Robotica (2025), 1–35
doi:10.1017/S0263574725101720

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

A gesture-based behaviour-driven development approach for end-user cobot programming

Anahide Silahli[1], Jose Pablo De la Rosa[1], Jorge Solis[2], Gustavo Alfonso Garcia Ricardez[3], Lotfi El Hafi[3], Johan Håkansson[4], Anders Stengaard Sørensen[1] and Thiago Rocha Silva[1]

[1]The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Odense, Denmark
[2]Karlstad University, Karlstad, Sweden
[3]Research Organization of Science and Technology, Ritsumeikan University, Kusatsu, Japan
[4]Goodtech Solutions AB, Karlstad, Sweden
Corresponding author: Anahide Silahli; Email: ansil@mmmi.sdu.dk

Received: 29 November 2024; **Revised:** 15 April 2025; **Accepted:** 22 April 2025

Keywords: End-User Development (EUD); robot programming; industry 5.0; collaborative robots; multimodal interaction; Behaviour-Driven Development (BDD); Domain-Specific Languages (DSLs)

Abstract

This study presents an innovative framework to improve the accessibility and usability of collaborative robot programming. Building on previous research that evaluated the feasibility of using a domain-specific language based on behaviour-driven development, this paper addresses the limitations of earlier work by integrating additional features like a drag-and-drop Blockly web interface. The system enables end users to define and execute robot actions with minimal technical knowledge, making it more adaptable and intuitive. Additionally, a gesture-recognition module facilitates multimodal interaction, allowing users to control robots through natural gestures. The system was evaluated through a user study involving participants with varying levels of professional experience and little to no programming background. Results indicate significant improvements in user satisfaction, with the system usability scale overall score increasing from 7.50 to 8.67 out of a maximum of 10 and integration ratings rising from 4.42 to 4.58 out of 5. Participants completed tasks using a manageable number of blocks (5 to 8) and reported low frustration levels (mean: 8.75 out of 100) alongside moderate mental demand (mean: 38.33 out of 100). These findings demonstrate the tool's effectiveness in reducing cognitive load, enhancing user engagement and supporting intuitive, efficient programming of collaborative robots for industrial applications.

1. Introduction

Improving the efficiency of today's manufacturing technology is paramount, and the use of collaborative robots (cobots) has become an essential part of the equation. Many countries, including those in Scandinavia and Japan, highly depend on the manufacturing industry but there are still several significant challenges. One challenge is the increasing employment rate among people aged 55–64 over the last 20 years in many Organisation for Economic Co-operation and Development (OECD) countries [1]. In some countries, like Japan and Korea, this has significantly increased the labour force participation among workers over 65. Although ageing societies are common in Europe, the share of the labour force over 65 years old remains relatively lower [2]. As a preventive measure, the European Union's Industry 5.0 program aims to employ research and innovation to achieve a more sustainable and human-centric European industry [3].

To address these challenges, this research proposes a framework to enhance communication and synchronization between humans and robots in collaborative tasks, using mixed reality (MR) and artificial intelligence (AI). With MR, the framework visually conveys robot plans, motions, and status, as well as the robot's processed environmental data (e.g., detected objects, detected human, task state). With

rich, intuitive visual information, human collaboration is expected to be enhanced, improving decision-making and assessment of the robot's context to ensure task continuity. The AI module (1) interprets multi-modal information from the human, such as hand gestures, body posture, and voice commands, (2) generates robot behaviours that are responsive to human actions, and (3) adaptively determines the next steps based on human intention and the task progress, relaying decisions back to the human via MR.

The feasibility of integrating an ML-based gesture recognition system into an industrial collaborative robot was previously verified [4], as well as using MR-based displays of cobot status through Microsoft HoloLens for user interaction in different simplified collaborative scenarios [5]. Additionally, recent research has proposed and evaluated safe and adaptable systems for human-robot collaboration that can modify robot behaviour through natural language, based on user safety perceptions [6]. Moreover, adaptive task-planning systems have further enabled reliable human-robot collaboration for non-predefined tasks instructed via multimodal communication. However, discussions with industrial partners on the preliminary results from the human-robot interaction (HRI) perspective pointed out the importance of providing flexibility to end users for customizing programmes [7].

End-user development (EUD) [8] has been studied for many years in software engineering to allow subject-matter experts with no programming expertise to program and/or adapt software applications in their domain. In robotics, an expanding body of literature has been proposed to simplify the programming of automation tasks through methods such as Programming by Demonstration, Visual Programming, and Natural Language specification.

Despite advancements in human-robot collaboration, accessibility remains a significant challenge, as many programming methods require substantial technical knowledge. This complexity, combined with a lack of integrated, intuitive interfaces for multimodal communication, limits the ability of non-experts to effectively interact with robots. While tools for natural language and gesture recognition exist, there remains a need for more user-friendly systems that simplify the definition and execution of robot behaviours. Addressing these gaps is crucial for advancing collaboration in industrial environments.

To address these challenges, this study focuses on enhancing the accessibility and usability of robot programming for non-expert users by introducing several key innovations. First, a new intuitive Blockly-based web interface is designed to enable non-experts to define robot actions with minimal programming knowledge. Users define robot actions using a domain-specific language (DSL) based on behaviour-driven development (BDD) [9], which provides clarity in the task definition. BDD is a software development approach in which domain experts can specify system behaviour through independent, functional examples called scenarios, written in a semi-structured natural language format.

In parallel, an expanding body of literature on visually enhanced DSLs, particularly when integrated with popular tools like Scratch [10] or Blockly [11], encourages simple natural language through a logical visual representation of the coded sequences. Such approaches have shown promise in enhancing non-expert development by reducing syntax errors and improving the discoverability of development environments [12].

The proposed interface is a web-based visual environment using Blockly [11], which simplifies scenario creation with a modular, drag-and-drop interface that makes the process straightforward and free of syntax errors. Additionally, it incorporates gesture-based interaction, allowing end users to control tasks through natural movements, further enhancing the user experience and making the process more fluid and engaging.

This paper continues the research presented in [13], which introduced a DSL based on a *Given-When-Then* framework to improve human-cobot interaction through intuitive programming scenarios. In this earlier study, we applied this approach to the system developed by [4]. Implemented on an ABB Dual-Arm YuMi robot with a 3D gesture recognition system, the end-user programming framework allowed users to control robot actions using body gestures. This design aimed to make cobot programming accessible to non-experts, supporting the human-centric vision of Industry 5.0 [14]. To evaluate this framework, a user study was conducted with 12 participants without programming experience. Participants created BDD scenarios within a programming editor displayed on a computer screen,

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

enabling control of the robot in a pick-and-place task. The study demonstrated the tool's usability and adaptability for industrial tasks, identifying it as user-friendly and effective. However, areas for improvement were noted, including gesture recognition accuracy and task flexibility.

In this paper, the BDD-based end-user programming framework is optimized and further investigated through experiments involving 12 new participants without prior programming experience. The study's task now involves three distinct phases, *positioning, grasping and placing*, offering higher cognitive demand and a more elaborate sequence than the pick-and-place task from the previous study. Unlike the earlier research, where the pick-and-place task was pre-coded and user gestures triggered robot adjustments (e.g., changing speed or stopping), this study allows participants to build a complete sequential task themselves. The new BDD-based Blockly interface represents a significant improvement over the earlier study, where participants relied on a basic programming editor and used paper documents to label gesture signs. The updated platform allows participants to assemble Blockly blocks with distinct types, shapes and colours for easy differentiation. Some blocks include images of the required hand gestures and the interface is divided into fields with interactive buttons. Real-time feedback is integrated, providing error messages and tracking code progress, ensuring a more guided and user-friendly experience. Additionally, the study was conducted in Denmark rather than Sweden, introducing diversity to the user base and offering a comparative perspective on the interface's effectiveness. The task was carefully designed to evaluate the interface's usability, focusing on ease of understanding, user-friendliness, intuitiveness, and adaptability in a practical robotic context.

Given the improvements introduced in the present study, including a new interface and more complex task structures requiring higher cognitive involvement, the following research questions have been defined:

- **RQ1:** *How does prior programming experience affect non-expert users' perception of the tool's intuitiveness and usability?*
- **RQ2:** *How does general professional experience influence users' perceptions of the tool's intuitiveness and their ability to complete robot task assignments?*
- **RQ3:** *What aspects of user experience (*e.g.*, intuitiveness, ease of learning, system integration) show the most significant improvements in the new tool?*
- **RQ4:** *How do the System Usability Scale (SUS) metrics and user feedback in the current study validate or challenge the qualitative improvements identified in the previous research?*

Through these research questions, this study aims to evaluate the impact of the new interface on user experience, identify the areas that have shown the most significant improvements, and determine the tool's effectiveness in real-world industrial applications.

2. Related work

Human-robot collaboration (HRC) systems aim to bridge the gap between human users and robotic technologies, facilitating intuitive and continuous interaction even for non-expert users. Recent advancements in HRC systems focus on improving accessibility, usability, and functionality across various domains, including industrial robotics, social robotics, and educational tools. This section provides an overview of the current approaches and challenges in this field.

2.1. Behaviour-driven development

BDD, introduced in 2006 [15], serves as a practical approach for defining software requirements and acceptance criteria while addressing communication barriers between subject-matter experts and developers. By focusing on concrete examples of system behaviour [16], BDD helps stakeholders better understand how specific features contribute to business value. The method promotes collaboration by allowing requirements to be articulated in a format that is both testable and easy for technical and non-technical participants to interpret [17].

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

To achieve this, BDD utilizes scenarios [18] described in a semi-structured natural language format based on Gherkin syntax [19]. These scenarios typically include a title and three main parts: *Given* which sets the context for the scenario, *When* which specifies the action taken, and *Then* which outlines the expected outcome. Additional details, such as multiple contexts, actions, or outcomes, can be included using "And" statements within these steps as can be seen below.

```
Scenario: <title>
Given [context]
And [some more context]...
When [event]
And [another event]...
Then [outcome]
And [another outcome]...
```

2.2. Natural language programming systems

Natural language programming systems have been lastly deployed in the robotics field to make their programming intuitive and engaging for non-technical users. Added to multimodal interfaces, these systems have enhanced features to allow users to interact with robots using natural modes of communication such as speech and gestures. For example, Gorostiza and Salichs introduced a Natural Programming System to make programming social robots more accessible for non-expert users in education and entertainment (edutainment) contexts. Their system leverages multimodal interaction, enabling users to communicate with robots more naturally. While promising, the system is constrained by limited adaptability to diverse human communication styles (e.g., accents, gestures) and dependence on specific utterances. Additionally, small sample sizes and a lack of robust qualitative metrics have hindered its comprehensive evaluation [20].

Another key contribution is the CAPIRCI system by Beschi *et al.* [21], which integrates natural language and block-based programming to simplify robotic task definitions. This hybrid approach allows users to define tasks using both intuitive blocks and natural language. However, it faces challenges such as issues with pronoun resolution, difficulties with complex commands and the inability to handle dynamic workflows or real-time error recovery. These limitations emphasize the need for systems capable of managing complex, adaptive, multi-step tasks.

The Teaching-Learning-Collaboration model proposed by Wang *et al.* [22] employs natural language to enable robots to learn from human demonstrations, improving task efficiency and reducing programming effort. The model's key benefit is its ability to learn from human input, reducing the need for complex programming. However, the model's robustness is compromised by noisy inputs and ambiguous commands and its scope is limited to basic tasks. Furthermore, its application in complex industrial scenarios remains untested, underscoring the need for systems with enhanced safety and reliability in collaborative environments.

2.3. Visual programming and model-driven approaches

Visual programming environments empower non-expert users by providing intuitive, graphical interfaces for robot programming. Systems like EUD-MARS, developed by Akiki *et al.* [23], combine model-driven development with block-based programming, enabling task customization across various robot types. This hybrid approach supports diverse robot applications while keeping the interface accessible. Despite its accessibility, challenges such as block misinterpretation and limited multi-robot conflict resolution reduce its utility in complex environments.

The work of Silahli *et al.* [24] introduced a framework combining a DSL with a Neural Network (NN) model for intuitive robot motion specification. The DSL allows users to define motions with high-level commands, while the NN, trained on Dynamic Movement Primitives, generates smooth and precise robot movements. This approach simplifies robot programming for non-experts and adapts across various robotic platforms, as demonstrated through experiments on a robotic arm.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

No-code solutions are also gaining traction, exemplified by Blanc *et al.*'s block-based interface [25], which empowers users through guided tutorials. The main advantage of this system is its simplicity and usability, enabling non-technical users to complete tasks with minimal effort. While this system demonstrated strong usability for novice users, it struggled with unclear guidance and lacked evaluation of long-term adaptability. Additionally, its primary focus on beginners rather than shop-floor workers limits its practical industrial relevance.

Behaviour Tree frameworks offer another approach for simplifying robot behaviour design. Tulathum *et al.* [26] proposed a drag-and-drop system with integrated debugging tools, which allowed convenience store staff to program robot behaviours without needing deep technical knowledge. This system's main benefit is that it combines simplicity with powerful debugging features, though participants reported difficulties with organizing complex tasks, highlighting the need for better visual guidance and training.

