Advancing End-User Development in Robotics with Large Language Models and Digital Twins

Luigi Gargioni[1]

[1]University of Brescia - Department of Information Engineering, Via Branze 38, Brescia, 25123, Italy

Abstract

This paper introduces an end-user development environment designed to support non-expert users in creating pick-and-place robot tasks. The environment includes a chat-based interface powered by OpenAI's GPT-4o, guiding users through the task definition process in an intuitive and domain-oriented manner. Once a task is defined, it is visualized in a graphic interface built with Google Blockly, enabling users to review, edit, or expand the task through direct manipulation of block-like components. The system also allows users to generate a corresponding robot program and simulate its execution using a digital twin of the robot. This final simulation step is essential for validating the task and identifying potential issues before deploying the robot program in a real-world setting.

Keywords

Human-Machine Interaction, End-User Development, Human-Robot Collaboration, Collaborative Robots, Digital Twin

1. Introduction and related works

The integration of collaborative robots (cobots) into human-centered work environments has gained significant traction in recent years. Cobots are increasingly employed to support a variety of complex and repetitive tasks, thereby enhancing overall system flexibility and productivity [1, 2]. Their adoption helps strike a balance between automation and flexibility, a crucial requirement for mass customization and small-batch production [3, 4]. Cobots are designed to operate safely alongside humans, often handling precise or physically demanding operations that either precede or follow human interventions. Typically constructed with lightweight materials to reduce injury risk during accidental contact [2], cobots can also be easily moved between workstations to support location-specific tasks [1]. As a result, they are considered a key technology for enhancing both production efficiency and the overall quality of working conditions [5, 4].

To fully leverage the flexibility of cobots, it is essential that human workers can program new tasks easily and quickly. However, these workers are usually not experts in programming or robotics. Consequently, a major challenge in this field is to design methods and develop software environments that empower end users to create robot programs independently [4, 2, 6].

This work presents an End-User Development (EUD) [7, 8, 9] environment that supports both programming and testing through intuitive interaction modalities. For the programming phase, users can define robot tasks using natural language through a chat-based interface that integrates OpenAI GPT-4o. In addition to confirming the task through conversational interaction, the environment also provides a visual representation of the generated robot program using Google Blockly [10]. Blockly was chosen for its proven effectiveness in promoting end-user programming [11, 12]. The visual interface allows users to inspect and incrementally refine the tasks by manipulating graphical blocks, thereby supporting the progressive enhancement of task complexity. Unlike other approaches, users in our system are not required to build the Blockly program from scratch. Instead, Blockly serves as a visual medium for inspecting, correcting, and modifying the generated code via direct manipulation of the blocks.

IS-EUD 2025: 10th International Symposium on End-User Development, 16-18 June 2025, Munich, Germany
luigi.gargioni@unibs.it (L. Gargioni)
0000-0003-4354-916X (L. Gargioni)

Regarding testing, the environment incorporates a digital twin [13] of the robot, enabling users to simulate and validate the robot's behavior before deploying the program in a physical setting. The digital twin offers the possibility of executing the programmed tasks on a precise representation of the robot, facilitating the early detection of potential execution issues prior to deployment on the physical system. This simulation capability is powered by Gazebo[1], an open-source robot simulator compatible with the Robot Operating System (ROS)[2], allowing seamless integration with standard ROS-based development workflows.

As a case study, the work focuses on the specification of pick-and-place tasks. Within the EUD environment, users can first define the items relevant to their specific application domain, such as objects, actions, and locations, and then compose tasks that may include repetitions, conditional loops, and branching behaviors. This work builds upon the hybrid approach to user-centered programming described in [14], but introduces several enhancements. These include the use of GPT-4o for advanced natural language understanding, a more standardized and expressive block-based interface for visualizing and editing robot programs, and the integration of a digital twin, extending the system presented in [15], to enable sandbox-style testing and validation of the robot tasks.

