

iAm.md: Robot Skill Self-Assessment through Agentic Introspection for Unknown Open-Vocabulary Domains

Vincenzo Guarino^{1,*}, Emanuele Musumeci¹, Vincenzo Suriani¹ and Daniele Nardi¹

¹*Department of Computer, Control and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Via Ariosto 25, 00185 Rome, Italy*

Abstract

Agentic AI based on Large Language Model generalization capabilities offers a wide range of potential applications, including planning for embodied tasks. For example, embodied agents based on Foundation models can generate plausible plans in autonomous robotics scenarios. Due to limited context windows or hallucinatory phenomena in the next-token prediction formulation, behaviors may be generated without establishing whether the deployed robot and the observed environment actually support the requested operation, in what we call a "grounding failure". Thanks to the recent improvements in reasoning capabilities of foundation models, autonomous robot behavior generation problem can be formulated as a code generation problem. We present iAm.md, a Markdown standard and generation framework, that allows anchoring this process in complementary forms of deployment evidence. Through open-vocabulary semantic mapping, we combine local vision-language detections and object segmentation and refer them to persistent object records in this intermediate standardized representation, allowing agentic introspection. We then study this new technique on a simulated TIAGo, on navigation-and-manipulation tasks, showing how this standardized representation jointly supports skill self-assessment and executable task generalization.

Keywords

Robot skill self-assessment, agentic introspection, open-vocabulary semantic mapping, foundation-model agents, robot code generation, ROS 2

1. Introduction

Translating a Natural Language (NL) task request into physical robot behavior typically requires a closed representation of the task and operating environment: neuro-symbolic planning makes plan validity conditional on the quality of a provided abstraction of the world state and actions [1, 2, 3]. As robots move toward open environments, prompt engineering is an increasingly important step of plan and executable code generation [4, 5], but this capacity creates a grounding problem: a language model may complete an underspecified request with synthetic knowledge even when the necessary evidence has not been observed [6], and generated action sequences can violate the preconditions of the actions they invoke [7]. Once connected to robot tools, an unsupported assumption becomes a physical action, which can lead to obvious safety and ethical issues.

In this way, a generated program may refer to an unobserved object, or it may call for an operation that the deployed platform does not support. Scene grounding alone does not resolve the second problem, since a plan reasonable for one robot may fail on another. Robot self-assessment frames the missing judgment as the ability to "predict, estimate, or measure how well it can perform a task in a given context and environment" [8]. We investigate to what extent an embodied foundation model agent operating in an unknown open-vocabulary domain can use persistent scene memory and a structured robot self-description to perform *skill self-assessment* and produce executable plans. Here *agentic introspection* means the agent is capable of inspecting the representations available inside its own planning process before committing to generated code. Our contributions are: (i) an open-vocabulary

13th Italian Workshop on Artificial Intelligence and Robotics (AIRO 2026)

*Corresponding author.

✉ vincenzo.guarino@uniroma1.it (V. Guarino); musumeci@diag.uniroma1.it (E. Musumeci); suriani@diag.uniroma1.it (V. Suriani); nardi@diag.uniroma1.it (D. Nardi)

🆔 0009-0006-6794-6767 (V. Guarino); 0009-0004-2359-5032 (E. Musumeci); 0000-0003-1199-8358 (V. Suriani); 0000-0001-6606-200X (D. Nardi)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

semantic memory that uses local vision-language inference and segmentation to provide persistent object identities and locations; (ii) `iAm.md`¹, a structured Markdown standard and generation framework representing specific robot information, resolved interfaces, and explicit unknowns; and (iii) an introspective embodied agent that combines semantic-map queries, introspection, task code generation, and robot execution for NL requests.

2. Related Work

Hierarchical 3D scene graphs (Kimera [9], Hydra [10]) and dense vision-language feature maps (VLMs [11], ConceptFusion [12]) accumulate persistent spatial structure but do not maintain robot-specific execution evidence. Language-conditioned code-generation methods (SayCan [13], Code as Policies [14], ProgPrompt [15]) compose developer-provided APIs but assume feasible instructions and pre-authored platform documentation; feasibility-assessment methods (PlanBench [7], CodeBotler [16], KnowNo [17]) require a complete action model rarely available for a configured robot. Robot knowledge frameworks [18, 19, 20] relate tasks to capabilities through data entered before an episode. Closest to our work, EMPOWER-ADF [21] grounds task entities against a NL robot description but selects from a fixed action vocabulary without physical execution. We instead derive the robot description from the deployment itself and use it, with live interface evidence, to *write* task-specific code.