Similarly, Trigger-Action Programming has been employed to personalize humanoid robot behaviours in IoT-integrated environments. Leonardi *et al.* presented a platform enabling users to define context-dependent actions, such as speech and gestures. This approach's flexibility allows for diverse behaviour customization, but it faces significant scalability and error management challenges, with its applicability outside controlled environments remaining uncertain [27].

For children, block-based programming tools such as NaoBlocks by Sutherland and MacDonald [28] prioritize simplicity and engagement. The iterative development of this tool, based on user feedback, allows for enhanced learning experiences in programming. While these systems excel in encouraging creativity and supporting diverse programming strategies, usability barriers like robot selection difficulties and robustness issues indicate the need for guided assistance and improved generalizability.

2.4. Machine learning and AI for HRC

Machine Learning (ML) is pivotal in advancing HRC, enabling robots to adapt to dynamic, real-world scenarios. Mukherjee *et al.* [29] emphasize ML's role in enhancing pose detection, intention prediction and decision-making in industrial applications. By enabling robots to learn from experience, these systems can improve efficiency and adaptability. Despite its potential, ML integration faces problems such as limited training datasets, noise sensitivity and challenges in achieving real-time adaptability. Moreover, stringent industrial regulations add complexity to deploying ML-based systems in safety-critical environments, necessitating robust and reliable models.

2.5. Mixed reality and augmented reality interfaces

Augmented Reality enhances HRC by enabling intuitive and context-aware programming systems. Kapinus *et al.* [30] introduced a spatially situated programming framework that allows users to directly manipulate robot tasks in real-world contexts. This approach reduces cognitive load and increases task efficiency by minimizing the need to switch between programming and physical spaces. However, it faced challenges such as interface clutter and debugging difficulties.

Situated Live Programming (SLP) frameworks, such as the one developed by Senft *et al.* [31], represent an end-user programming approach that allows users with limited programming experience to create collaborative applications for human-robot interaction. In their framework, incremental task programming is facilitated through annotated augmented video feeds, blending the programming environment with real-world scenarios. The main advantage of this approach is its smooth integration of programming into the physical environment. However, while it is effective for simpler tasks, SLP faces challenges with debugging, managing complex tasks and reliance on predefined actions, which limits its applicability in dynamic workflows.

Mixed Reality (MR) systems simplify programming through 3D interfaces and hand gestures. Gadre *et al.* [32] demonstrate that this approach reduces cognitive effort and enhances task completion speed, allowing users to interact with robots more naturally. However, their dependency on MR technology and small-scale evaluations limit broader applicability.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

2.6. Evaluation challenges in tools and approaches for enhancing HRC

A critical limitation in the development of tools and approaches for improving HRC is the lack of comprehensive, real-world, long-term evaluations. For example, studies such as Beschi *et al.* [21] and Wang *et al.* [22] are constrained by short-term assessments or controlled environments, which fail to capture usability and robustness under practical, dynamic conditions. Moreover, participant diversity in evaluations remains a significant issue. Many studies focus narrowly on specific user groups, such as students or IT professionals, neglecting the needs of other demographics like older adults, children, or non-technical users, who may interact differently with HRC systems.

2.7. Multi-domain applications and scalability of HRC tools

Many tools and approaches for HRC suffer from limited scalability and adaptability across different domains. For instance, systems designed for industrial environments, such as Akiki *et al.*'s model-driven adaptive robotics framework [23], often lack the flexibility to be deployed in healthcare or domestic applications. Similarly, tools tailored for social or educational robots may not perform effectively in dynamic industrial settings. Addressing this challenge requires a focus on developing scalable and versatile solutions that can accommodate diverse task requirements and adapt seamlessly to multiple contexts, thereby broadening the scope of HRC applications.

3. Methods

This study follows established empirical methods for evaluating the proposed tool and its effects on HRC. The methodology draws on ***Engineering Research***, ***Qualitative Surveys*** and ***Experiments (with Human Participants)***, each of which contributes to assessing the tool's usability, effectiveness, and real-world applicability. Below, we briefly describe the adopted approaches.

3.1. Engineering research

The primary methodology employed in this study is ***Engineering Research*** (or Design Science), which involves the creation and evaluation of technological artefacts; in this case, a BDD-based Blockly web interface for robot programming. This approach focuses on developing new solutions (the new programming tool) and empirically assessing their effectiveness. The artefact here is evaluated through experiments with human participants who interact with the tool to assess its impact on usability and task performance.

This methodology allows for a focused evaluation of the tool's design, its integration with existing systems and its potential for real-world application in the context of HRI. The engineering research standard ensures that the artefact is tested against clear research questions, with empirical results contributing to both practical insights and theoretical contributions to the field.

3.2. Experiments (with human participants)

To assess the impact of the tool in a controlled environment, the study adopts the ***Experiments (with human participants)*** standard. This method allows for the manipulation of independent variables (such as the interface and task complexity) to observe their effects on dependent variables like task completion time, error rates, and user satisfaction. Participants are assigned to different experimental conditions to evaluate the tool's usability across different user groups, including those with basic and without prior programming experience.

Experiments with human participants provide empirical evidence of the tool's effectiveness in real-world settings and showcase its strengths and weaknesses in facilitating robot programming tasks. Post-task standardized questionaries are applied to assess the tool's usability and the user's cognitive load, along with open-ended questions to allow participants express their overall experience and the perceived opportunities for improvements.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

```
# Sequence and Scenario definitions
<sequence>              ::= (<scenario>) *
<scenario>              ::= 'Scenario:' <description>
                            'Given' <initial state>
                            'When'  <event>
                            'Then'  <resulting state>

# State-related clauses for robots and objects
<initial state>         ::= (<robot state clause> | <object state clause>)
<resulting state>       ::= (<robot state clause> | <object state clause>)

# Event triggered by the user affecting the system's state
<event>                 ::= <user entity> <user action>

# Robot and Object state clauses
<robot state clause>    ::= 'the' <robot entity>  'is' 'not'? <robot state>
<object state clause>   ::= 'the' <object entity> 'is' 'not'? <object state>

# Robot and Object states
<robot state>           ::= 'positioned' <robot position>
<object state>          ::= 'picked from' | 'placed at' <object position>

# Entities and user actions
<user action>           ::= 'do the' <gesture name> 'sign'
<user entity>           ::= 'I'
<robot entity>          ::= 'robot'
<object entity>         ::= 'object'

# Terminals
<description>           ::= <STRING>
<gesture name>          ::= <STRING>
<robot position>        ::= <STRING>
<object position>       ::= <STRING>
```

Listing 1: *Backus–Naur Form grammar definition of the DSL*

4. A BDD-based Blockly web interface for robot programming

This work investigates the use of BDD as a method to facilitate task specification for end-user programming in bodily human-robot interaction scenarios. As a proof of concept, a DSL was developed that allows users to define interactive scenarios with a collaborative robot using a semi-structured natural language approach guided by BDD. Subsequently, an EUD environment was implemented comprising a suite of tools to support end users, specifically non-expert robot users, throughout the authoring and execution processes of interactive scenarios based on the proposed DSL.

This section provides an overview of (a) the proposed DSL, (b) the EUD environment implementation, and (c) the mechanisms and principles supporting the execution of user-defined code.

4.1. BDD-based domain-specific language

In this approach, BDD scenarios are interpreted as complete state transitions within a state machine. A definition of the DSL syntax is presented in Listing 1. A programme, or `<sequence>`, consists of one or more instances of the non-terminal `<scenario>`. Each `<scenario>` follows a *Given-When-Then* format, where `<initial state>` defines the preconditions for the scenario and `<resulting state>` describes the outcomes after an `<event>` occurs. Both `<initial state>` and `<resulting state>` can be specified using either a `<robot state clause>` or an `<object state clause>`, describing the current condition or placement of the `<robot entity>` or `<object entity>`, respectively.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

```
Scenario: Position the robot in the middle !
Given the robot is not positioned in the middle
When I do the Hello sign
Then the robot is positioned in the middle

Scenario: Pick up the object from the Base position
Given the robot is positioned in the middle
When I do the Thumbs up sign
Then the object is picked from Base
```

Listing 2: *Sample scenario: Positioning robot and picking up object from "Base"*

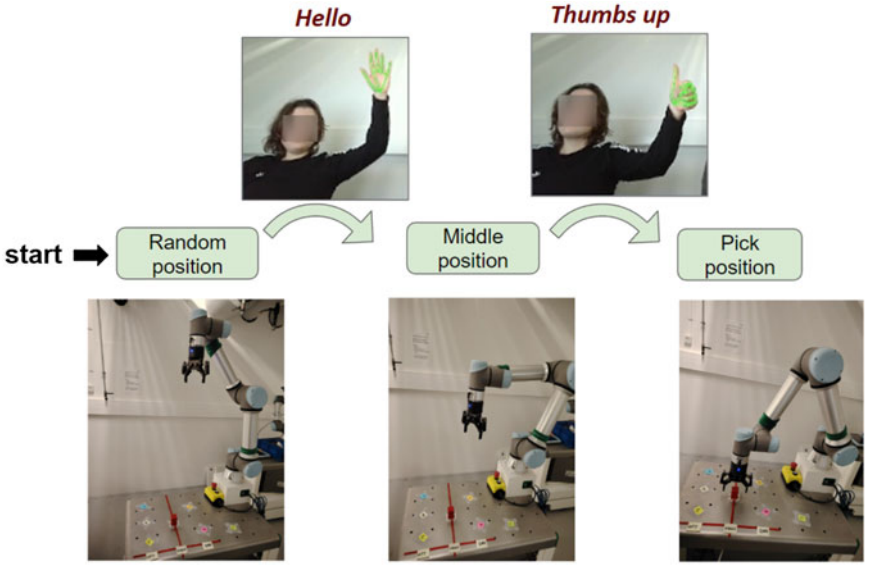


Figure 1. *State diagram from sample sequence in Listing 2. User gestures trigger robot state transitions.*

Events are triggered by user actions, where the `<user entity>` performs a specific gesture (`<user action>`). The terminal `<description>` provides a brief explanation of the scenario, serving a documentation purpose without affecting task execution. The terminals `<gesture name>`, `<robot position>` and `<object position>` are user-defined strings that are translatable into computer-understandable values. `<gesture name>` defines a specific gesture performed by the user, such as a hand sign, while `<robot position>` and `<object position>` refer to the placement of the robot and object, respectively. A mapping file translates these user-defined terminals into computer-understandable values: `<gesture name>` is mapped to a category recognized by a gesture recognition model and `<robot position>` and `<object position>` are mapped to vectors of specific robot positions, pre-trained through demonstration.

Listing 2 shows a sample sequence derived from the provided syntax. This programme can be described using a three-state finite-state machine (Figure 1) that governs the robot's behaviour. Each scenario corresponds to a transition between these states, with user gestures acting as triggers for the transitions. The user-defined Hello sign moves the robot to the middle, while the Thumbs up sign prompts the robot to pick up an object.

4.2. EUD environment architecture

An EUD environment was implemented on a multi-tiered architecture separating client, server and robot layers to ensure modularity and clear division of responsibilities (Figure 2). The interactive elements of the IDE represent the client-side perspective of the architecture. These components enable the end

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

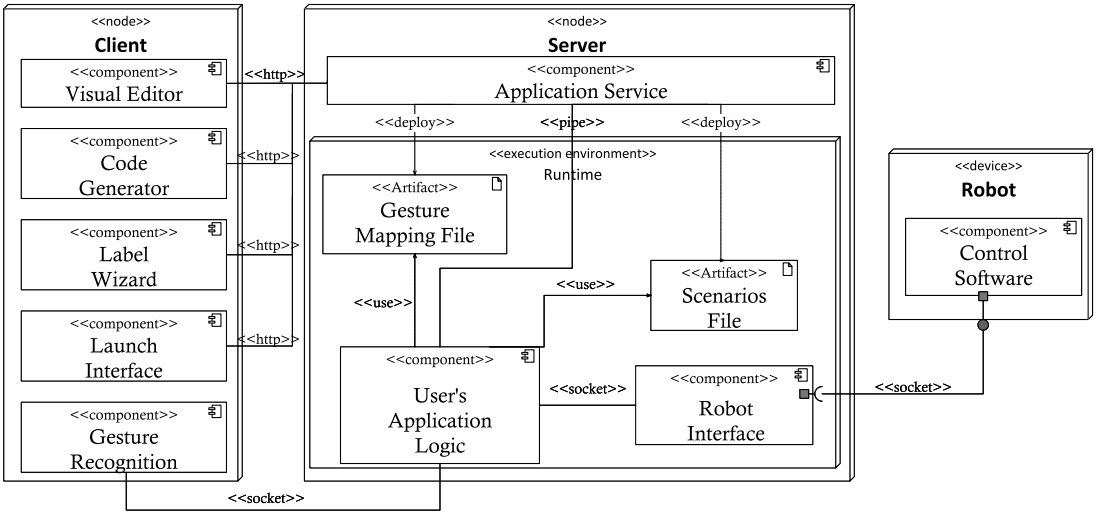


Figure 2. *Overview of the system's architecture.*

user to configure and manage sequences while interacting with the robot (Figure 3), based on five main components:

1. ***Visual Editor***: A block-based editor built using Blockly[1] that allows users to compose independent scenario blocks by dragging and dropping elements from a toolbox into a Workspace zone (Figure 3a), assembling them in a jigsaw-like manner. The grammar of the DSL is enforced through specific block shapes and association rules, which restrict how blocks can connect.
2. ***Launch Interface***: An interactive view adjacent to the development editor (Figure 3b), dynamically updating based on the current stage of the development process. It provides essential functionalities such as launching, stopping, saving, or restoring previous versions of the application. Additionally, an embedded console offers real-time feedback regarding the application's state, enabling users to monitor the execution process, identify current system states, and explore potential transition paths within the application's workflow.
3. ***Code Generator***: This component automatically parses the blocks assembled in the visual editor's workspace to produce a list of scenario strings that adhere to Python's Behave[2] *feature* file format. Upon the user's *Launch* command, the generated feature file is deployed on an application server controlling the robot's execution according to the scenarios defined. Additionally, a secondary code generator, linked to the *Save/Load* functionality, creates or retrieves an XML representation of the workspace's block structure, allowing users to save or restore previous versions of their scenarios for future use.
4. ***Label Wizard***: This tool enables users to create custom labels for available gestures (Figure 3d). User-defined labels are then dynamically integrated as selectable options within the visual editor, specifically for the `<gesture name>` terminal string. For instance, gestures such as *Hello* and *Thumbs up* (shown in Figure 3a) are represented as blocks within the editor. This allows users to define scenarios using action descriptors in their language terms.
5. ***Gesture Recognition***: The gesture recognition view displays a live camera feed, monitoring the user's gestures. Once a gesture is detected, a set of key landmarks is overlaid onto the image, with the corresponding gesture label displayed (Figure 3e). During runtime, identified gesture categories are transmitted to an application server, which uses this data to initiate transitions between the robot's states according to the control logic specified by the user's scenarios.

[1] https://developers.google.com/blockly
[2] https://behave.readthedocs.io/en/stable/gherkin.html#gherkin-feature-testing-language

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

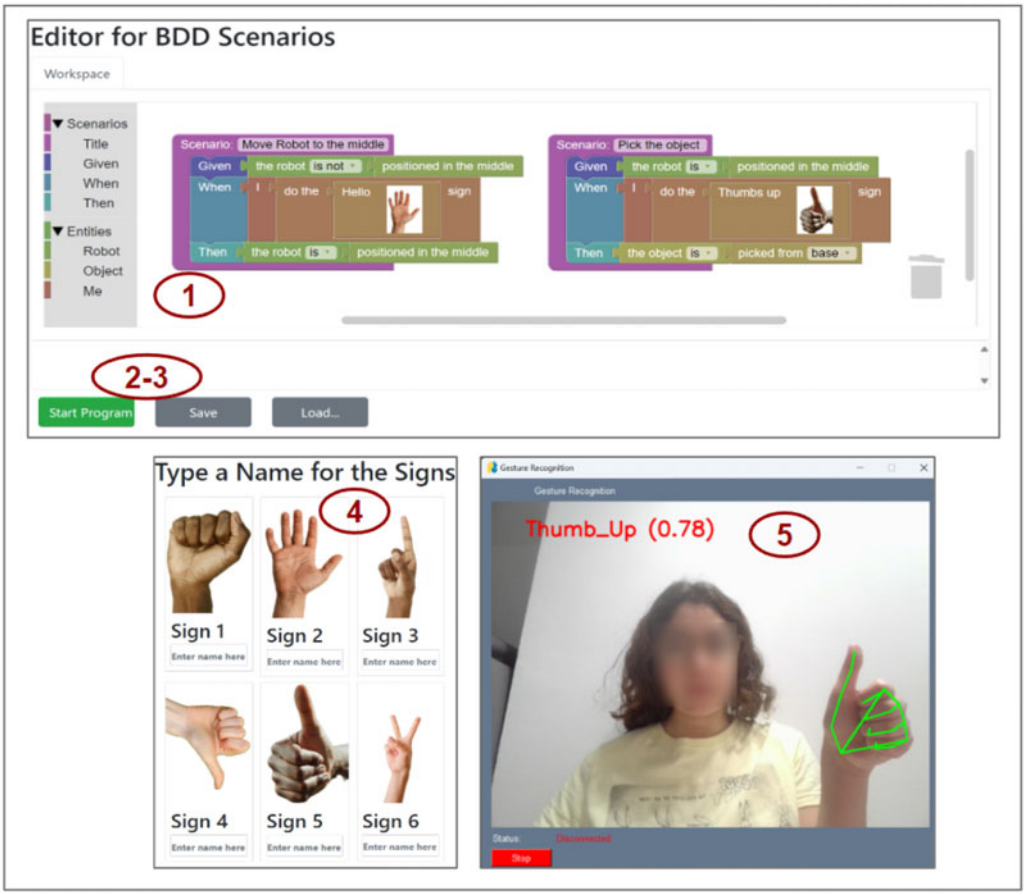


Figure 3. *Interface elements of the development environment.*

On the server side, the computational logic is managed by an *Application Service*, which acts as the central middleware, coordinating between client requests and backend operations. This includes processing gesture interpretations and executing control commands based on user-defined scenarios. Two artefacts, the *Mapping File* and the *Scenarios File*, are used to store configuration mappings and user-defined scenarios, respectively. A *Robot Interface* component within the server communicates with the robot's API, translating user-defined commands into actionable instructions for the robot's proprietary *Control Software*.

4.3. Execution environment

Figure 4 provides an overview of the runtime model. This can be resembled to the pipes-and-filters model [33], in which a complex task is decomposed into a series of successive subtasks. Each subtask is executed by a separate, autonomous unit referred to as a filter. The integration and communication among these filters are facilitated through the exchange of data via channels known as pipes. The architecture is composed of three main filters, developed as independent processes: Gesture Recognition, User's Application Logic, and the Robot Interface. Communication between the filters is piped via TCP web sockets between the processes. Each filter is briefly described below:

1. **Gesture Recognition Filter (GR)**: A continuous running process that receives a stream of data from the camera sensor, providing a recognition category to the output pipe. As a proof of concept, Google's MediaPipe Gesture Recognizer [34] model was implemented to detect and write the category numbers of seven standard hand gestures.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

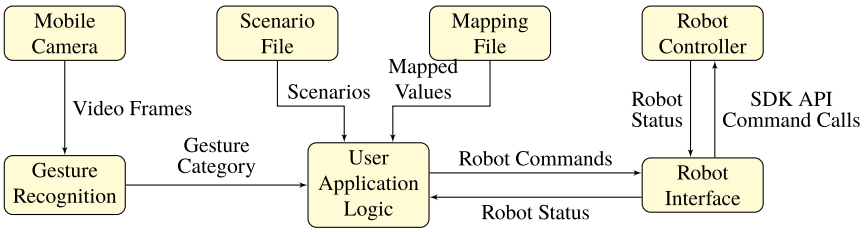


Figure 4. *System architecture based on a pipe-and-filter pattern.*

2. **User Application Logic Filter (UAL)**: A process executing Python's Behave library ([35]) to parse and execute the end user's scenarios, which are contained in text files called features. Behave executes all the *Given*, *When*, and *Then* clauses from the scenarios of such feature files as sequential steps. Each step is mapped to an implementation function by using annotations with regular expressions. The input pipe of this process contains the gesture categories recognized by the GR. To allow the end user to describe gestures and robot positions in their own words, the process uses a mapping file (JSON) that relates the values used by the system with their preferred naming in natural language. If a scenario returns a successful evaluation, the resulting state from the *Then* clause is written to the output pipe, which is used by the Robot Interface for updating the robot's behaviour.
3. **Robot Interface**: This process continuously communicates with the robot's controller (i.e., FlexPendant) using its proprietary SDK API to a) collect information about the robot, and b) perform actions specified by the user's scenarios.

The execution procedure is depicted in Figures 5 and 6, which begins as the user triggers a launch event from the authoring client. This action sends a request to the application server, which, in turn, spawns the UAL process. During the *Given* phase, the UAL interacts with the robot by requesting and awaiting a status update. Upon receiving the robot's response, the process evaluates the *Given* condition. If the condition is evaluated as false, the preconditions for running the scenario are not fulfilled, hence the gesture recognition and robot command stages are bypassed and the process moves directly to the next scenario. For each scenario in the sequence where the *Given* condition is true, the process advances to the *When* clause, at which point the system waits for user gestures. Once the expected gesture is recognized, the UAL sends commands to the robot to perform specific actions, such as executing a predefined trajectory. The robot subsequently reports its outcome back to the UAL, which is then relayed to the server and communicated to the client. Upon completion of all scenarios, the UAL process terminates and the server notifies the client of the process's conclusion, which updates the view to unlock new actions for the user, including the ability to relaunch the code or re-engage programming.

5. Experiment

This section presents the experiment conducted to address our research questions. It provides an overview of the experimental procedure and summarizes the key findings. The experiment involved one to two proctors and a single participant working in the experimental environment.

5.1. Before the experiment

5.1.1. Participants recruitment

Participants were primarily individuals from academia or professionals with experience in companies. Recruitment was conducted using flyers for advertisement and through networking and outreach. The majority of participants shared a similar profile of being educated individuals. Specifically, most participants from academia were PhD students from the University of Southern Denmark (SDU) in Odense, Denmark.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