For the purpose of this project, the COBOTTA robot [3], developed by DENSO WAVE Ltd, was selected. COBOTTA is a compact, lightweight, and portable collaborative robot designed with safety and ease of use in mind, particularly in environments involving human operators. The robot features a six-axis articulated arm and is equipped, in its standard configuration, with an integrated camera and an electric gripper, making it particularly suitable for pick-and-place applications.

The programming paradigm presented in this work is expected to significantly improve the accessibility and usability of COBOTTA for non-technical users. By lowering the barriers to robotic programming, this approach aims to support a diverse range of task scenarios executable by COBOTTA or comparable collaborative robots. It is important to note, however, that the current prototype supports only a subset of COBOTTA capabilities. Specifically, the version used in the experiments does not allow remote operation or integration with additional equipment beyond the standard configuration comprising the camera and electric gripper.

2. Core components of the EUD environment

The EUD environment has been developed as a web-based platform aimed at simplifying the creation of robot tasks through an intuitive interface. Its implementation adopts widely used web and scripting technologies to ensure compatibility, ease of deployment, and integration with existing tools. Additional technical details regarding the system architecture are provided in [15]. To support the modeling and interaction with physical systems, a digital twin has been incorporated into the workflow using a ROS-based architecture.

To clarify the role of each component, a realistic running example is used throughout this section: let us imagine a technician who needs to automate a routine lab procedure to process a series of test tubes; the technician can use a cobot to perform a shaking operation on each test tube and place it into a container; however, the cobot can initiate the task only after the technician physically touches it, using this interaction as a signal to begin. This scenario is used in the following to illustrate the three primary components of the EUD environment, namely the chat interface, the graphic interface, and the digital twin, which can be used by the technician to define, validate, and simulate the robot tasks.

2.1. The chat interface

The main function of the chat interface is to support users in defining new pick-and-place tasks through natural language interaction. Users can communicate with the system either via text or voice.

[1] https://gazebosim.org/home
[2] https://www.ros.org/
[3] https://www.densorobotics-europe.com/product-overview/products/collaborative-robots/cobotta/

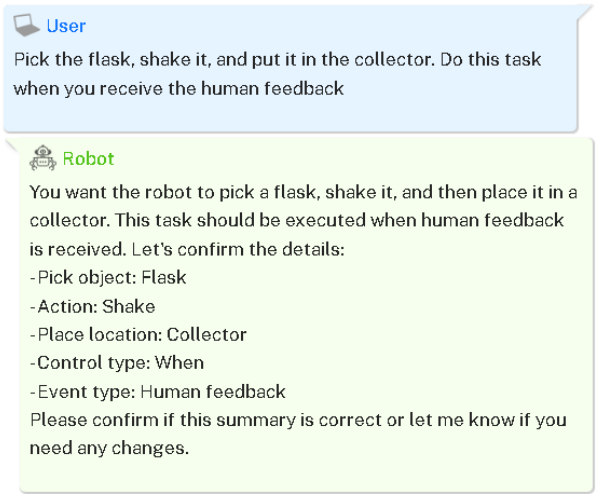

Figure 1: Example of a task definition via chat interface

Underlying this interface is a Large Language Model (LLM) responsible for extracting and interpreting user intent from natural language inputs. In this implementation, the LLM used is OpenAI *GPT-4o*. To guide its behavior, the model is initialized with a detailed prompt that defines the scope and structure of the task. This prompt specifies the model's role in assisting users with the task definition and outlines a custom JSON schema to be used as the output format. This intermediate representation was introduced due to the LLM's limitations in generating outputs directly compatible with Blockly's syntax. The use of a well-structured and machine-readable JSON format is essential to formalize the user's request in a standardized way, making it parsable across subsequent steps of the application, such as the graphic interface and the digital twin. To ensure maximal determinism and reproducibility of outputs, the model's temperature parameter is set to 0. Notably, the pick-and-place task may include an intermediate processing step between the pick and the place actions, such as a shaking action or a movement toward a specific position. Furthermore, tasks can consist of multiple pick-processing-place sequences, incorporate loops (e.g., execute a sequence of actions five times) and conditional logic (e.g., execute a sequence of actions only if there is a feedback from the human, such as the human touches the robot), and support hierarchical structures with varying levels of nesting, such as a loop inside a conditional statement.

