







Empowering Production Workers to Program Robots: A No-Code, Skill-Based Approach

Charly Blanc^{1,2}(✉) , Lionel Boudry³, Andreas Sonderegger³ , Julien Nembrini² ,
and Sarah Dégallier-Rochat¹ 

¹ HuCE Institute, Bern University of Applied Sciences, Quellgasse 21, Biemme 2502, Switzerland

² Human-IST Institute, University of Fribourg, Boulevard de Pérolles 90, Fribourg 1700, Switzerland

charly.blanc@bfh.ch

³ Institute for New Work, Bern University of Applied Sciences, Bern, Switzerland

Abstract. The current market requires automated production systems to be reprogrammed by the shop floor workers to meet dynamic production needs. This requires new interfaces allowing the workers to acquire the needed skills for efficient and safe programming. In this article, an intuitive interface is introduced to foster both upskilling and empowerment through guided tutorials. A no-code approach to programming based on the notion of robotic skills enables interactions that are based on the worker's competencies. A preliminary study with students (N = 58) using between-group testing was performed to evaluate the usability of the interface and skill acquisition through the tutorials. The effect of a basic understanding of robots' behavior on users' performance was evaluated: a demonstration with a real robot was presented to half of the participants before the study. Our results indicate that the proposed approach enabled most novice users to achieve simple programming tasks. The demonstration with the robot had a positive impact on performance suggesting the need for real robot interaction to improve learning. In summary, the combination of a no-code, skill-based approach with problem-based tutorials and demonstrations with real robots can help non-expert users develop the competencies and confidence to autonomously program a robot. Further tests with intended target users are planned in the future.

Keywords: Robotics · Human computer interaction · Programming learning · Upskilling · Empowerment

1 Introduction

Traditionally, most assembly processes are either fully automated or manual. Automation solutions have been optimized for decades to become ever more precise, rapid, and robust. Such solutions are ideal for repetitive tasks in a structured environment, but not feasible for tasks requiring fine manipulation, problem-solving capabilities and adaptability, where manual work is still required [10]. Recently, a new paradigm for automation as emerged with collaborative robots (cobots). These robots fulfill safety

requirements that allow them to share the workspace with humans, contrarily to traditional industrial robots that work in a closed, separated environment [4]. This possibility of sharing a common workspace enables a new form of automation, called partial automation, where the human and the robot can work simultaneously together on the same task, leading to new, narrower forms of human-robot collaborations [38].

If full automation has been the main focus in the industry until recently, the ever growing versatility of the market, the constant acceleration of technology development, the growing demand for customized products and the tendency for shorter product life duration has lead to a growing interest in partial automation [30,35]. Indeed, while machines are faster, more precise and have a greater repeatability, humans are much more flexible and can rapidly adapt to new tasks [37]. The aim of partial automation is thus to combine the flexibility of human work with the efficiency of automation [41]. In particular, it should be possible to use the same robot for different tasks and to easily and rapidly reprogram the robot according to the market demands, bringing the needed agility for the company to cope with market versatility [14].

However, companies that could benefit from partial automation are typically small to medium companies that cannot afford automation experts. Outsourcing the re-programming takes time and is expensive and thus strongly reduces the advantages of partial automation. In order to fully leverage its potential, it is thus required for the workers on the shop floor to be able to reprogram the process by themselves [32]. The goal is indeed that an expert in the process, instead of an expert in the technology, shall be the one to program the robot. A new approach to worker-machine interaction is therefore required for partial automation to work [7], similar to what window-based graphical user interfaces did for personal computers 40 years ago: rather than expert systems, intuitive solutions that can leverage the worker's task expertise are needed. This also represents an opportunity to make production work more attractive: the robot takes over the dull, tedious or dangerous tasks, while the worker focuses on the advanced tasks requiring human capabilities [29]. For the worker, it could also mean a greater autonomy and opportunities for self-development [15].