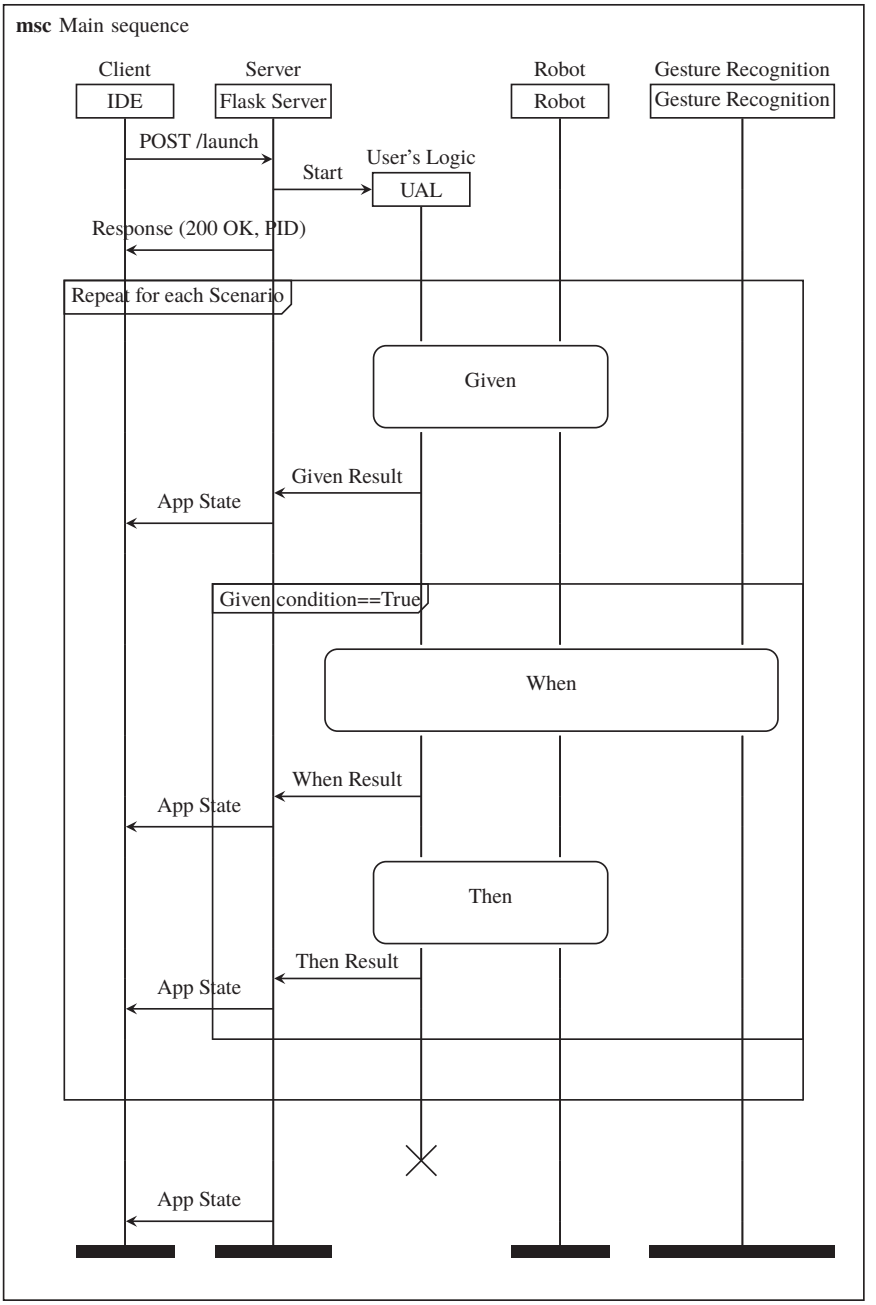


Figure 5. *Main execution sequence of user's application logic.*

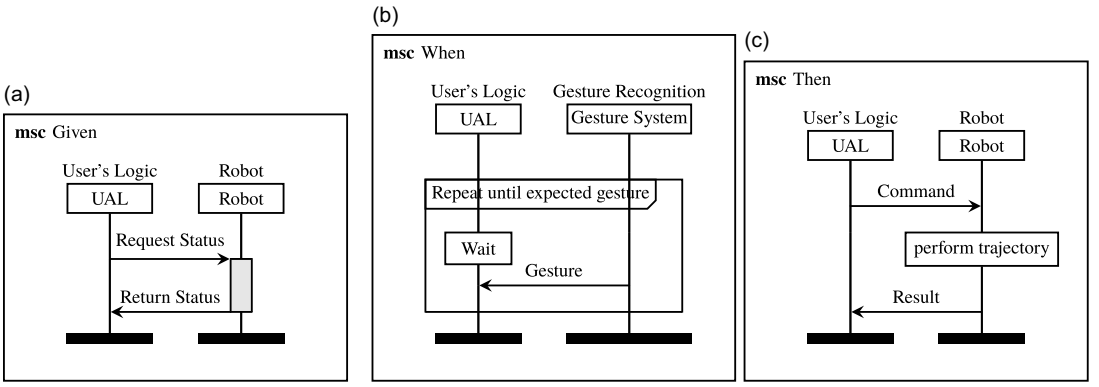
5.2. During the experiment

5.2.1. Experimental setup

As illustrated in Figure 7, the working environment consists of a collaborative UR5 robot mounted on a mobile platform, which remains stationary and unused for this experiment. A cubic object, built with LEGO bricks, is placed at the centre of the table at a location marked as *"Base"*. The workstation is divided into two separate areas, labelled as *Left* and *Right*.

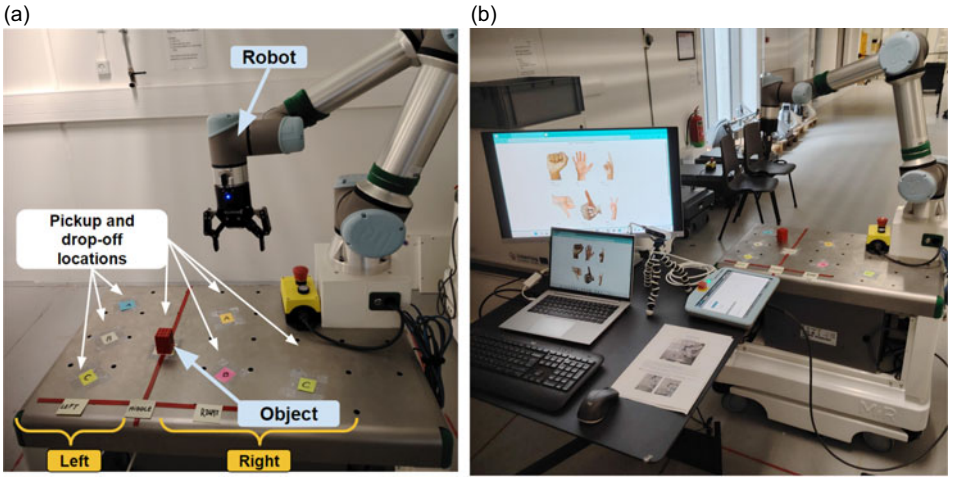
Each area includes three marked locations for object placement, labelled *"A"*, *"B"* and *"C"*, providing the user with multiple positioning options.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press



Given sequence. *When* sequence. *Then* sequence.

Figure 6. *Sub-sequences of execution for the user's application logic: (a) Given, (b) When, (c) Then.*



Physical workstation. *Participants' workspace with robotic platform.*

Figure 7. *Overview of the experimental setup.*

5.2.2. Tools and documents used during the experiment

Table I and Figure 8 below outline the tools and documents provided to the participants and used by the proctors to facilitate the flow of the experiment.

5.2.3. Onboarding and experimental procedure

The experiment was conducted in a structured manner to ensure the protocol adherence and participant comfort to engage with the testing interface. Upon arrival, each participant was welcomed by the proctor, who provided a detailed explanation of the session's objectives and the scientific goals related to the usability testing of the interface.

Participants were guided through the necessary documents, including the consent form and the first survey, which they were required to complete before beginning the task. The first survey, shown in Figures A1 and A2 in Appendix A, is a demographic survey consisting of eight questions. These questions collected personal background information, such as age and gender, as well as other key factors essential for the subsequent data analysis. The final three questions focused on the participants' overall professional experience (in years), programming experience and experience with robot programming (in years). For the first and third of these questions, participants selected one of four categories: *Less than one year*, *Between 1 and 3 years*, *Between 3 and 5 years* or *More than 5 years*. In the second

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

Table I. *Resources provided to participants and proctor.*

Participants' tools	**Proctors' tools**
Paper-format documents	Paper-format documents
Task instructions (A4 document, see Figure 10b),	Consent form,
NASA-TLX survey,	Demographic survey,
SUS survey	Proctor notes
Hardware tools (see Figure 8b)	Recording tools
Webcam/RealSense camera,	GoPro camera (see Figure 8b),
Monitor,	Video recordings,
Mouse,	Screen recordings,
Robotic platform and LEGO block	Timer for task duration
Software tools	
Web interface for gesture labelling (see Figure 8a),	
BDD-based visual editor (see Figure 8c)	

question, participants were asked to rate their programming proficiency on a scale from 1 to 5, where 1 indicated *less proficient* and 5 indicated *most proficient*.

After completing the survey, the proctor provided an introductory presentation, lasting *15 min*, during which the available tools and the BDD-driven Blockly visual editor were explained. Following the introduction, participants were guided on how to assign customized names to gesture signs using the web interface shown in Figure 8(a). They were informed that they were free to choose any name they preferred, in any language they felt comfortable with, and that there would be no need to remember these names. Instead, the customized names would be displayed alongside images of the signs in the corresponding blocks during the task.

5.2.4. Task selected for the experiment

The experiment involved programming the cobot in three sequential steps to manipulate a cubic object, move it within the workstation and place it in specific locations. At the beginning of the experiment, the cobot's configuration is random, positioned over the workstation with the tool facing the plane of the table, as illustrated in the first subfigure of Figure 9.

The task is structured into three main phases. Figure 9 illustrates the intermediate and final arrangements of the object and the robot after each step.

1. ***Positioning and initial grasp***: In the initial step, the user positions the robot at the central starting point of the workstation. Once the robot reaches this point, the user programmes it to grasp the cubic object located at the table's centre, referred to as the *"Base"*.
2. ***Placing the cube at position right B***: After grasping the object, the robot is guided by the user to the right side of the workstation, where it places the object at position *"B"*.
3. ***Moving the cube to location left A***: In the final phase, the user programmes the robot to pick up the object from position *"B"* and move it to position *"A"* on the left side of the workstation, completing the sequence.

The aforementioned task was selected for the experiment because it involves performing a series of fundamental actions commonly found in robotics, such as positioning, object manipulation and placement within a robotic framework. Additionally, the task was adapted to assess the ease of understanding and intuitiveness of the proposed BDD-based Blockly visual interface for the participants.

5.2.5. Task execution and observation

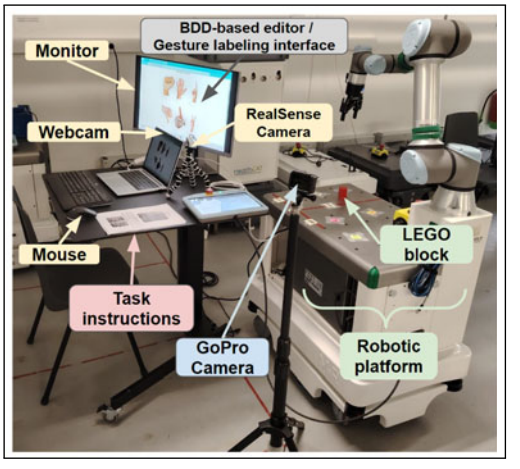
Participants were free to use the *45 min* allocated for the experiment as they wished to solve the task. The proctor was not permitted to assist with solving the task but could address participant inquiries unrelated

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

(a)

Web interface for gesture labelling.

(b)



Experimental platform with associated tools

(c)

BDD-based Blockly visual editor with an example of blocks assembled to pick an object.

Figure 8. *Visual representation of the tools listed in Table I.*

to task completion, such as clarifications regarding the instructions, the task description, or support with technical issues. If participants indicated that they had completed the task but had not executed their programme, the proctor requested them to run it to verify their solution. If the programme did not solve the task successfully and time was still available, participants were allowed to make further attempts to improve their solution.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

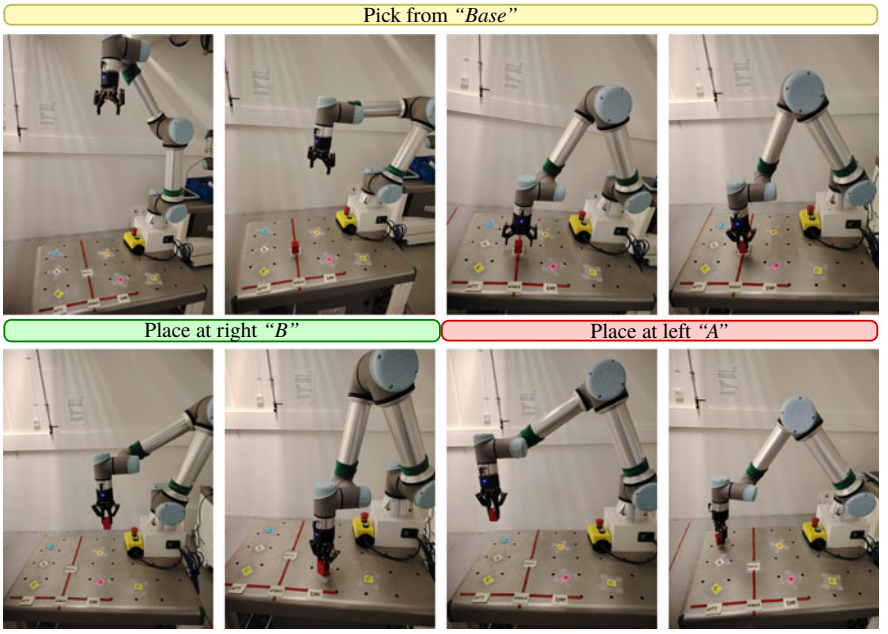


Figure 9. *Task overview; three phases of robot object manipulation.*

5.2.6. Data annotation

Throughout the session, data collection was carefully conducted for subsequent analysis. Video and screen recordings were initiated with the participant's consent for a detailed review of interactions with the interface. A proctor supervised the experiment at all times and took notes on participant performance and behaviour. Figure B1, Appendix B illustrates the spreadsheet used by the proctor during the experiment for data collection.

The duration of each of the three steps of the experiment was recorded using a digital clock, along with the start and end times. The proctor also tracked specific events during the experiment, including the number of times participants executed their programme solution. Once a participant had completed the task, the proctor reviewed the code and recorded the number of scenarios used in their solution.

5.3. Post-experiment feedback

After completing the experiment, participants were asked to fill out two post-experiment surveys: the NASA-TLX [36] survey and a System Usability Scale (SUS) [37] survey.

The NASA-TLX survey is a standardized tool used to measure perceived workload across multiple dimensions, such as mental and physical demand, temporal demand and frustration level. This allows for a nuanced assessment of how mentally and physically demanding participants found the tool to use.

The SUS survey, on the other hand, focuses specifically on usability, providing a simple but reliable measure of how effectively and efficiently participants could navigate the tool. These two surveys were chosen to offer complementary insights: the NASA-TLX captures the cognitive and physical effort involved in using the tool, while the SUS survey evaluates overall usability and user satisfaction. To provide clarity on the assessment process, Figure C1 in Appendix C displays the SUS survey forms used in the study.