An example of a chat interaction is reported in Figure 1. As discussed in [16], the complexity of tasks that can be conveyed through natural language is inherently limited. Even in human communication, it is challenging to articulate logic involving multiple nested levels. Consequently, this application restricts task definitions to a maximum of two levels of nesting (e.g., a conditional statement inside another conditional), a constraint that significantly narrows the scope of intent recognition and simplifies the model's interpretive responsibilities. For instance, the user can define a task consisting of a sequence of pick-processing-place actions, which may be executed a given number of times in a loop (i.e., internal statement), depending on an external condition (i.e., external statement), such as a sensor signal. Once a task has been defined, the chat interface provides a summary of the interpreted information for user verification. At this stage, the user can either revise the task by providing an explanation for the modification or confirm it. Upon confirmation, the task is rendered in the graphic interface, offering a visual representation of the robot program.

In the lab scenario, assuming that the necessary items, like the flask, the shaking action, and the collector, have already been defined in the EUD environment, the technician can describe the task through the chat interface saying: *"Pick the flask, shake it, and put it in the collector. Do this task when you receive the human feedback"*. If needed, the user can refine it, for example, by switching the object from a flask to a test tube, without rewriting the entire task. The LLM interprets this instruction and generates a JSON-formatted task representation. Figure 1 illustrates an example of this interaction.

2.2. The graphic interface

The graphic interface serves as a complementary tool for both reviewing tasks defined via the chat interface and creating new tasks from scratch. Unlike the chat-based interaction, the graphic interface supports the creation of tasks with arbitrary levels of nesting. During the review process, the graphic interface enables users to address any ambiguities or misinterpretations arising from the natural language interaction by allowing direct manipulation of the task structure through drag-and-drop interaction. This includes rearranging, inserting, or deleting task items represented as blocks. The interface is built upon the Google Blockly library, leveraging its flexibility to define custom blocks and enforce domain-specific connection rules that align with the application semantics. These customizations are implemented by the system developers with the goal of making them accessible and usable by end users.

A total of six custom block categories were designed to structure task creation. The *Logic* category includes control structures such as loops and conditionals (e.g., "When-Do", "Repeat"), enabling the definition of task logic. The *Events* category comprises blocks that serve as conditional triggers within the logic, such as "Sensor signal" and "Different object detected". The *Steps* category contains the core action blocks, i.e., "Pick", "Processing", and "Place", used to construct the fundamental flow of a task. The remaining categories, *Objects*, *Actions*, and *Locations*, provide the contextual items required by the steps. Specifically, these blocks represent the objects to be manipulated, the processing actions to be performed, and the target locations for placement, respectively. Each of these contextual items can be defined by the user within the EUD environment. For example, new objects (e.g., a flask) can be defined using the robot's camera to capture their shape; new actions (e.g., a shaking movement) can be recorded by physically guiding the robot's arm and saving the corresponding point cloud; and new locations (e.g., a collector) can be set by manually positioning the arm and saving that position. Further technical details on these definition processes are available in [14].

To ensure syntactic correctness in task construction, a set of rules governs how blocks may be connected within the interface. These rules help users to assemble well-structured workflows. While the blocks reflect standard programming constructs, their labels reflect users' terminology. For instance, conditional statements are expressed using the "When-Do-Otherwise" construct rather than the conventional "If-Then-Else," thereby enhancing accessibility for users without programming experience. An example can be: "*when* a sensor signal is received, *do* pick the square pill; *otherwise*, pick the circular pill".

Going back to the running example, the user moves to the graphic interface, where the block-based representation of the task is shown. The interface uses familiar terminology, making it easy to understand and modify the task. Even at this stage, adjustments, like changing the final destination of an object, can be made by simply replacing a block. The graphic visualization in Blockly of the task previously defined with chat is shown in Figure 2.