Giving the possibility to the worker to reprogram the machines represents a big paradigm shift, sometimes refer to as Operator 4.0 [30], in the context of Industry 4.0, or augmented worker [25]: traditionally, automation solutions are expert systems designed to reduce as much as possible the intervention of non experts to minimize the risk of errors [23]. For partial automation to work, the power relationship between the worker and the machine needs to be changed, with the worker taking control over the system. This requires not only the worker to acquire the needed competencies to interact safely and efficiently with the machine, *i.e. upskilling*, but also to develop a feeling of ownership over the system, *i.e. empowerment* [6]. To better understand how to foster the transition from passive user to active programmer, a socio-technological perspective is needed to guide the design of the human-machine interaction.

In this article, we propose a first step in this direction. An interface for robotic programming was implemented based on the tried and tested block-based approach to learning programming [26,39]. This approach doesn't require the user to learn any programming syntax (compared to text based approach) making it easier to use. A set of step-by-step tutorials were developed to help the user to progressively acquire

the needed competencies and gain a feeling of control over the system. Combining a block-based approach with robotic specific language was already tested [39] and shows that user could successfully implement robot programs with high score of usability, learnability and overall satisfaction. But since our goal is to empower workers to autonomously control the robot, the feeling of control is an important aspect that, to our knowledge, have not been yet investigated. We believe that a feeling of control on the machine can be achieved by providing a good understanding of how does this one work, combined with an effective and satisfactory interaction with the machine enabling the user to reach their goal efficiently. Improving the feeling of control over the machine will be therefore the main focus of this article. The research question tackled is the then the following: How does providing workers with a deep understanding of how a robot works, along with an effective and satisfactory interaction with the machine, impact their perceived feeling of control over the robot and their ability to autonomously control it for efficient task completion? A study focusing on upskilling was conducted with students to evaluate the relevance of our approach. A 2-group experiment was performed to assess the effect of basic knowledge about robotics on programming performance. The usability of the interface was also evaluated.

We start by presenting the relevant literature in the next section. Section 3 presents the interface and the tutorials that were developed for this study. The study is presented and discussed in Sect. 4. A discussion is proposed in Sect. 6, including future work. Finally, a conclusion is presented in Sect. 8.

2 Related Work

In the past decade, there has been a large movement to democratize the usage of robotics, sometimes referred as the no-code revolution [28]. Cobots have opened the path to this democratization: since they allow for physical interactions with the device, ad hoc solutions can be developed by non-expert users by directly interacting with the robot, without having to code. Most cobots are now endowed with intuitive programming interfaces allowing to easily concatenate pre-defined, parametrizable behaviors. However, these systems are typically design to be as intuitive to use as possible, which results in a lack of flexibility that limits the usage of the cobots to simple, repetitive tasks in controlled environments [38], which can explain the relative low adoption of plug-and-play cobotic systems in the industry. To fully leverage the potential of partial automation systems for non-expert users, an interface is required that is not only easy to use, but also easy to learn.

In our framework, we propose a block-based interface, similar to the well-known Blockly [40] or Scratch [26], build on top of a skill-based architecture [32]. Robot skills are associated to predefined behaviours combining sensory and motor information. Previous work successfully combined Blockly with industrial robots [40], enabling the programming of a pick-and-place task of a virtual robot by adult novices. Other studies investigated robot programming teaching for adult novices [39] using Blockly with a framework called CoBloX, compared to two widely used industrial programming approaches. The results demonstrate that participants successfully completed the robot programming tasks while receiving higher scores for usability, learnability, and

general satisfaction. The participants were specifically instructed to focus on a limited range of functionality, such as implementing a *pick and place* task. However, important programming concepts like loops and conditions were not introduced, despite their necessity when working on more extensive programming tasks in real case scenario.

In fact, the concept of loops in programming is central but was shown to be difficult to understand by novices [12]. Another study [13] showed that students usually lack familiarity with variables and have misunderstandings regarding their usage. Other difficulties come from the visual representation of the program itself, and were explored by the Blockly team [9] who describe a non-exhaustive list of problems faced by novice programmers while learning programming with Blockly. To tackle these problems, a collection of user-centered tutorials were developed (Blockly games) to teach how to use the interface as well as the difficult programming functionalities. The combination of a block-based interface with tutorials has shown good results in terms of breaking learning barriers [9]. We will therefore use a similar approach for our framework and adapt it our target group, the production workers.