6. Results

6.1. Analysis of participants' demographics

The data collected from the experiment surveys were processed to perform a qualitative analysis and assess whether the objectives of the study were met.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

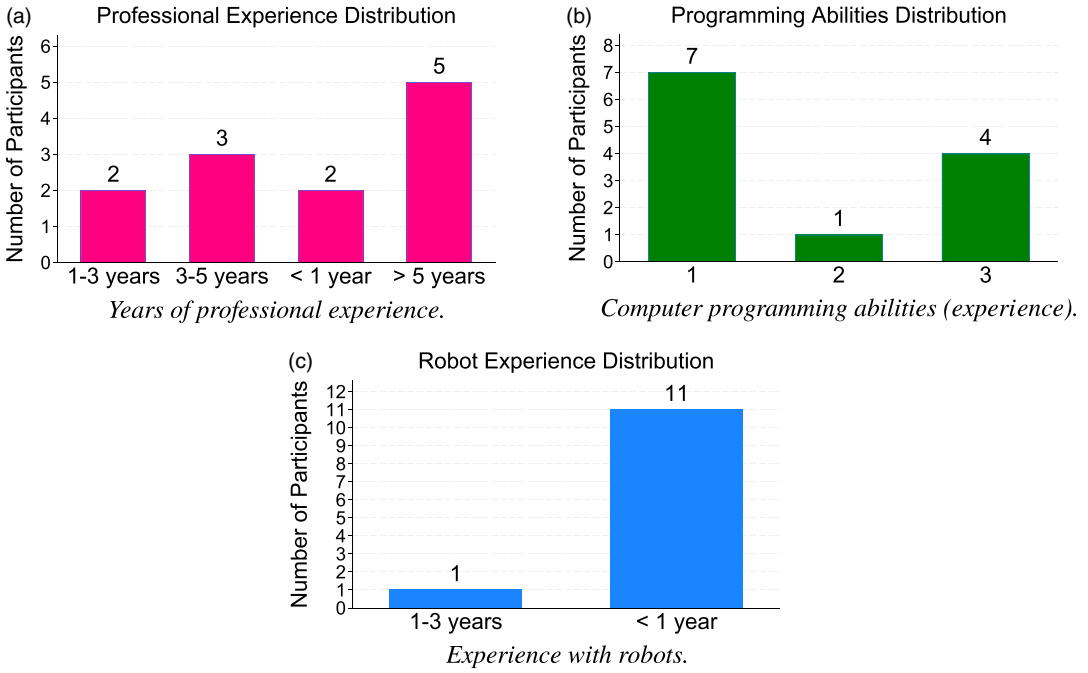


Figure 10. *Survey results of participants' background and experience.*

For this research, the last three questions of the demographic survey mentioned in Section 5.2.3, were selected and were particularly important:

1. *How many years of professional experience do you have?*
2. *How would you evaluate your computer programming abilities?*
3. *Do you have experience working with robots?*

Using the participants' responses, the following graphs were generated to visualize the distribution of participants within each category.

Upon examination of the results, the graph Figure 10(a) demonstrates the diversity of professional experience in all groups of participants. This is particularly beneficial as it ensures a diverse range of working profiles within the study. The inclusion of participants with varying levels of professional experience offers a broader perspective on how different backgrounds might influence both the effectiveness and ease of use of the tool. This diversity enables an evaluation of whether participants with more professional experience are able to use the tool more effectively. In turn, this helps assess its user-friendliness and overall performance across different experience levels.

Analysing the graph Figure 10(b), it can be observed that most participants rated themselves at the lowest level for programming experience, indicating no experience, while a smaller group assigned themselves a score of 3, suggesting minimal or introductory knowledge. This indicates that while most participants had limited or no programming experience, some had basic familiarity with technical tools. Including both groups allows for a more comprehensive evaluation of the tool's usability across different levels of technical familiarity.

Finally, the last graph shows that, except for one participant, all others had a low exposure, or limited experience with robots (less than one year), which helps to provide a clearer picture of how the tool performs for individuals who are unfamiliar with robotic technology. It allows us to assess how intuitive and effective the tool is for those with little to no experience in a human-robot interaction (HRI) context.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

Table II. *Evaluation variables created from SUS survey.*

Category	Description	Variables	Participant questions
Task/System functionality	Variables in this category evaluate participants' perception of the system's task execution and integration	*Workflow*	"I found the workflow for implementing new tasks for the robot to be simple"
		EasyUse	"I thought that the programming tool for the robot was easy to use"
		GestureRecog	"I found that the robot, the gesture recognition module and the programming tool are well integrated"
		Consistency	"I found that the robot consistently responded to body gestures while using the programming tool"
Internalization	Variables in this category evaluate the participants' interest in using the tool for future tasks and integration into their routine work/life	*LikeUse*	"I think that I would like to use this programming tool frequently for automating other tasks around me"
Accessibility	Variables in this category assess how easily participants feel they can access and use the tool, even without technical support	*ProgramNew*	"I think that I could programme new scenarios for the robot without the support of a technical person"
		Learnable	"I would imagine most people would learn to use the programming tool for robots very easily"
		UseWithout-Learning	"I could use the programming tool without having to learn anything new"
Comfort of user	Variables in this category evaluate how comfortable and intuitive participants found the tool during the experiment	*Intuitive*	"I found the programming tool to be very intuitive"
		Comfort	"I felt comfortable and in control while using scenarios to program the robot's behaviours"
		Rating	"In general terms, I would rate my experience using the programming tool as. . ."

6.2. Hypothesis formulation

To initiate the data analysis, testing hypotheses were defined to evaluate the tool's suitability for non-technical users within an industrial HRI context. The hypotheses to be examined are as follows:

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

- **Null Hypothesis 1 (H_{0_1}):** *Users' programming experience level does not significantly impact the tool's intuitiveness in completing robot task assignments without extensive training.*
- **Alternative Hypothesis 1 (H_{1_1}):** *Users' programming experience level significantly impacts the tool's intuitiveness in completing robot task assignments without extensive training.*
- **Null Hypothesis 2 (H_{0_2}):** *Users' general professional experience level does not significantly impact their perception of the tool's intuitiveness in completing robot task assignments without extensive training.*
- **Alternative Hypothesis 2 (H_{1_2}):** *Users' general professional experience level significantly impacts their perception of the tool's intuitiveness in completing robot task assignments without extensive training.*

For the tool to be viable for broad deployment in industrial HRI applications, it should be intuitive for users without programming backgrounds. Therefore, the analysis will investigate whether programming experience correlates with participants' ratings of the tool's intuitiveness and other features, which are determined by the values of the associated variables.

Additionally, to explore whether other dimensions of expertise play a role, a secondary hypothesis was defined to assess the impact of general professional experience in the industry. This analysis seeks to determine whether those with longer or shorter industrial experience find the tool more intuitive, regardless of their programming knowledge. This added layer of investigation allows for a broader understanding of user background influences on the tool's usability in real-world settings, focusing specifically on how familiarity with industry workflows might impact perceptions of the tool.

Based on the answers to the SUS survey (Table V), several variables were created for data analysis and organized into distinct categories. Table II presents these variables by category, with accompanying descriptions of each category's evaluation goals and the specific questions asked to the participants. Each category targets a particular aspect of the tool and its variables capture participants' feedback on related experiences.

Data analysis was conducted using Stata [38], a statistical software for both quantitative and qualitative analysis. To evaluate the first alternative hypothesis (H_{1_1}) and potentially reject the associated null hypothesis (H_{0_1}), the participant sample was segmented based on *programming experience*. This categorization was based on responses to the previously introduced question 2 from the demographic survey (see Section 6.1). A new independent variable, *prog_experience_group* was created to categorize participants based on their self-assessed programming experience (Figure 10b).

Two categories were defined: participants who rated their programming experience as 1 or 2 (presumably indicating no programming skills) and those who rated it as 3 (presumably indicating notions or low programming skills). Subsequently, analysis of variance (ANOVA) tests were conducted for each variable listed in Table II (e.g., Workflow, EasyUse, ProgramNew, etc.) to determine whether statistically significant differences existed across the two programming experience groups. In these tests, the variables listed in Table II served as the dependent variables, while *prog_experience_group* served as the independent grouping variable. Each ANOVA test produced an F-statistic and *p*-value to indicate whether the differences between the two groups were significant. Statistically significant *p*-values (e.g., below 0.05) would suggest that professional experience has a meaningful effect on the mean ratings for a given variable.

In a secondary analysis, the participant sample was segmented based on *years of professional experience*, to evaluate whether general professional experience had a significant effect on participants' ratings of various aspects of the system. The purpose was to evaluate the second alternative hypothesis (H_{1_2}) and potentially reject the associated null hypothesis (H_{0_2}). Four experience groups were created (see Figure 10a): participants with less than one year of experience, those with 1–3 years, 3–5 years, and more than 5 years. Each group was assigned a corresponding numerical value (1 through 4) through the creation of a new variable, *experience_group*. Similar to the first analysis, a second wave of ANOVA tests were conducted for each variable in Table II, with these variables as dependent variables and *experience_group* as the independent grouping variable.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

Table III. *ANOVA results for each dependent variable listed in Table II based on programming experience.*

Variable	*F*-value	*p*-value
Workflow	0.48	0.5059
EasyUse	0.09	0.7760
GestureRecog	0.61	0.4543
Consistency	0.00	1.0000
LikeUse	0.39	0.5465
ProgramNew	0.06	0.8105
Learnable	0.39	0.5465
UseWithoutLearning	2.97	0.1156
Intuitive	0.37	0.5564
Comfort	0.61	0.4543
Rating	0.28	0.6097

6.3. ANOVA results

6.3.1. Results on programming experience groups

From the initial set of tests, the F-value and p-value for each ANOVA output were gathered in Table III.

- ***Results reflecting high p-values (no statistical significance):*** All variables associated with task and system functionality displayed p-values close to or above 0.5, which is far above the threshold for statistical significance (0.05). For instance, *Workflow* resulted in a p-value of 0.5059, indicating that there is no significant difference in ratings of the system's task-related workflow between groups with different levels of programming experience. Likewise, *EasyUse* ($p = 0.7760$) and *GestureRecog* ($p = 0.4543$) show no significant impact of programming experience on perceived ease of use or gesture recognition performance. Interestingly, *Consistency* displayed an F-value of 0 and a p-value of 1, suggesting that ratings were identical across groups. These results together indicate that programming experience does not significantly affect participants' views regarding the system's ability to execute tasks and provide integrated functionality. Regarding internalization, the *LikeUse* variable, with a p-value of 0.5465, suggests that programming experience does not significantly impact participants' interest in using the tool regularly for future tasks. Both groups show similar tendencies toward frequent use, independent of prior programming experience. For *ProgramNew* ($p = 0.8105$) and *Learnable* ($p = 0.5465$), no significant differences in accessibility ratings were observed between experience groups. Both rated the system's learnability similarly, indicating that programming experience does not heavily influence perceptions of accessibility or the need for technical assistance. However, results for *UseWithoutLearning* will provide further insights on this aspect. The *Intuitive* variable ($p = 0.5564$) shows no significant difference between groups in terms of intuitive use. Similarly, *Comfort* ($p = 0.4543$) and overall *Rating* ($p = 0.6097$) indicate that programming experience does not particularly affect participants' comfort or general satisfaction. Both experienced and inexperienced users found the tool comfortable and intuitive.
- ***Results with lower p-value (approaching significance):*** For *UseWithoutLearning* ($F = 2.97$, $p = 0.1156$), the results were closer to significance, with a p-value closer to the 0.05 threshold. Although not statistically significant, this finding suggests a slight trend that programming experience could influence ratings. While most participants felt extensive training unnecessary, additional testing with a larger sample might clarify whether programming experience subtly affects perceptions of ease without additional learning.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

Table IV. *ANOVA results based on professional experience.*

Variable	*F*-value	*p*-value
Workflow	4.00	0.0519
EasyUse	0.34	0.8001
GestureRecog	1.29	0.3430
Consistency	0.38	0.7696
LikeUse	1.04	0.4268
ProgramNew	1.56	0.2737
Learnable	0.99	0.4465
UseWithoutLearning	7.38	0.0108
Intuitive	0.70	0.5797
Comfort	0.87	0.4961
Rating	0.67	0.5927

6.3.2. Results on professional experience groups

For a second analysis, the purpose was to evaluate whether professional experience has a significant effect on participants' ratings of various aspects of the tool/system. The F-value and p-value for each test were gathered in Table IV.