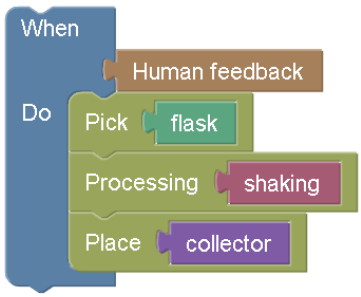

Figure 2: Example of a task visualization via graphic interface

2.3. The digital twin

The digital twin serves as an effective tool for verifying the correctness and completeness of a defined task. Through desktop-based virtual reality visualization, users are able to assess the accuracy of task specifications and ensure the coherence of task items, such as objects, actions, locations, conditions, and control structures like loops.

The digital twin is implemented using the Gazebo simulator and includes both a virtual model of the robot and a representation of its operational environment. Figure 3 illustrates the integration of the various components within the EUD environment.

The process begins by interacting with the chat interface (depicted as the green box in the figure), where users formulate their requests using natural language. The goal of this LLM-powered interface is to interpret the user's input and generate a structured representation of intent. Upon completion of the dialogue, the LLM produces a task description in a custom JSON format.

This JSON output is subsequently processed by Python-based functions that parse and convert it into a Blockly-compliant JSON structure. During this conversion, the functions also retrieve from the database the identifiers for task-relevant entities, such as objects, locations, and actions, augmenting the task definition with necessary metadata. These data play a crucial role in executing tasks in real-world environments, as they enable the system to retrieve the necessary technical information, such as object images or precise location coordinates, required for accurate and effective operation. The use of synonyms defined during the item definition process improves the accuracy of database queries by allowing the system to recognize and match different terms that refer to the same item, thereby reducing the risk of missed or incorrect results.

Within the Blockly graphic interface (shown as the blue box in the figure), users can review the generated task representation and make any necessary adjustments.

Finally, a Python-based program template processes the Blockly JSON to extract key task items and synthesize them into a sequence of executable robot instructions. This sequence can then be run either in the Gazebo-based simulation environment (orange box in the figure) or on the physical robot.

As a final step of the running example, the user can simulate the task using the digital twin. Since real-world sensor data must be approximated in the simulation, the user must specify whether the

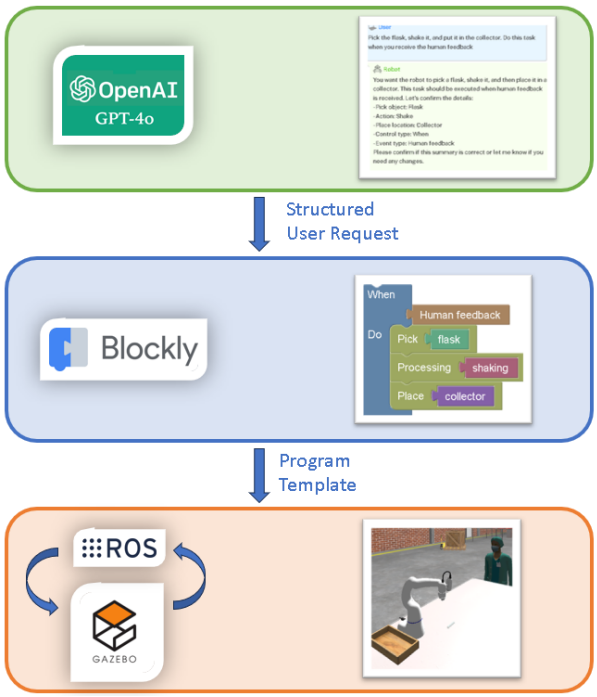

Figure 3: Core components of the EUD environment

condition, such as human feedback, is to be treated as true or false.

This end-to-end running example demonstrates how users can intuitively define, review, and test complex tasks without needing to engage with technical programming tools.