Another critical step while programming is the creation of functions (or skills in the context of robotics). In fact, functions provide a better organization of a code and enable to reuse of this last one in an efficient way. In the context of block-based programming, Blockly [9] and Scratch [20] gives the possibility to create functions, by passing arguments to the body of the function, similarly to text based programming. This approach is based on a top-down approach (the user needs to know first what the function needs to do), requiring a fairly good level of expertise of the user. In Sect. 3.1, we will present a new, bottom-up approach, enabling the user to more easily create functions/skills from the body of a function.

More generally, our focus lies on providing interfaces that are not only *easy to use*, but also *easy to learn*. It has been observed that if the introduction of cobots in manufacturing environments can improve the quality of activities and enhance the significance of workers' role, reskilling is needed [16]. In this context, our work aims to bridge the gap between intuitiveness and empowerment by exploring user-centered methods to enable greater customization and autonomy in robot programming, ultimately fostering a more adaptable and efficient human-robot collaboration.

3 An Intuitive Robotic Interface

We believe that the desired empowerment of the production workers can be achieved by designing an interface that enables them to develop a better understanding of the system and, consequently, a feeling of control over the machine. In order to achieve this, an interface is needed that is easy to take in hand, but that also fosters the learning to acquire the needed competencies to fully leverage the flexibility of the system. Indeed, an interface emphasizing intuitiveness may typically trade-off flexibility to increase the worker's autonomy, while an interface offering too many possibilities will be difficult to handle for beginners. To overcome this trade-off, we propose an adaptive interface that takes in account the users' skills and needs, but also tutorials to allow them to gain the skills needed to use the more advanced functionalities.

Our adaptive interface is based on a block-based approach, where blocks can be combined to create complex behaviors. Different levels of blocks are defined that correspond to different types of users (normal, intermediary, advanced), which allow us to easily adapt the interface to the user needs. The high-level blocks are based on natural language and provide a very intuitive interface, while the lower level blocks reflect the robot commands and thus provide a flexibility similar to typed languages, but without the burden of the syntax. An intermediate layer, based on the concept of skills, allow for the smooth transition between the high and the low-level layers.

In addition, a collection of tutorial was developed to foster learning-by-doing and upskilling of the users and to teach them progressively the different functionalities of the interface, as well as some programming concepts.

In the following subsections, the concept and the implementation of the interface and the tutorials are explained in more details.

3.1 A New Block-Based Programming Interface: PrograBlock

One of the main problems faced by computer programming learners is the syntax of the programming language. To simplify the programming approach and remove the complexity of syntax, tools like Blockly [40] and Scratch [26] have been developed, both of which use block-based programming interfaces. Deeply inspired by this programming paradigm, we have developed a new block-based interface named *PrograBlock* tailored to the specific needs of robot programming in industrial settings. The user can choose a block from a library, drag and drop it in the programming area, and by connecting the different blocks together, can create a variety of programs. The different parameters can be changed directly within the block itself. Loops and if-else conditions are implemented the same way Blockly or Scratch does by wrapping blocks to be repeated, providing control over the program flow (see Fig. 1).

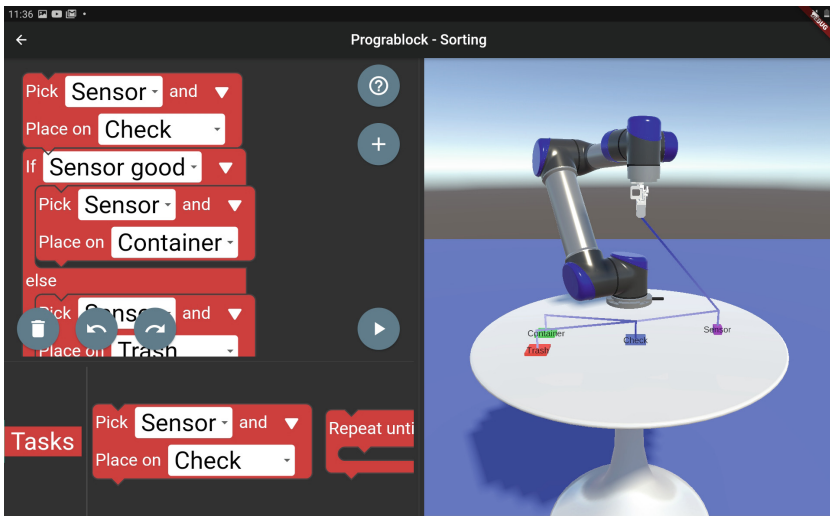


Fig. 1. PrograBlock interface including a programming area (top left) and a library of blocks (bottom left), with the 3D simulation (right).