- ***Results with high p-values:*** The F-value of 4.00 for *Workflow* suggests some variation between groups in how they rate the *Workflow* feature. The p-value of 0.0519 is close to the 0.05 threshold, indicating a trend toward significance, meaning there may be differences in how participants with varying levels of experience evaluate the task workflow. Further investigation with a larger sample size might be needed to confirm this. For *EasyUse*, the F-value of 0.34 indicates a very small variance between groups and the p-value of 0.8001 is well above the typical 0.05 threshold, suggesting that professional experience does not impact perceptions of the tool's ease of use. Similarly, the F-value of 1.29 for *GestureRecog* and 0.38 for *Consistency* suggests minimal variance between groups. High p-values of 0.3430 and 0.7696 respectively indicate no significant difference in the evaluation of the tool's gesture recognition system or consistency based on professional experience. For the *LikeUse* variable, the p-value of 0.4268 suggests that participants' interest in regularly using the tool for future tasks is consistent across groups and independent of professional experience. An F-value of 1.56 for *ProgramNew* suggests moderate variance between groups, but the high p-value of 0.2737 indicates no significant difference in how participants rate the tool's *ProgramNew* feature. The *Learnable* variable, with a p-value of 0.4465, shows both groups rated the system's learnability similarly, suggesting that professional experience does not significantly influence perceptions of accessibility or the need for technical assistance. The outcomes of the three variables evaluating user comfort and intuitiveness with the tool, *Intuitive* ($p = 0.5927$), *Comfort* ($p = 0.4961$), and *Rating* ($p = 0.5927$), show general satisfaction across participants. These non-significant p-values suggest no notable differences between groups in terms of intuitive use, indicating that professional experience does not impact comfort and ease of use.
- ***Results with low p-values:*** For *UseWithoutLearning*, the F-value of 7.38 indicates significant variation between groups. The p-value of 0.0108, below the 0.05 threshold, confirms a statistically significant result. Professional experience significantly influences participants' perceptions of the ability to use the tool without prior learning, with more experienced participants rating this feature differently than less experienced ones.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

Table V. *Mean scores and standard deviations for usability scale responses from this study and the previous study. Values from the previous study are shown in violet.*

		This study		Previous study	
	Question (Q)	**Mean**	**SD**	**Mean**	**SD**
1	I think that I would like to use this programming tool frequently for automating other tasks around me	4.00	0.95	4.17	0.99
2	I found the workflow for implementing new tasks for the robot to be simple	4.25	0.87	4.50	0.87
3	I thought that the programming tool for the robot was easy to use	4.42	0.67	4.58	0.64
4	I think that I could programme new scenarios for the robot without the support of a technical person	4.08	0.79	3.92	0.86
5	I found that the robot, the gesture recognition module and the programming tool are well integrated together	4.58	0.51	4.42	0.49
6	I found that the robot consistently responded to body gestures while using the programming tool	4.00	0.85	4.25	0.72
7	I would imagine most people would learn to use the programming tool for robots very easily	4.00	0.95	4.08	0.76
8	I found the programming tool to be very intuitive	4.33	0.65	4.25	0.72
9	I felt comfortable and in control while using scenarios to program the robot's behaviours	4.42	0.51	4.25	1.23
10	I could use the programming tool without having to learn anything new	3.58	0.90	3.83	0.99
11	In general terms, I would rate my experience using the programming tool as:	8.67	1.50	7.50	2.43

6.4. Comparison with previous study

This experiment built on the previous study [13], replicating its main objective, evaluating an end-user programming framework for collaborative robots, while introducing enhancements to tools and task design. Unlike the first study, which used a standard programming editor to write scenarios using a BDD-based DSL, the second study employed a web-based Blockly interface. This allowed users to assemble scenarios using drag-and-drop blocks with preformatted sentence structures and gesture images to simplify interaction and eliminate manual coding. The first experiment involved 12 participants with entry-level programming skills performing a basic pick-and-place task with an ABB Dual-Arm YuMi robot in Sweden, while the second involved 12 new participants with little to no programming experience in Denmark, using a UR5e robot. The task expanded to a more complex, sequential pick-and-place operation with three phases, positioning, grasping, and placing, requiring users to build complete scenarios rather than trigger pre-coded adjustments. Both studies maintained similar durations and usability assessments using System Usability Scale (SUS) scores, but the second added NASA-TLX workload evaluations, reflecting reduced cognitive burden for non-expert users. This experiment reused the same SUS survey from the previous study to calculate the mean scores and standard deviations for each question (Table V) and evaluate the user experience and the usability of the tool. By comparing these metrics from both experiments, this study assesses whether the enhanced tool improved usability.

The comparison of results reveals improvements over the first experiment. Across the values in Table V, the mean scores for most questions are higher in the second experiment, indicating a generally more positive user experience with the new tool. For example, the results for Question 2, which asks about the simplicity of the workflow for implementing new tasks, showed a mean value of 4.25 in the new experiment. Despite a slight decrease from the first experiment (mean = 4.50), the results

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

Table VI. *Mean scores and standard deviations for each NASA-TLX subscale.*

NASA-TLX subscale	Mean score	Standard deviation
Mental demand	38.33	19.08
Physical demand	4.58	4.77
Temporal demand	15.91	19.75
Performance	18.75	24.25
Effort	25.00	15.81
Frustration	8.75	12.44

suggest that participants still found the workflow simple and easy to understand, even with the enhanced interface. In contrast, the results for Question 5 showed an increase in the mean score, from 4.42 to 4.58. This suggests that the new tool was perceived as better integrated and contributed to a smoother user experience.

Question 1, which asked participants whether they would like to use the programming tool frequently, showed a slight decrease from 4.17 (SD = 0.99) in the previous experiment to 4.00 (SD = 0.95) in the new experiment. Another notable difference is seen in question 3, which assesses the ease of use of the programming tool. In the first experiment, the mean was 4.58 (SD = 0.64), while in the new experiment, it slightly decreased to 4.42 (SD = 0.67). Despite the mean scores slightly dropping for both questions, the standard deviations remained low, signifying that the responses were more consistent in the second experiment.

A notable accomplishment observed in this evaluation is the tool's improved accessibility. Participants noted that the interface and its features were intuitive enough to use effectively without requiring extensive learning. Indeed, the mean score for question 4 increased from 3.92 in the original experiment to 4.08 in the new experiment. Additionally, the results indicate a clear increase in satisfaction. Question 11, which asked participants to rate their overall experience with the programming tool, showed a significant improvement, with the mean score rising from 7.50 in the first experiment to 8.67 in the new experiment.

6.5. Workload assessment results using NASA-TLX scores

The NASA-TLX [39] uses a 20-point scale for each of the six workload subscales: Mental Demand, Physical Demand, Temporal Demand, Performance, Effort and Frustration. Participants rate each subscale by marking one of 20 boxes, with each box corresponding to a score increment of 5 points, ranging from 0 to 100, with higher values indicating greater demand or frustration. The first box represents a score of 0 and the last box represents a score of 100.

For further analysis in this study, ratings from 12 participants were collected and gathered in Table VI. The table presents the mean scores and standard deviations for each subscale, calculated based on the ratings provided by all participants.

The dimensions with the highest mean scores were *Mental Demand* and *Effort*, both surpassing the first quartile. The *Mental Demand* dimension showed a mean score of 38.33, indicating that participants generally experienced a moderate level of mental workload. However, the high standard deviation of 19.08 suggests notable variation in participants' experiences with mental effort during the task.

For *Effort*, the mean score was 25.00 with a standard deviation of 15.81, indicating that participants felt a moderate level of effort was required to complete the task. The marginally lower standard deviation suggests that participants' ratings for the required effort were slightly more consistent.

The remaining dimensions recorded mean scores below the first quartile threshold. *Physical Demand* had the lowest mean score, at 4.58, suggesting that participants did not perceive the task as physically demanding. The standard deviation of 4.77 shows that while some participants perceived slightly more

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

Table VII. *Quantitative data collected: Task timing and code interaction.*

Participant	Total duration	Time (Start–Task 1)	Time (Task 1–2)	Time (Task 2–3)	Code runs	Blocks count
1	16 min	4 min	8 min	4 min	1	7
2	30 min	4 min	9 min	17 min	6	5
3	53 min	30 min	9 min	14 min	4	8
4	33 min	18 min	4 min	11 min	7	7
5	30 min	7 min	6 min	17 min	2	7
6	16 min	6 min	3 min	7 min	3	5
7	23 min	5 min	9 min	9 min	7	6
8	9 min	2 min	4 min	3 min	1	7
9	24 min	2 min	6 min	16 min	1	6
10	11 min	3 min	2 min	6 min	2	6
11	36 min	20 min	5 min	11 min	2	8
12	21 min	5 min	7 min	8 min	3	8

physical demand than others, physical effort was generally minimal. The participants found the task slightly time-pressured by referring to the mean score for *Temporal Demand*, though a large standard deviation indicates considerable variability in how participants experienced this aspect.

The second-lowest mean score was for *Frustration*, at 8.75, with a standard deviation of 12.44. This low score indicates that frustration was not a significant factor for most participants. However, the substantial standard deviation indicates that some participants experienced higher levels of frustration than others during the task.

Regarding *Performance*, participants generally felt moderately positive about their performance, but this dimension had the highest standard deviation (24.25), meaning a broad range of responses in participants' self-assessment of their performance.

6.6. Timing metrics in task performance analysis

The quantitative data collected during the experiment are presented in Table VII, which includes the overall duration required to complete the task, tracked by proctors for each session through start and end times. Additionally, the table shows the time taken by each participant to complete the three main phases or milestones: *Positioning and initial grasp*, *Placing the cube at position right B* and *Moving the cube to location left A*, as detailed in Section 5.2.4 and illustrated in Figure 9.

Additional metrics include the number of programme executions on the physical system, referred to as *Code Runs* in Table VII and the total number of scenario blocks assembled to accomplish the task, labelled *Blocks Count*.

From the collected information, we can gain several perspectives on the tool's usability and efficiency.

- ***Total Task Duration and Consistency:*** The total duration for completing the task varies considerably, from as low as 9 min(Participant 8) to as high as 53 min (Participant 3). This wide range suggests significant variability in participant familiarity or comfort with the task and tool. It also indicates that certain participants took longer to familiarize themselves with the task requirements. The fastest completion times, particularly those under 20 min, suggest that the tool can be learned, understood and effectively applied by some participants with minimal difficulty. However, additional factors outside of the proctors' control, such as participants' physical state (fatigue) or emotional readiness, may also have influenced completion times. Participants who were not fully focused or mentally prepared may have encountered greater challenges during task execution.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

- ***Milestone Times:*** The time taken for each phase also varies widely. For example, times for Task 1 completion range from 2 min (Participant 9) to 30 min (Participant 3). Such differences might reflect individual differences in problem-solving approaches or the need for more structured guidance in the initial introduction. Phases 2 and 3 show shorter times overall, which could mean that participants found these milestones simpler or became more familiar with the tool as they progressed.
- ***Tool Usability and Code Runs:*** The number of code runs per participant ranges from 1 to 7, with an average of around 3-4 runs. Higher numbers of code runs, such as for Participant 4 (7 runs), may indicate trial-and-error in reaching the solution. Participants with fewer code runs generally have lower total durations, indicating a potential link between tool ease of use and efficiency.
- ***Scenario Blocks Assembled:*** The assembled blocks range from 5 to 8, with no participant needing a large number of blocks, implying that the tool allows participants to accomplish the task with a manageable set of actions. However, variations here may still indicate differing levels of optimization or clarity in problem-solving approaches.

7. Discussion

This study aimed to evaluate a new end-user programming tool designed to cobot task definition BDD-based DSL and a Blockly interface. The results obtained from the user study, combined with statistical analyses, provide useful insights into the effectiveness of the proposed framework, answering our research questions and addressing the challenges outlined in the introduction. Hereafter we discuss our findings in light of the RQs.

RQ1: How does prior programming experience affect non-expert users' perception of the tool's intuitiveness and usability? The analysis, particularly through ANOVA testing, indicated that prior programming experience did not have a statistically significant impact on the tool's usability, supporting the hypothesis that the tool is accessible to users without programming expertise. However, the variable *UseWithoutLearning* did show, for both ANOVA tests, a trend where programming experience may influence participants' perceptions of the necessity for prior training, suggesting that some users might feel more confident using the tool independently, while others may need a learning phase. This result implies that while the tool is designed for ease of use, there is still room for improvement in onboarding users without prior technical experience.

RQ2: How does general professional experience influence users' perceptions of the tool's intuitiveness and their ability to complete robot task assignments? Similarly, professional experience did not significantly affect ratings across most tool features. However, the variable *UseWithoutLearning* again stood out, showing that more experienced participants felt they could use the tool with less training. This suggests that professional experience, especially in technical domains, might reduce the perceived learning curve.

RQ3: What aspects of user experience (e.g., intuitiveness, ease of learning, system integration) show the most significant improvements in the new tool? The results show significant improvements in workflow simplicity, ease of integration and overall user satisfaction, addressing key concerns from the previous study. The new tool's enhancements, such as the introduction of a drag-and-drop interface for scenario creation, eliminating the need for manual coding, real-time feedback during code execution, a simplified gesture labelling interface and the development of a web-based platform, have significantly improved usability ratings. However, there was some fluctuation in specific usability metrics, indicating that the tool, while generally positive, could benefit from further refinement in certain areas, such as task guidance and feedback clarity during the early stages of use.

RQ4: How do the System Usability Scale (SUS) metrics and user feedback in the current study validate or challenge the qualitative improvements identified in the previous research? The SUS metrics confirm that the new tool provides a clear improvement in user satisfaction compared to the previous system. Key enhancements, such as better task flow simplicity and a more intuitive interface, were reflected

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

in higher mean scores for several survey questions. For instance, the overall user experience rating (Q11) showed a significant increase from 7.50 to 8.67 out of a maximum of 10, indicating greater participant satisfaction. Additionally, integration between the robot, gesture recognition module and programming tool (Q5) improved, with the mean score rising from 4.42 to 4.58 out of 5. While some areas, like ease of use (Q3) and workflow simplicity (Q2), saw slight decreases in mean scores, these were minor and accompanied by lower standard deviations, suggesting more consistent participant feedback in the new study. Importantly, participants felt more confident using the tool independently (Q4), with the mean score increasing from 3.92 to 4.08 out of 5. These findings validate the enhancements introduced in the system and demonstrate meaningful progress in improving its usability.