3. Conclusion and Future Work

The proposed EUD environment supports a comprehensive and structured workflow for robot programming. This workflow begins with the definition of domain-specific items, proceeds through task specification via both the chat interface and the block-based graphic interface, continues with task validation using a digital twin, and culminates in the execution of the validated task on a physical robot.

In contrast to existing applications that rely on simulation environments [17] or employ Gazebo and ROS technologies [18, 19], the proposed system removes the need for end-users to interact with complex technical tools or programming languages. Instead, it offers a web-based interface tailored to non-expert users. Adhering to human-centered AI design principles [20], the system employs domain-consistent terminology to facilitate understanding and usability.

Despite these strengths, there are still some limitations in the current implementation that will be overcome in the next months. First, the fidelity of the mapping between the physical environment and its digital representation should be improved. Key aspects, such as object manipulation, image acquisition via the robot's camera, and environmental simulation, require greater realism to more accurately reflect real-world conditions. Second, the current version of the EUD environment enables relatively simple task definitions (e.g., pick-and-place tasks with basic processing actions) and lacks the expressiveness needed to define and simulate more complex operations. Finally, while it is essential to demonstrate the technical capabilities of the system, conducting an empirical evaluation would enhance the overall research contribution. It will be important to plan a user study with non-programmers to assess the system's capacity in interpreting users' requests and generating accurate outputs. Preliminary testing has already been conducted on a customized application developed for pharmacists specialized in galenic preparations [21]. This version does not employ either Blockly or the Digital Twin; however, it has a dedicated graphical interface that has been tailored to the specific domain. The results of these preliminary studies have provided valuable insights, and the feedback has been, for the most part, positive.

Future enhancements will aim to improve the accuracy of the digital twin, support the specification and simulation of more complex robotic tasks, and validate the system's usability and effectiveness through structured user evaluations. Additional efforts will explore techniques to better align the LLM with domain-specific requirements. In particular, Retrieval-Augmented Generation (RAG) methods will be investigated to provide contextual information that enhances task interpretation and output accuracy. Furthermore, the personalization of user interactions, tailored to individual preferences and prior behavior, will be explored to further improve user experience.

This work advances the field of end-user robot programming by introducing a human-centered artificial intelligence approach that harnesses the capabilities of large language models and digital twin technology. The proposed workflow empowers users, regardless of their programming or robotics background, to design, test, and refine robot programs using intuitive tools. By centering the development process around the user, the approach aims to deliver accessible, efficient, and effective programming experiences tailored to individual needs. The overarching goal of this work is to foster a progressive learning path for human users, facilitating the acquisition of robot programming competencies and promoting the development of computational fluency [22, 23]. Through this approach, users are expected to advance seamlessly in their programming abilities over time.

Declaration on Generative AI

During the preparation of this work, the author used ChatGPT and Grammarly in order to: grammar and spelling check, paraphrase, and reword. After using these services, the author reviewed and edited the content as needed, thus, he takes full responsibility for the publication's content.

References

[1] V. Villani, F. Pini, F. Leali, C. Secchi, Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications, Mechatronics 55 (2018) 248–266. doi:10.1016/j.mechatronics.2018.02.009.

[2] J. E. Michaelis, A. Siebert-Evenstone, D. W. Shaffer, B. Mutlu, Collaborative or simply uncaged? understanding human-cobot interactions in automation, in: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, CHI '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 1–12. doi:10.1145/3313831.3376547.

[3] M. Löfving, P. Almström, C. Jarebrant, B. Wadman, M. Widfeldt, Evaluation of flexible automation for small batch production, Procedia Manufacturing 25 (2018) 177–184. doi:10.1016/j.promfg.2018.06.072, proceedings of the 8th Swedish Production Symposium (SPS 2018).

[4] S. El Zaatari, M. Marei, W. Li, Z. Usman, Cobot programming for collaborative industrial tasks: An overview, Robotics and Autonomous Systems 116 (2019) 162–180. doi:10.1016/j.robot.2019.03.003.

[5] L. Peternel, W. Kim, J. Babič, A. Ajoudani, Towards ergonomic control of human-robot co-manipulation and handover, in: 2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids), 2017, pp. 55–60. doi:10.1109/HUMANOIDS.2017.8239537.