The biggest difference with other block-based programming interface (Blockly, Scratch, etc.) is the simplification of the creation of new blocks: when a new functionality is created through a program (Fig. 2a), the user has the possibility to create a block out of it. Parameters specifying the program functionality can then be chosen by the user (Fig. 2b) to appear as block parameters, in such a way that less important parameters are hidden within the newly created block (Fig. 2c). The creation of new blocks is also possible in the framework Blockly, but the user has to use variables to achieve the passing of arguments, which was shown to be difficult for novices [13].

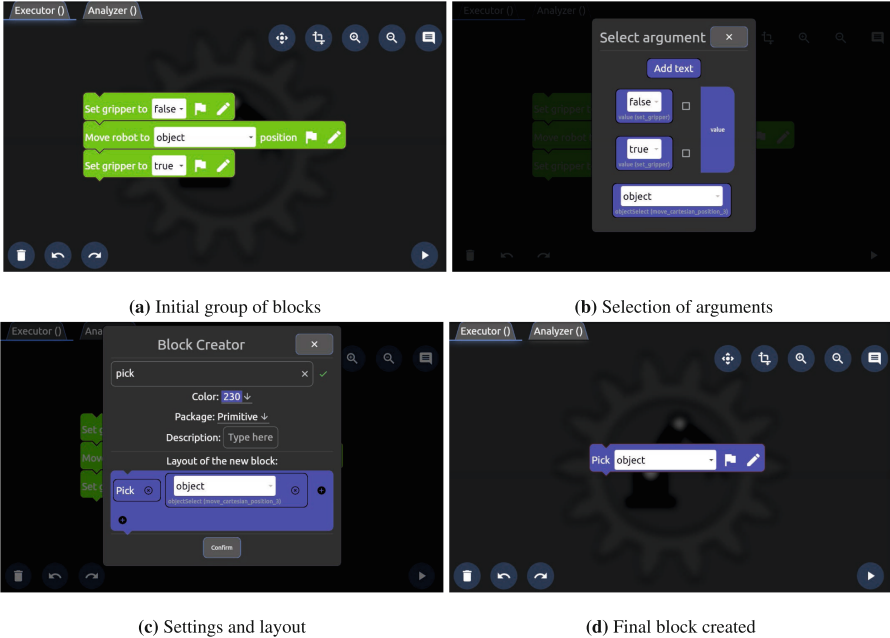


Fig. 2. Simplified block creator.

Thanks to this approach, parameters such as the objects to interact with can be parameterized, enabling the implementation of more abstract blocks that can be easily used by non-expert users.

3.2 A Skill-Based Architecture

In order to empower operators in their interaction with robots, a three-layer, block-based architecture that enable users to progressively gain control over the system was developed. There are three levels of blocks: tasks, skills and device primitives [24], examples of each level of blocks are illustrated on Fig. 3. Tasks correspond to the natural language that an operator would use with a colleague, while device primitives correspond to the capabilities of the devices and require advanced knowledge about robotics. Skills are object-oriented and serve as a communication bridge: they offer an intermediate language that can be used to describe the process steps in relation to the object being

handled [32]. For example, the skill *Pick object* describes a strategy allowing the grasping of a specific object, being the parameter of the skill. New tasks can then be created by simply parameterizing and combining skills. All blocks can be parameterized and thus easily adapted to new situations.

