initial population of the database with a subset of ROS components.

4.1.5 WP5: Evaluation and Testing

Objective:
Refine the semantic model and tools based on user feedback.

Tasks:

- Analyze user feedback to improve the model and UI prototype.
- Conduct further testing to ensure robustness and usability.

Deliverables:

- Enhanced semantic model and prototype with documented improvements.

4.1.6 WP6: Project Report

Objective:
Summarize project outcomes and provide recommendations for future work.

Tasks:

- Compile findings on MDE, semantic modeling, and taxonomy design.
- Provide recommendations for scaling the framework to other domains of robotics and automation.

Deliverables:

- Final project report summarizing all findings and outcomes.
- Supporting documentation and recommendations for future work.

4.2 Contingency Plan

To ensure uninterrupted progress in case of delays or lack of external input, the project includes fallback strategies:

1. Fallback Data Sources

- **Risk:** No response or delayed input from IAM4RAIL partners or collaborating robotics teams.
- **Action:** Use publicly available ROS/ROS2 packages from GitHub, ROS Index, and ROS Wiki.
- **Tools:** Extract attributes from `package.xml`, `README`, and `launch` files; apply static code analysis tools where necessary.

2. Independent Progress on Modeling

- Proceed with classification and taxonomy design using well-known open-source packages in navigation, perception, and manipulation domains.
- Ensure the ontology remains general enough to accommodate future industrial datasets.

3. Community-Based Validation

- **Fallback Review:** Seek feedback from peers, researchers, and the online ROS community.
- **Online Sources:** Use ROS Answers, GitHub issues, and user forums for validation insights.

4. AI/NLP-Based Metadata Generation

- Implement basic NLP (e.g., keyword extraction, topic modeling) on documentation to auto-generate semantic tags.
- Use this metadata to populate the semantic model and test UI search functionalities.

5. Scalable, Modular Architecture

- Maintain modular separation of ontology, UI, and database layers.
- Enable future integration with industry-specific data with minimal refactoring.

4.3 Milestones

M1 Completion of literature review and identification of best practices.

M2 Development of initial semantic models and taxonomy.

M3 Implementation of the database

M4 Validation through case studies.

M5 Submission of the final project report.

4.4 Project Schedule

Task	Status	Start date	End date
WP1: Literature Study	In progress	6/9/2025	6/30/2025
WP2: Component Analysis	In progress	6/1/2025	7/31/2025
WP3: Semantic Model Design	Not started	7/1/2025	8/31/2025
WP4: Proof-of-Concept Implementation	Not started	7/1/2025	9/30/2025
WP5: Evaluation and Tesing	Not started	10/1/2025	11/20/2025
WP6: Project Report	Not started	10/15/2025	12/9/2025

Figure 1: Project timeline.

4.5 Deliverables

Minimal Viable

- List of components with structured textual descriptions.
- Hierarchy of component classes with textual descriptions for each class.

- Definitions of common component attributes.
- Suggestion of a semantic representation framework.
- Implementation of component classes and textual descriptions for at least 25% of identified components within the selected semantic framework.

Expected

- All deliverables from the Minimal Viable category.
- Well-documented semantic models and model design.
- Models (component classes and instances) designed for expansion into other domains or for describing ROS components not covered by the demonstrators.
- Implementation of descriptions for at least 75% of the identified components within the selected semantic framework.
- Proof-of-concept implementation of a search interface and search functionality in a user-friendly UI framework (e.g., NiceGUI or a similar framework agreed upon with the supervisor and IAM4RAIL project partners).

Maximal

- All deliverables from the Minimal and Expected categories.
- User evaluation of semantic models and the database UI with IAM4RAIL project users.
- Recommendations on how to incorporate AI/NLP techniques to enhance the usability of the component database for intuitive and efficient searches.

References

[1] Davide Brugali. Model-driven software engineering in robotics: Models are designed to use the relevant things, thereby reducing the complexity and cost in the field of robotics. *IEEE Robotics Automation Magazine*, 22(3):155–166, 2015. doi: 10.1109/MRA.2015.2452201.

[2] Herman Bruyninckx, Markus Klotzbücher, Nico Hochgeschwender, Gerhard Kraetzschmar, Luca Gherardi, and Davide Brugali. The brics component model: a model-based development paradigm for complex robotics software systems. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, page 1758–1764, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450316569. doi: 10.1145/2480362.2480693. URL https://doi.org/10.1145/2480362.2480693.

[3] Cattivera G. Ciccozzi Casalaro, G.L. et al. Model-driven engineering for mobile robotic systems: A systematic mapping study. *Software and Systems Modeling*, 21:19–49, 2022.

[4] Kolovos D. de Lara J. Di Ruscio, D. et al. Low-code development and model-driven engineering: Two sides of the same coin? *Software and Systems Modeling*, 21:437–446, 2022.

[5] Tobias Dürschmid, Christopher Steven Timperley, David Garlan, and Claire Le Goues. Rosinfer: Statically inferring behavioral component models for ros-based robotics systems. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400702174. doi: 10.1145/3597503.3639206. URL https://doi.org/10.1145/3597503.3639206.

[6] Nadia Hammoudeh Garcia, Mathias Lüdtke, Sitar Kortik, Björn Kahl, and Mirko Bordignon. Bootstrapping mde development from ros manual code -part 1: Metamodeling. 02 2019. doi: 10.1109/IRC.2019.00060.

[7] Deshpande H. Santos A. Hammoudeh García, N. et al. Bootstrapping mde development from ros manual code: Part 2—model generation and leveraging

models at runtime. *Software and Systems Modeling*, 20:2047–2070, 2021. doi: 10.1007/s10270-021-00873-2.

[8] Tiago Neto, Rafael Arrais, Armando Sousa, André Santos, and Germano Veiga. *Applying Software Static Analysis to ROS: The Case Study of the FASTEN European Project*, pages 632–644. 01 2020. ISBN 978-3-030-35989-8. doi: 10.1007/978-3-030-35990-4_51.

[9] IAM4RAIL Project. Iam4rail: European railway research project, 2025. URL `https://projects.rail-research.europa.eu/eurail-fp3/`. Accessed: January 2025.

[10] Katherine Scott and Tully Foote. 2024 ros metrics report. `https://www.ros.org/news/2025/01/2024-ros-metrics-report.html`, 2025. URL `https://www.ros.org/news/2025/01/2024-ros-metrics-report.html`. Open Robotics Foundation.

[11] Mickael Trezzy, Ileana Ober, Iulian Ober, and Raquel Oliveira. Applying mde to ros systems: A comparative analysis. *Scientific Annals of Computer Science*, 31:111–144, 08 2021. doi: 10.7561/SACS.2021.1.111.

[12] Stefan Zander, Georg Heppner, Georg Neugschwandtner, Ramez Awad, Marc Essinger, and Nadia Ahmed. A model-driven engineering approach for ros using ontological semantics, 2016. URL `https://arxiv.org/abs/1601.03998`.