[6] G. Ajaykumar, M. Steele, C.-M. Huang, A survey on end-user robot programming, ACM Comput. Surv. 54 (2021). doi:10.1145/3466819.

[7] H. Lieberman, F. Paternò, V. Wulf, End User Development (Human-Computer Interaction Series), Springer-Verlag, Berlin, Heidelberg, 2006.

[8] F. Paternò, V. Wulf (Eds.), New Perspectives in End-User Development, Springer, Cham, 2017. doi:10.1007/978-3-319-60291-2.

[9] B. R. Barricelli, F. Cassano, D. Fogli, A. Piccinno, End-user development, end-user programming and end-user software engineering: A systematic mapping study, Journal of Systems and Software 149 (2019) 101–137. doi:10.1016/j.jss.2018.11.041.

[10] Google, Blockly, 2012. URL: https://developers.google.com/blockly.

[11] J. Huang, M. Cakmak, Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts, in: Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, HRI '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 453–462. doi:10.1145/2909824.3020215.

[12] D. Weintrop, A. Afzal, J. Salac, P. Francis, B. Li, D. C. Shepherd, D. Franklin, Evaluating coblox: A comparative study of robotics programming environments for adult novices, in: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1–12. doi:10.1145/3173574.3173940.

[13] B. R. Barricelli, E. Casiraghi, D. Fogli, A survey on digital twin: Definitions, characteristics, applications, and design implications, IEEE access 7 (2019) 167653–167671.

[14] D. Fogli, L. Gargioni, G. Guida, F. Tampalini, A hybrid approach to user-oriented programming of collaborative robots, Robotics and Computer-Integrated Manufacturing 73 (2022) 102234. doi:10.1016/j.rcim.2021.102234.

[15] L. Gargioni, D. Fogli, Integrating chatgpt with blockly for end-user development of robot tasks, in: Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction, HRI '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 478–482. doi:10.1145/3610978.3640653.

[16] A. K. Lampinen, Can language models handle recursively nested grammatical structures? A case study on comparing models and humans, 2023. URL: https://arxiv.org/abs/2210.15303. arXiv:2210.15303.

[17] A. Schoen, N. White, C. Henrichs, A. Siebert-Evenstone, D. Shaffer, B. Mutlu, Coframe: A system for training novice cobot programmers, in: 2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2022, pp. 185–194. doi:10.1109/HRI53351.2022.9889345.

[18] I. Peake, J. La Delfa, R. Bejarano, J. O. Blech, Simulation components in Gazebo, in: 2021 22nd

IEEE International Conference on Industrial Technology (ICIT), volume 1, 2021, pp. 1169–1175. doi:10.1109/ICIT46573.2021.9453594.

[19] C.-J. Liang, W. McGee, C. C. Menassa, V. R. Kamat, Real-time state synchronization between physical construction robots and process-level digital twins, Construction Robotics 6 (2022) 57–73. doi:10.1007/s41693-022-00068-1.

[20] B. Shneiderman, Human-centered AI, Oxford University Press, Oxford, UK, 2022.

[21] L. Gargioni, D. Fogli, P. Baroni, Preparation of personalized medicines through collaborative robots: A hybrid approach to the end-user development of robot programs, ACM Journal on Responsible Computing (2025).

[22] B. R. Barricelli, D. Fogli, A. Locoro, EUDability: A new construct at the intersection of end-user development and computational thinking, Journal of Systems and Software 195 (2023) 111516. doi:10.1016/j.jss.2022.111516.

[23] B. R. Barricelli, D. Fogli, L. Gargioni, A. Locoro, S. Valtolina, Towards the unification of computational thinking and eudability: Two cases from healthcare, in: Proceedings of the 2024 International Conference on Advanced Visual Interfaces, AVI '24, Association for Computing Machinery, New York, NY, USA, 2024, pp. 1–9. doi:10.1145/3656650.3656671.