3. Methodology

An embodied code generation system places a tool-calling foundation-model agent between a task request and the deployed robot. Persistent robot facts enter the agent context through `iAm.md`, whereas the current pose and camera observation describe the state in which the request must be executed. One general command tool connects this context to the live ROS 2 environment [22], allowing the agent to observe the execution state, write executable programs, and run them in the deployment that supplied the supporting evidence. The semantic map supplies object identities and spatial support through ROS 2 topics and a serialized snapshot, so generated programs resolve task targets without placing transient observations in the persistent robot description. Figure 1 summarizes these evidence paths and the execution loop that connects the generated program to the deployed robot.

Open-vocabulary semantic memory. A locally served vision-language model (Qwen3.6-35B via llama.cpp) proposes whole-object detections with open-vocabulary labels and bounding boxes; after non-maximum suppression, SAM 2 [23] produces a binary mask for each surviving proposal. Each validated mask is reprojected through depth into a map-frame point cloud, cleaned by morphological closing, foreground filtering, and DBSCAN clustering [24], and voxelized into a sparse `VoxelBlockGrid` [25] whose active voxels carry object identifiers. Association is decided by a voxel-overlap score between the new observation A and each stored support B ,

$$s(A, B) = \max\left(\frac{|A \cap B|}{|A|}, \frac{|A \cap B|}{|B|}\right), \quad (1)$$

so an observation below threshold for every candidate becomes a new object, a single match updates the stored record, and multiple matches trigger a merge. Stored voxels yield centroids, bounds, and oriented bounding boxes exported as a JSON snapshot and RViz markers that downstream programs query to resolve NL references, including entities outside the current camera view.

¹Project page: <https://yurimachine.github.io/iAm.md/>.

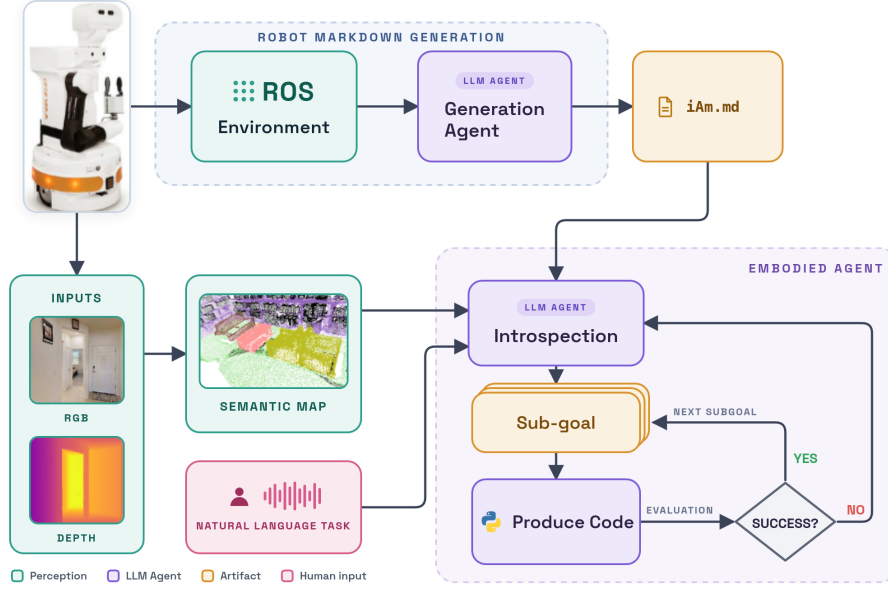


Figure 1: System architecture of the robot-description generation and embodied-agent execution stages. The generation agent extracts resolved robot evidence from the ROS 2 environment and records it in `iAm.md`. During task execution, the embodied agent combines this description with the semantic map and NL request, produces code for successive subgoals, and uses execution outcomes for introspective revision.

iAm.md: embodiment evidence. `iAm.md` is a standard for describing the persistent embodiment of a configured robot deployment in plain text that a foundation-model agent can read in context. A conforming document follows a fixed schema from identity and embodiment summary, through the URDF-derived physical structure, actuation, and sensing, to the higher-level software functions exposed by the robot stack and the unknowns that remain after generation. Every block carries explicit cardinality, a value that cannot be defensibly established is recorded as *unknown* in place, and software-function records separate the *acceptance* criteria of a command interface from the physical *outcome* of the commanded operation. An instance is generated once per deployment from the URDF, augmented with the live ROS 2 graph and configuration artifacts for otherwise-unobservable interfaces. The TIAGo [26] instance used here records its kinematic structure, navigation and gripper envelopes, controller bindings, and Navigation2 [27]/MoveIt function records. The acceptance/outcome separation appears concretely in the gripper records: a successful gripper trajectory result establishes only that controller tolerances were met, and grasp retention is listed among the unknowns because the deployed configuration includes no sensor that could establish it.