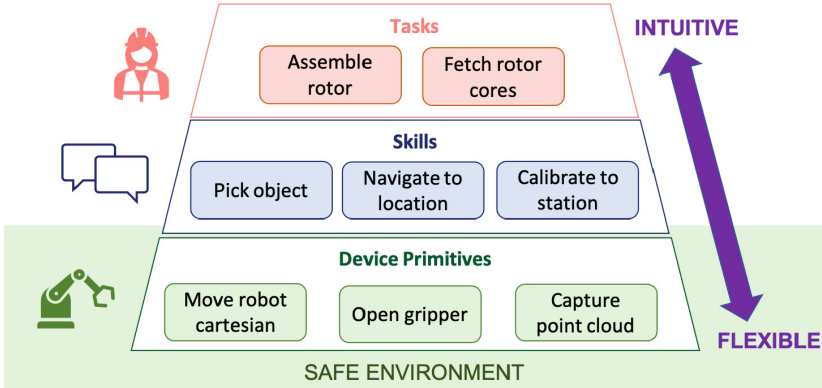


Fig. 3. Abstraction Layers.

By offering different levels of abstraction, such structure efficiently overcome the traditional flexibility-usability trade-off [32]. On one hand, it allows novices to easily adopt the high level functionalities, such as tasks and skills, to quickly create functioning programs, therefore lowering programming barriers [1]. On the other hand, experts keep the full control on the hardware capabilities thanks to the device primitives, enabling them to develop complex functionalities.

Another advantage of this architecture is the possibility to easily navigate between the different abstraction layers. It allows the user to progressively learn more complex functionalities according to their needs. In order to support this learning process, tutorials were developed that allow for the users to progressively learn new functionalities.

3.3 Virtual Environment

Complementary to the programming interface, a virtual environment (based on the game engine Unity) was developed to provide visual feedback to the user. The final application was deployed on a tablet, where the user could visualize the workspace with the robot, the camera and different objects available in the workspace. We took advantage of the virtual environment to show additional information about the state of the system, to improve the shared ground between the user and the machine and therefore the interaction. The user was able to rotate the view around the robot to show potentially hidden objects or to better visualize estimated trajectories (see Fig. 1, right).

A preview of the robot trajectory, represented by a blue line in the Fig. 1, was implemented to help the user to debug its program, and represent the trajectory of the tool center point of the robot. The difference of color along the trajectory indicate the different position over time: dark blue indicates the beginning of the trajectory and slowly

turn to light blue until reaching the end of the program. A collision detection functionality was as well developed to warn the user and allow her/him to correct the program. It was indicated by a red cross at the location of the potential collision between the robot and an object. By using a virtual environment, we intended to reduce the fear of the user of breaking or being hurt by the robot and therefore to improve the learning conditions. This hypothesis will be further tested in a future experiment.

3.4 Tutorials

Our skill-based architecture and block-based visual programming interface simplifies the programming of the system, but inexperienced programmers still have to understand the fundamentals of programming. Learning programming is a complex process [27], so a collection of tutorials was developed to progressively introduce some programming concepts to divide the intrinsic cognitive load [34] of the learner. It includes the different functionalities of the interface itself, in parallel with complex programming concepts, such as loops and conditions. The tutorials developed are a collection of six scenarios (see Fig. 4) based on problem-solving, inspired from standard industrial use cases, like *Pick and Place* or *Palletizing*.

The tutorials consist of four guided scenarios (see Fig. 4a, 4b, 4c and 4d) that teach participants about the different functionalities of the interface, as well as some programming concepts, such as loops and conditions, that are important for the automation of repetitive tasks. The four guided scenarios can be described as follows:

- **Pick and Place:** Teach how to create a simple program and how to parameterize it. In this scenario, the goal is to pick the cube, and place it on the green square.
- **Palletizing:** Teach the concept of loops. Here the goal is to pick and place cubes on the pallet until this last one is full.
- **Sorting:** Teach the concept of conditions. The goal here is to pick a cube, check the color of this last one and then place it on one of the corresponding square.
- **Pick Navigate and Place:** Teach the concept of task modification. If the user program this task the same way he did in the first tutorial 4a, a collision will occur with the obstacle (indicated with a red cross). To avoid the collision, the user has to modify the *Pick and Place* block and add a way-point between the *Pick* and the *Place* blocks. By doing so, the robot can successfully bring the cube on the green square.