7.1. Insights from task performance metrics analysis

The task performance metrics provide additional context for understanding the tool's usability. Task completion times varied significantly, ranging from 9 min for the fastest participant to 53 min for the slowest, reflecting wide variability in user familiarity and efficiency. The number of code runs ranged from 1 to 7, with higher runs often indicating trial-and-error approaches, particularly for participants who took longer. Despite this variability, the number of blocks assembled was consistent across participants, ranging from 5 to 8 blocks, demonstrating that the tool supports task completion with a manageable set of actions.

The NASA-TLX workload assessment showed mental demand and required effort primarily affect task execution. The high mental workload observed originates from the combination of a new tool, a new environment, and a novel task, which required significant reflection even for a roboticist. An important factor is that the robotic setup and laboratory environment were unfamiliar to all participants. Even though they had achieved high levels of education, their expertise lay in different fields like literature or chemistry, not robotics. Humans are known to retain only about 80% of information presented to them initially, as argued by Ericsson *et al.* [40], meaning regular practice and training are required to master a new tool or habit. In this experiment, the tool was introduced in 20 min, which is relatively brief. Participants needed mental effort to recall the grammar rules, understand the unfamiliar pick-and-place task, a concept they had never encountered before, and adapt to working with a robot, increasing their cognitive load.

If the tool was deployed in industrial settings, repeated use would reduce this burden. As end-users become familiar with the tool through practice and the robotic tasks become routine, the mental demand should decrease substantially, diminishing initial cognitive load with habitual use.

It is interesting to consider the threshold of this tool, where experienced users are better suited for robot interaction. As task complexity increases, such as when more scenarios are added, mental demand also rises. However, novice users may experience increased mental demand due to discomfort with technology or fear of robots, regardless of task complexity.

Another potential limitation could arise from the complexity of task definition. Many tasks in robotics, such as pick-and-place, seem conceptually simple but remain challenging due to hardware and environmental constraints. In industrial settings, however, such tasks are typically repetitive and structurally straightforward, making their high-level definition relatively simple. A potential limit might arise in more complex multi-robot setups, which require the construction of numerous scenarios.

Thus, the main barrier may not be task complexity, but rather technological unfamiliarity. Future studies with diverse user backgrounds and multi-robot setups could help clarify this threshold.

7.2. Key improvements and remaining challenges

The optimizations introduced in this study have successfully addressed several key limitations identified in earlier work. For instance, compared to the CAPIRCI system [21], which relied on unstructured natural language for task definitions, the new tool's use of a structured DSL based on BDD ensures more clarity and reduces ambiguity in task definitions. This change directly addresses identified limitations in task precision and flexibility. Furthermore, the integration of a Blockly-based interface with modular, drag-and-drop blocks enhances usability by making the programming process more intuitive and visually

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

clear. The new tool also expands task complexity, moving beyond most of the basic workflows presented in the other tools discussed in existing research, to accommodate more dynamic industrial scenarios, involving sequential workflows like positioning, grasping and placing.

In comparison to systems like EUD-MARS [23] and visual programming tools, the new tool demonstrates a distinct advantage in minimizing cognitive load. Participants could seamlessly create robot actions by assembling colour-coded blocks, accompanied by gesture-specific images and real-time feedback. This design offers a more intuitive and user-friendly experience compared to many visual programming systems, which often depend on abstract representations that non-expert users may find difficult to interpret. Moreover, the integration of gesture-based commands introduces a multimodal dimension, further enhancing both accessibility and user engagement.

However, challenges remain regarding task flexibility and real-time adaptability. Like many tools in the existing research, the system requires further improvements in handling complex tasks, particularly in areas such as error recovery and real-time feedback during execution. This was evident in the variability observed in task completion times, where participants encountered difficulties in the early stages. Enhancing task instructions, onboarding processes and feedback mechanisms could further streamline the user experience and improve overall efficiency.

These usability challenges also raise questions about the influence of participant demographics on the study outcomes. Specifically, we believe that the relatively homogeneous educational background of our participant group, comprising primarily engineering-focused PhD students and students from non-scientific fields such as literature and business, directly influenced their perception of the tasks. Our group lacked diversity, consisting mainly of educated individuals from academia. Research suggests that academics develop improved cognitive skills compared to less-educated individuals, such as prolonged focus, rapid memorization, and efficient reasoning, due to the demands of academic training, as proposed by Peng and Kievit 2020 [41]. Education strengthens these cognitive abilities and improves the ability of academics to extract key details from complex tasks.

To address this, we plan to enrich future experiments by diversifying the participant pool to include a broader range of profiles varying in age, occupation, educational level, and gender, to better assess the tool's usability across different demographics.

8. Conclusion and future work

This paper introduces a new programming framework designed to bridge the gap between end users and collaborative robots, emphasizing accessibility for non-expert users. The integration of a BDD-inspired DSL and a visual Blockly web interface allows for intuitive task definition and execution, supporting a wide range of industrial applications. Through a user-centric approach, the system simplifies the complexities of collaborative robot programming, providing a suitable experience for users with minimal technical training.

To enhance human-robot interaction, this paper proposes a comprehensive solution that interprets multi-modal information from users – hand gestures and constructed commands – and generates responsive robot behaviours. The system adapts to human intentions and communicates decisions effectively. A feedback mechanism delivers real-time updates and a history of actions within the interface, enabling users to track executed commands and helping their understanding and interaction. This multimodal design advances Industry 5.0's goal of human-centric automation by encouraging intuitive and efficient collaboration.

The results from the user study highlight important improvements in the tool's usability. The overall user experience rating from the SUS survey increased significantly from 7.50 to 8.67 out of a maximum of 10. Additionally, integration between the robot, gesture recognition module and programming tool improved, with the mean score rising from 4.42 to 4.58 out of 5. Task completion times varied widely, ranging from 9 min (fastest) to 53 min (slowest), showcasing variability in user familiarity and comfort with the tool. Despite this variability, participants completed tasks using a consistent number of blocks (ranging from 5 to 8 blocks), indicating that the tool allows for task completion with a manageable set

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

of actions. The NASA-TLX workload assessment scores indicated moderate mental demand (38.33 out of 100) and low frustration (8.75 out of 100), reflecting the tool's accessibility and ease of use.

Despite these achievements, challenges persist, particularly in enhancing real-time adaptability and managing errors dynamically during task execution. Addressing these limitations in future work will further strengthen the framework's applicability to diverse and dynamic industrial scenarios.

Building on the findings of this study, future research will target both immediate enhancements to the current framework and broader contributions to the field of human-robot collaboration. A more comprehensive quantitative data collection is planned by including additional sensors and measurement devices. Future efforts will focus on increasing quantitative parameters and employing advanced statistical analysis to evaluate system performance more rigorously.

To improve scalability and adaptability, the framework is set to transition to run on the robot operating system, enabling seamless integration with a variety of robotic platforms and applications. This transition will allow the system to evolve into a more generalizable solution, adaptable to diverse robotic systems and environments.

When considering the tool deployed in industrial settings, it represents a proof of concept requiring initial optimization. Currently, robot poses are hardcoded from a preprocess involving manual movement to target locations and saving those movements. For on-site use, the underlying code must adapt to new environments and object locations. To facilitate this, the web interface could incorporate a "free drive" button to activate learning by demonstration, allowing users to move the robot to desired positions and save them, thus enabling non-experts to configure it independently of specialists. Physical introductions could be substituted with a tutorial video and straightforward integration steps for industrial workers. For deployment, safety measures should be considered, a digital twin could replicate the setup to test scenarios virtually before real execution, and a gesture-based or digital stop button might provide a more intuitive option than the physical red button. Hardware such as a tablet or computer and a camera, necessary for the interface, remains affordable and simple to install. To further improve communication between human operators and robots, the tool could integrate multimodal feedback such as visual (e.g., LED indicators), auditory (e.g., confirmation beeps) or haptic cues (e.g., vibration or handled device) to support real-time interaction and increase user confidence. Additionally, incorporating voice commands alongside gesture recognition could make the system more intuitive and responsive.

Future studies will also expand the participant pool, dividing them into two groups: those with programming or robotics experience and those without. This distinction will help establish a baseline for how familiarity with technology influences tool usability, isolating factors like intuitiveness and effectiveness. Furthermore, experiments will include multiple levels of task complexity to evaluate the system's effectiveness across varying cognitive demands. Key quantitative parameters, such as task completion time, error rates and user adaptation, will be identified to establish standardized metrics for evaluating human-robot interaction performance. These enhancements will be paired with longitudinal studies to assess learning curves, usability improvements over time and integration into industrial workflows, ensuring that the framework evolves to meet the demands of real-world applications. In this regard, studies such as the one conducted by Solis *et al.* [42] highlight the importance of robust evaluation methodologies that incorporate learning curves to better understand trainee skill progression and guide system improvements effectively.

Author contributions. AS and JR conceived and designed the study as well as performed statistical analyses. JS and TRS co-designed the conceptual design and the user experiments and revised the article. GG and LEH co-designed the system architecture from the robot integration perspective. JH provided intellectual contributions from the industrial perspective. AS provided intellectual contributions from an academic perspective. AS, JR and JS wrote the article.

Financial support. This research was financed in part by VINNOVA (registration number: 2021-04810) and now financed in part by Ritsumeikan International Collaborative Research Promotion Program "Sustainable Human-Robot Synergy in Extended Reality by Foundation Model Integration for Super Smart Society 5.0" as well as in part by the DIREC project "Low-Code Programming of Spatial Contexts for Logistic Tasks in Mobile Robotics" at the University of Southern Denmark.

Competing interests. The authors declare no conflicts of interest exist.

Ethical approval. Not applicable.

References

[1] OECD, Pensions at a glance 2023: OECD and G20 indicators (2023).
[2] Eurostat, Employment by sex, age and citizenship (2023).
[3] E. Commission, Industry 5.0: Human-centric, sustainable and resilient (2020).
[4] J. Solis, K. Nakamori, G. A. G. Ricardez and J. Håkansson, "Body Gesture Recognition for Collaborative Robots," *The 16th World Congress of the International Federation for the Promotion of Mechanism and Machine Science, Tokyo, Japan, November 5-10, 2023* (2023) pp. 61–62.
[5] G. A. G. Ricardez, C. Töerg, L. E. Hafi, J. Solis and T. Taniguchi, "Toward Safe and Efficient Human-Robot Teams: Mixed Reality-based Robot Motion and Safety Index Visualization," *The 16th World Congress of the International Federation for the Promotion of Mechanism and Machine Science, Tokyo, Japan, November 5-10, 2023* (2020) pp. 53–54.
[6] B. Bastin, S. Hasegawa, J. Solis, R. Ronsse, B. Macq, L. E. Hafi, G. A. G. Ricardez and T. Taniguchi, "GPTAlly: A Safety-Oriented System for Human-Robot Collaboration Based on Foundation Models," *2025 IEEE/SICE International Symposium on System Integrations (SII)* (2025) pp. 878–884.
[7] E. Martin, S. Hasegawa, J. Solis, B. Macq, R. Ronsse, G. A. G. Ricardez, L. E. Hafi and T. Taniguchi, "Integrating Multimodal Communication and Comprehension Evaluation During Human-Robot Collaboration for Increased Reliability of Foundation Model-based Task Planning," *2025 IEEE/SICE International Symposium on System Integrations (SII)* (IEEE, 2025) pp. 1053–1059.
[8] H. Lieberman, F. Paternò and V. Wulf, *End User Development*, Human-Computer Interaction Series (Springer Dordrecht, 2006).
[9] T. R. Silva, "Towards a Domain-Specific Language for Behaviour-Driven Development," *2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (2023) pp. 283–286.
[10] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver and B. Silverman, "Scratch: Programming for all," *Commun. ACM* **52**(11), 60–67 (2009).
[11] N. Fraser, "Ten Things We've Learned from Blockly," *2015 IEEE blocks and beyond workshop (blocks and beyond)* (IEEE, 2015) pp. 49–50.
[12] M. V. Merino, T. Van Der Storm and Google, "Block-based Syntax from Context-Free Grammars," *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering* (2020) pp. 283–295.
[13] J. P. De la Rosa, J. Solis, K. Nakamori, G. A. G. Ricardez, J. Håkansson, A. S. Sorensen and T. R. Silva, "From Gestures to Behaviours: An Empirical Study on Behaviour-Driven Development Scenarios to Support End-User Programming of Collaborative Robots," *IFToMM International Symposium on Robotics and Mechatronics* (Springer, 2024) pp. 369–381.
[14] X. Xu, Y. Lu, B. Vogel-Heuser and L. Wang, "Industry 4.0 and industry 5.0—inception, conception and perception," *J. Manuf. Syst.* **61**, 530–535 (2021).
[15] D. North, Introducing BDD (2006).
[16] G. Adzic, *Specification by Example: How Successful Teams Deliver the Right Software* (Simon and Schuster, New York, 2011).
[17] J. P. De La Rosa Gutierrez, T. R. Silva, Y. Dittrich and A. S. Sorensen, "Design goals for end-user development of robot-assisted physical training activities: A participatory design study," *Proc. ACM Hum. Comput. Interact.* **8**, 1–31 (2024).
[18] D. North, What's in a story? (2022).
[19] Cucumber, Gherkin (2024).
[20] J. F. Gorostiza and M. A. Salichs, "End-user programming of a social robot by dialog," *Robot. Auton. Syst.* **59**(12), 1102–1114 (2011).
[21] S. Beschi, D. Fogli and F. Tampalini, "CAPIRCI: A Multi-Modal System for Collaborative Robot Programming," *End-User Development: 7th International Symposium, IS-EUD 2019, Hatfield, UK, July 10-12, 2019, Proceedings 7* (Springer, 2019) pp. 51–66.
[22] W. Wang, R. Li, Y. Chen, Z. M. Diekel and Y. Jia, "Facilitating human–robot collaborative tasks by teaching-learning-collaboration from human demonstrations," *IEEE Trans. Autom. Sci. Eng.* **16**(2), 640–653 (2018).
[23] P. A. Akiki, P. A. Akiki, A. K. Bandara and Y. Yu, "EUD-MARS: End-user development of model-driven adaptive robotics software systems," *Sci. Comput. Program.* **200**, 102534 (2020). Publisher: Elsevier.
[24] A. Silahli, A. Kramberger and T. R. Silva, "A Domain-Specific Language Framework for Specification and Generalization of Robot Motion," *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)* (IEEE, 2024) pp. 3733–3739.
[25] C. Blanc, L. Boudry, A. Sonderegger, J. Nembrini and S. Dégallier-Rochat, "Empowering Production Workers to Program Robots: A No-Code, Skill-Based Approach," *International Conference on Computer-Human Interaction Research and Applications* (Springer, 2023) pp. 21–39.
[26] P. Tulathum, B. Usawalertkamol, G. A. G. Ricardez, J. Takamatsu, T. Ogasawara and K. Matsumoto, "Robot behavior debugger for non-expert users in convenience stores using behavior trees," *Adv. Robot.* **36**(17), 951–966 (2022).
[27] N. Leonardi, M. Manca, F. Paternò and C. Santoro, "Trigger-Action Programming for Personalising Humanoid Robot Behaviour," *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (2019) pp. 1–13.