Agentic introspection and robot code execution. Each NL request is one execution episode. The startup prompt defines the command-tool protocol and embeds the generated `iAm.md` between explicit delimiters. The agent inspects the ROS 2 graph and interface schemas through the same command tool it uses to act, so interface availability is established from the deployed system during the episode. Skill self-assessment runs inside the agent loop: the agent relates the request to observed objects, then checks the robot description and live interfaces needed to realize the operation/navigation needs the object location and navigation interface, whereas manipulation additionally requires the manipulator and gripper contracts. When evidence supports the operation the agent proceeds to observation, program construction, or a physical command; otherwise a structured outcome records a blocked result. Introspection continues during execution, because observed behavior can expose an unsupported assumption and prompt a revision of the program.

4. Experimental Results

4.1. Semantic-Map Evaluation

Twelve objects in a simulated scene were independently annotated as labelled spheres in the map frame through an interactive RViz interface. An estimated record is compatible with an annotation when its centroid lies inside the sphere and its label matches; a maximum one-to-one bipartite matching over compatible pairs yields object-level *Found* (fraction of annotations matched) and *Correct* (fraction of estimates matched) scores, following Hydra [10]. The matching selected 12 pairs from 12 annotations and 13 estimated records: every annotated object was recovered ($\text{Found} = 100\%$), and the single unmatched record was a duplicate estimate ($\text{Correct} = 92.31\%$), localizing the observed error to duplicate consolidation during map updates and merges.

4.2. Agent-Produced Tool Evaluation

The agent evaluation isolates the contribution of `iAm.md` by varying only the persistent robot knowledge available during tool construction. Both conditions run the same embodied coding agent, a DeepAgents harness driving `gpt-5.6-sol` at medium reasoning effort, on a simulated single-arm TIAGo, and both receive the same fixed request together with a distilled NL specification of the navigation-and-manipulation tool to be built. The specification was distilled from a reference implementation developed with Claude Fable 5 on the same platform, with every deployment fact removed, so the agent must recover the robot connection itself, and `iAm.md` is placed only in the documented workspace. A phase ends when the submitted tool passes independent verification (a simulator reset, unattended execution, and a static audit rejecting tools that do not use robot depth data), or when the agent reaches a 110-minute budget after at most four attempts.

Phase 1: construction. For “*grasp a drink can and place it on a sofa*”, the documented agent produced a tool that passed verification after a simulator reset in 50.4 minutes, using 103 model calls and 8.50 million input tokens. The agent without `iAm.md` reached the 110-minute budget (186 model calls, 20.32 million input tokens) without a verified grasp or a completed task (Table 1A). The gap appeared at the earliest milestones: the documented agent wrote a first tool version after 4.8 minutes and reached a verified grasp approach after 7.8 minutes, against 8.2 and 26.4 minutes without the description, and it first verified a grasp after 25.7 minutes while the other agent obtained none within budget (4/10 versus 0/19 successful grasp attempts). Lacking the recorded embodiment, the undocumented agent calibrated the gripper through execution and incurred 13 initial inverse-kinematics failures, recovering constraints that `iAm.md` supplies directly; it also made 23 live-discovery calls, against 14 discovery calls and five `iAm.md` reads for the documented agent.

Phase 2: reuse. Retaining the documented agent’s state and tool, the request changed to *move a cube to the opposite side of a table*, a different object geometry and placement relation. The agent raised the torso, verified clearance, and executed the verified joint configuration, avoiding an elbow-wrapping IK switch during retreat. To represent *the other side*, it extended the tool interface with a map-frame approach direction rather than a cube-specific behavior. The plan passed its first verification after a simulator reset. Reuse required only three additional discovery calls and no further `iAm.md` reads (Table 1B); the changed geometry nonetheless propagated into grasp planning and placement, requiring 49 tool edits and 34 reported step failures. Retaining the command vocabulary while extending the generic kinematic and directional mechanisms constitutes few-shot generalization.