In order for the user to test his/her progress and see how much was learned, two more scenario are implemented where the user is not guided anymore and has to solve the task by her/himself. The two unguided scenarios are the following:

- **Palletizing and Sorting:** Very similar to the tutorials and mainly tests the acquisition of the programming concept. Here the goal is to pick all the cubes, check their colors and place them on the corresponding pallet.
- **Palletizing and Sorting with Obstacle:** Same as Palletizing and sorting with an additional complexity due to the addition of an obstacle between the mirror and the cubes, forcing to re-program trajectories to avoid collisions.

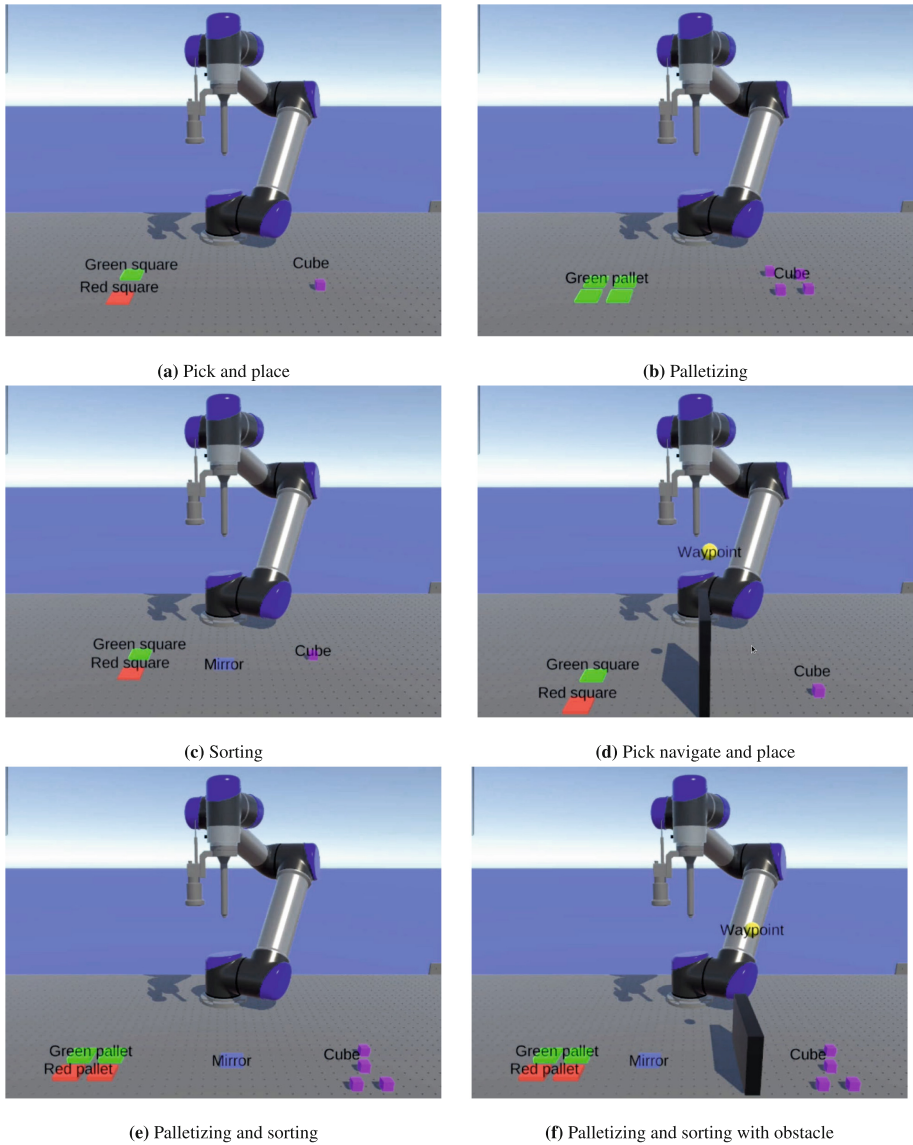


Fig. 4. Tutorials.

The goal is the same than the previous exercise 4e: pick all the cubes, check their colors and place them on the corresponding pallet.

In order to test the relevance of our approach, the interface was tested with students in a between-group study. The details of the experiment are explained in the next section.