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

[28] C. J. Sutherland and B. A. MacDonald, “Naoblocks: A Case Study of Developing a Children’s Robot Programming Environment,” *2018, 15th International Conference on Ubiquitous Robots (UR)* (IEEE, 2018) pp. 431–436.
[29] D. Mukherjee, K. Gupta, L. H. Chang and H. Najjaran, “A survey of robot learning strategies for human-robot collaboration in industrial settings,” *Robot. Comput. Integr. Manuf.* **73**, 102231 (2022).
[30] M. Kapinus, V. Beran, Z. Materna and D. Bambušek, “Spatially Situated End-User Robot Programming in Augmented Reality,” *2019, 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)* (IEEE, 2019) pp. 1–8.
[31] E. Senft, M. Hagenow, R. Radwin, M. Zinn, M. Gleicher and B. Mutlu, “Situated Live Programming for Human-Robot Collaboration,” *The 34th Annual ACM Symposium on User Interface Software and Technology* (2021) pp. 613–625.
[32] S. Y. Gadre, E. Rosen, G. Chien, E. Phillips, S. Tellex and G. Konidaris, “End-User Robot Programming Using Mixed Reality,” *2019 International Conference on Robotics and Automation (ICRA)* (IEEE, 2019) pp. 2707–2713.
[33] M. Shaw, Larger Scale systems require higher-level abstractions, *ACM Sigsoft Software Engineering notes* **14**(3), 143–146 (1989).
[34] G. A. I. Edge, Mediapipe gesture recognizer (2023).
[35] J. Engel, B. Rice and R. Jones, Behave: BDD, python style (2024).
[36] S. G. Hart and L. E. Staveland, “Development of NASA-TLX (task load index): Results of empirical and theoretical research,” *Adv. Psychol.* **52**, 139–183 (1988).
[37] P. Kortum, C. Z. Acemyan and F. L. Oswald, “Is it time to go positive? assessing the positively worded system usability scale (SUS),” *Hum. Factors*. **63**(6), 987–998 (2021).
[38] U. Kohler and F. Kreuter, *Data analysis using Stata* (Stata press, College Station, TX, 2005).
[39] S. G. Hart, “Nasa-task load index (NASA-TLX); 20 years later,” *Human Factors and Ergonomics Society Annual Meeting*, **50**(9) (Sage Publications, Los Angeles, CA, 2006), pp. 904–908.
[40] K. A. Ericsson, R. T. Krampe and C. Tesch-Römer, "The role of deliberate practice in the acquisition of expert performance," *Psychol. Rev*. **100**(3), 363 (1993).
[41] P. Peng and R. A. Kievit, The development of academic achievement and cognitive abilities: A bidirectional perspective,” *Child Dev. Perspect.* **14**(1), 15–20 (2020).
[42] J. Solis, N. Oshima, H. Ishii, N. Matsuoka, K. Hatake and A. Takanishi, “Towards understanding the suture/ligature skills during the training process using WKS-2rii,” *Int. J. Comput. Assist. Radiol. Surg.* **3**, 231–239 (2008).

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

Appendix A. Demographic Survey Administered to the Participants Before the Experiment

The Figures below represent the first survey shared with the participants to require personal background information and gather other crucial key factors.

Figure A1. *Demographic survey Page 1.*

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

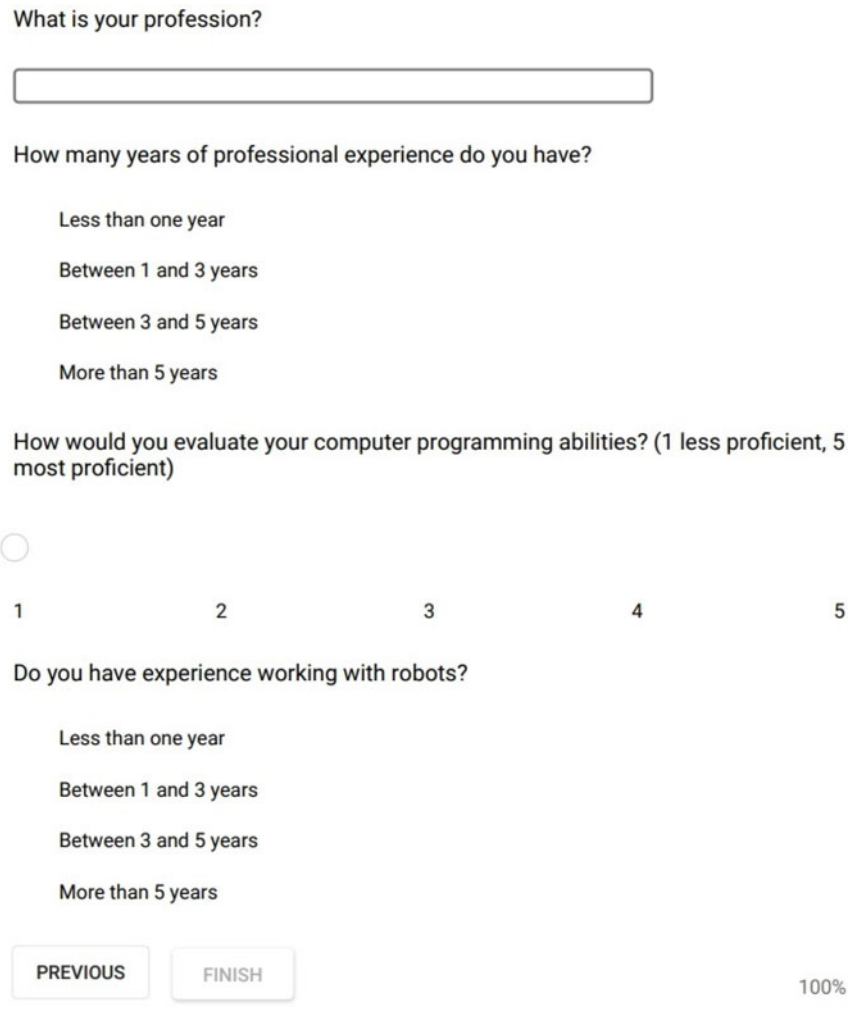


Figure A2. *Demographic survey Page 2.*

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

Appendix B. Proctor Notes used During the Experiment

The document below presents the relevant quantitative information gathered by the proctors during the experiment.

Data Annotation - Participant P#

During the task

Start time:
Time used task #1:
Time used task #2:
Time used task #3:
End time:

The number of times a participant executed their program solution:

Once task is finished

Number of scenarios created to solve:

Figure B1. *Proctor notes.*

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

Appendix C. SUS Survey Administered to the Participants After the Experiment

An SUS survey was used to provide a reliable measure of the usability of the proposed system from the users' perspective.

Figure C1. *SUS Page 1 (Questions 1–5).*

https://doi.org/10.1017/S0263574725101720 Published online by Cambridge University Press

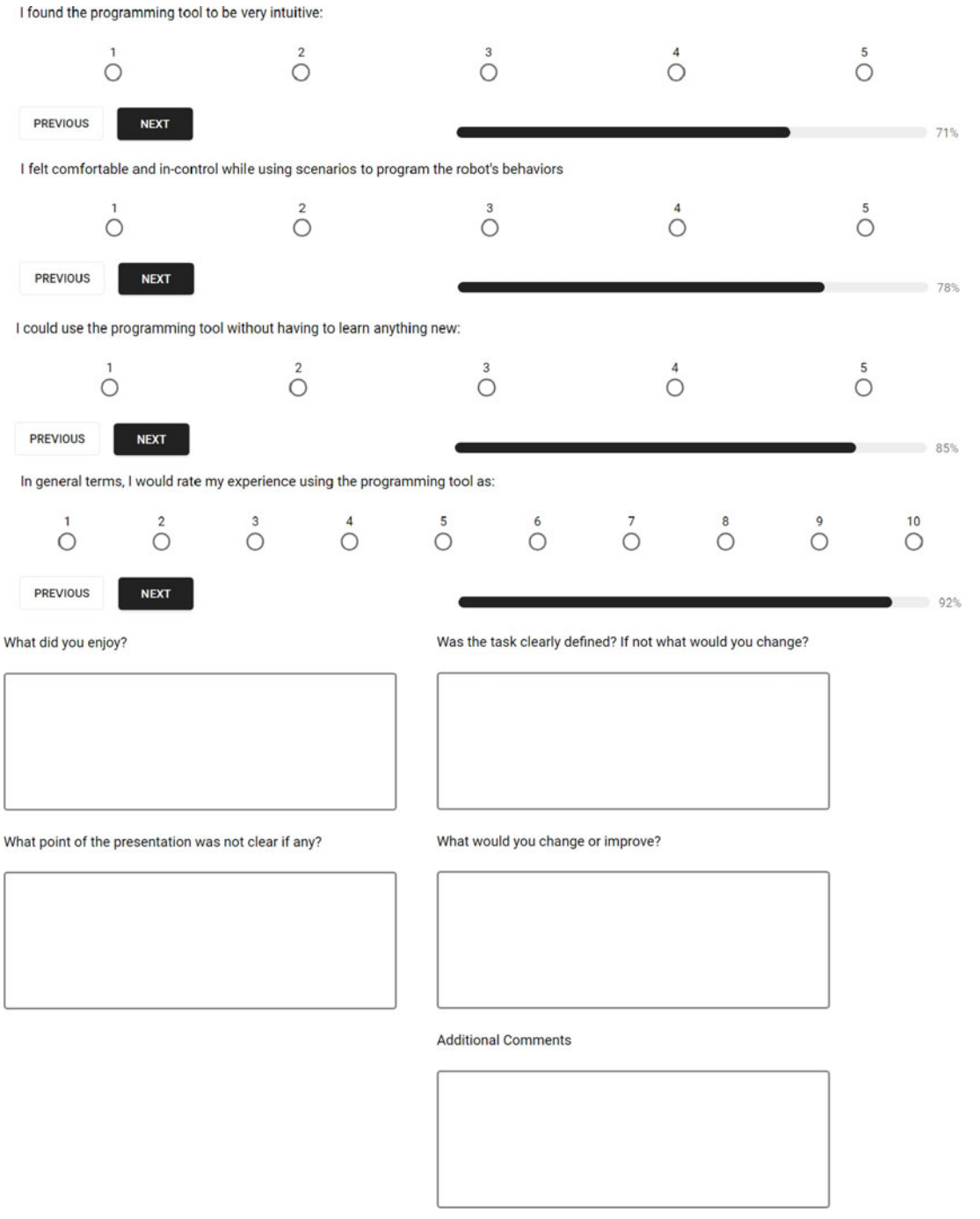


Figure C2. *SUS Page 2 (Questions 6–10).*

Cite this article: A. Silahli, J. P. De la Rosa, J. Solis, G. A. Garcia Ricardez, L. El Hafi, J. Håkansson, A. Stengaard Sørensen and T. Rocha Silva, "A gesture-based behaviour-driven development approach for end-user cobot programming", Robotica. https://doi.org/10.1017/S0263574725101720