4.3. Discussion

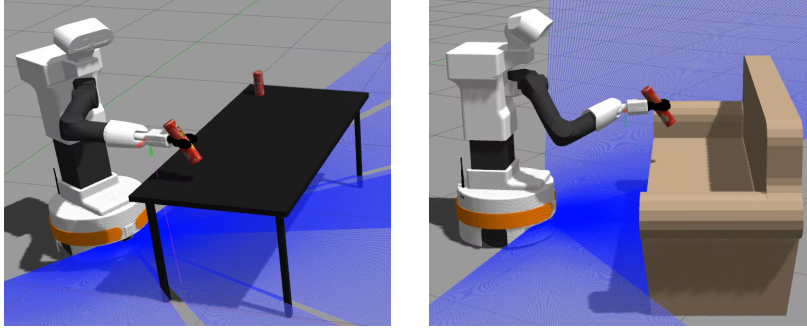
The two evaluations identify complementary roles for persistent and live evidence during skill self-assessment. The semantic map supplies the object referents and spatial estimates needed to interpret a request; `iAm.md` relates those estimates to the configured robot; and the resulting assessment is

Table 1

(A) Phase 1 construction with and without `iAm.md`; the condition without the document reports the effort before the 110-minute budget was reached. (B) The documented agent across construction (P1) and reuse (P2).

Table 1A			
Metric	With	Without	Ratio
Verified task completion	Yes	No (budget)	-
Time to verified compl. (min)	50.4	>109.9	>2.2×
Model calls	103	186	1.8×
Input tokens (million)	8.50	20.32	2.4×
Tool edits	33	47	1.4×
Reported step failures	22	48	2.2×
Successful grasp attempts	4/10	0/19	-

Table 1B		
Metric	P1	P2
Verified completion	Yes	Yes
Time to verified (min)	50.4	94.5
Model calls	103	150
Input tokens (M)	8.50	16.90
Tool edits	33	49
Grasp attempts	4/10	8/13
Discovery calls	14	3
<code>iAm.md</code> reads	5	0

**Figure 2**

Simulated TIAGo executing the agent-produced tool: picking up the can (left) and placing it on the sofa (right).

expressed through executable checks that change when execution exposes an unsupported assumption. Because checks derived from failures remained part of the tool and were reused when the request changed, self-assessment emerged as a feedback loop connecting task interpretation, code generation, execution evidence, and revision.

5. Conclusions and Future Work

We showed that an embodied foundation-model agent can combine persistent scene memory, a robot self-description, and live ROS 2 observations to assess and realize NL tasks in an open-vocabulary domain. The semantic map recovered all twelve annotated objects with one duplicate record. In a controlled comparison, access to `iAm.md` was associated with lower construction cost and verified completion: the documented agent produced a verified tool in 50.4 minutes, versus the 110-minute budget without a verified grasp for the undocumented agent, expending roughly twice the calls and tokens. Retaining that tool, the agent extended it to a second task with different object geometry without rereading `iAm.md`-evidence of few-shot generalization, from a single simulated run. Future work will evaluate `iAm.md` generation across embodiments and add an independent safety boundary validating trajectories before actuation on physical robots.

Declaration on Generative AI

During the preparation of this work, the authors used a large language model (LLM)-based assistant to proofread and optimize the phrasing of parts of this manuscript, including language editing, sentence-level optimization, and ambiguity resolution. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

- [1] R. E. Fikes, N. J. Nilsson, STRIPS: A new approach to the application of theorem proving to problem solving, *Artificial Intelligence* 2 (1971) 189–208. doi:10.1016/0004-3702(71)90010-5.
- [2] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, D. Wilkins, PDDL—the Planning Domain Definition Language, Technical Report, AIPS-98 Planning Competition Committee, 1998.
- [3] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, T. Lozano-Pérez, Integrated task and motion planning, *Annual Review of Control, Robotics, and Autonomous Systems* 4 (2021) 265–293. doi:10.1146/annurev-control-091420-084139.
- [4] T. B. Brown, et al., Language models are few-shot learners, *Advances in Neural Information Processing Systems* 33 (2020) 1877–1901.
- [5] M. Chen, et al., Evaluating large language models trained on code, *arXiv preprint arXiv:2107.03374* (2021).
- [6] Z. Ji, et al., Survey of hallucination in natural language generation, *ACM Computing Surveys* 55 (2023) 1–38. doi:10.1145/3571730.
- [7] K. Valmeekam, M. Marquez, A. Olmo, S. Sreedharan, S. Kambhampati, PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change, in: *Advances in Neural Information Processing Systems*, volume 36, 2023.
- [8] A. Norton, et al., Metrics for robot proficiency self-assessment and communication of proficiency in human-robot teams, *ACM Transactions on Human-Robot Interaction* 11 (2022) 1–38. doi:10.1145/3522579.
- [9] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, L. Carlone, Kimera: From SLAM to spatial perception with 3D dynamic scene graphs, *The International Journal of Robotics Research* 40 (2021) 1510–1546.
- [10] N. Hughes, Y. Chang, L. Carlone, Hydra: A real-time spatial perception system for 3D scene graph construction and optimization, in: *Robotics: Science and Systems (RSS)*, 2022.
- [11] C. Huang, O. Mees, A. Zeng, W. Burgard, Visual language maps for robot navigation, in: *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 10608–10615. doi:10.1109/ICRA48891.2023.10160969.
- [12] K. M. Jatavallabhula, et al., ConceptFusion: Open-set multimodal 3D mapping, in: *Robotics: Science and Systems (RSS)*, 2023.
- [13] M. Ahn, et al., Do as I can, not as I say: Grounding language in robotic affordances, *arXiv preprint arXiv:2204.01691* (2022).
- [14] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, A. Zeng, Code as policies: Language model programs for embodied control, in: *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 9493–9500. doi:10.1109/ICRA48891.2023.10160591.
- [15] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, A. Garg, ProgPrompt: Program generation for situated robot task planning using large language models, *Autonomous Robots* 47 (2023) 999–1012. doi:10.1007/s10514-023-10135-3.
- [16] Z. Hu, F. Lucchetti, C. Schlesinger, Y. Saxena, A. Freeman, S. Modak, A. Guha, J. Biswas, Deploying and evaluating LLMs to program service mobile robots, *IEEE Robotics and Automation Letters* 9 (2024) 2853–2860. doi:10.1109/LRA.2024.3360020.
- [17] A. Z. Ren, et al., Robots that ask for help: Uncertainty alignment for large language model planners, in: *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.
- [18] M. Tenorth, M. Beetz, Representations for robot knowledge in the KnowRob framework, *Artificial Intelligence* 247 (2017) 151–169. doi:10.1016/j.artint.2015.05.010.
- [19] D. Beßler, R. Porzel, M. Pomarlan, A. Vyas, S. Höffner, M. Beetz, R. Malaka, J. Bateman, Foundations of the socio-physical model of activities (SOMA) for autonomous robotic agents, in: *Formal Ontology in Information Systems*, IOS Press, 2021. doi:10.3233/FAIA210379.
- [20] G. R. Silva, J. Päßler, S. L. T. Tarifa, E. B. Johnsen, C. H. Corbato, ROSA: A knowledge-based solution for robot self-adaptation, *Frontiers in Robotics and AI* 12 (2025) 1531743. doi:10.3389/

frobt.2025.1531743.

- [21] V. Suriani, M. Brienza, F. Argenziano, D. Nardi, D. D. Bloisi, Modeling self-awareness in embodied task planning with LLM-driven heuristics, *Big Data and Cognitive Computing* 10 (2026) 176. doi:10.3390/bdcc10060176.
- [22] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, Robot operating system 2: Design, architecture, and uses in the wild, *Science Robotics* 7 (2022) eabm6074. doi:10.1126/scirobotics.abm6074.
- [23] N. Ravi, et al., SAM 2: Segment anything in images and videos, *arXiv preprint arXiv:2408.00714* (2024).
- [24] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, AAAI Press, 1996, pp. 226–231.
- [25] Q.-Y. Zhou, J. Park, V. Koltun, Open3D: A modern library for 3D data processing, *arXiv preprint arXiv:1801.09847* (2018).
- [26] J. Pagès, L. Marchionni, F. Ferro, TIAGo: The modular robot that adapts to different research needs, in: *International Workshop on Robot Modularity, IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.
- [27] S. Macenski, F. Martín, R. White, J. G. Clavero, The marathon 2: A navigation system, in: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. doi:10.1109/IROS45743.2020.9341207.