4 Evaluation of the Interface

A study was conducted to evaluate the usability of the interface. The goal was to evaluate if the four first guided tutorials were sufficient to enable the users to perform the last two test scenarios without external help. Since most of our users had never seen a real industrial robot before, we also tested if a small demonstration of a robot performing an industrial task would have an influence on the performance and the affective state (valence and arousal) of the users. During the demonstration, the challenges associated with robotics were explained, such as vision or picking.

4.1 Participants

A total of 58 business school master students (21 identified themselves as female and 37 as male) participated in this study. Participants were aged between 23 and 46 ($M = 30.4$, $SD = 5.1$) years, and had no previous experience in programming.

4.2 Procedure

Participants were randomly attributed in two experimental groups. In one group (introductory session), participants were introduced to a real robotic cell in function while in the second group (control group) no demonstration of the robot was offered and participants started the programming tutorial directly (see Fig. 5).

The introductory session consisted of the presentation of a working robot cell that was composed of a collaborative robot and a gripper, allowing an interaction with its environment, and a camera, to detect objects' locations. Participants could observe the execution of a sorting task by the robot arm in real time. The idea was to indicate to the participants that the robot is not endowed with intelligence, but instead simply follows the instructions previously programmed by a human.

Participants were then teamed in groups of two to enhance knowledge sharing. Each pair of participants completed a tutorial and a subsequent test session interacting (see Fig. 6).

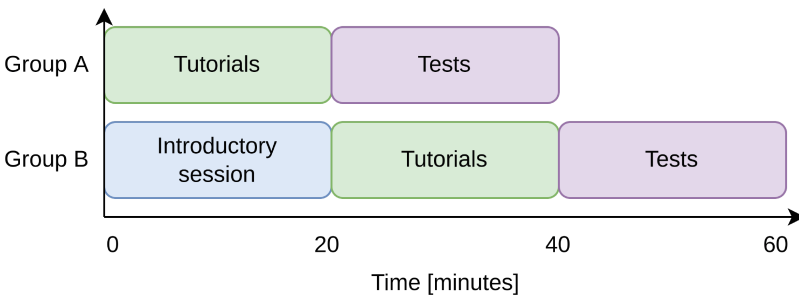


Fig. 5. Groups procedure.

4.3 Measures

The following indicators of performances were automatically logged by the system for each phase of the task: the **time** needed to successfully finish a task, the **number of modifications** (i.e., program changes) such as connecting or disconnecting blocks (increased number of modifications indicates reduced performance), the **number of trials** (i.e., executions of the program) before successfully finishing an exercise (large number of trials can be interpreted as low performance, but also as an exploratory behavior). In addition, the following subjective measures were assessed (for each participant individually):

System Usability Scale (SUS) [5] measures the usability of the interface allowing us to compare it with existing systems and future version of the interface. Over the years, the System Usability Scale has become an increasingly popular tool for measuring the usability of a wide range of products and systems. The SUS score is quick and easy to administer, making it a popular choice for evaluating usability in a variety of contexts. In the field of Human-Robot Interaction (HRI), some study [42] highlight the need and relevance of the usability assessment of robotic system interfaces.

Affective States Using Animated Visualization (AniSAM) [33] for the evaluation of valence and arousal of the user through the use of animated visualization. It is based on the circumplex model of affect described in [31] over time to give us information of the evolution of the user's emotions and detect potential design issues encountered during the use of the interface. The affective state was evaluated four times along the different tutorials (see Fig. 6).

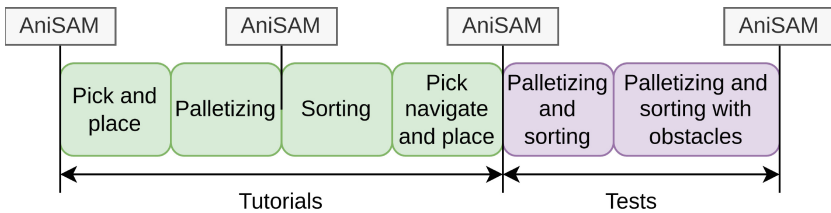


Fig. 6. Affective states evaluation procedure.

4.4 Data Analysis

To test the hypotheses regarding the behavioural data, two-sided t-test were calculated. For the analysis of affective states (AniSAM), repeated measures ANOVA (GLM) were calculated, with the first measure (baseline measure) entered as covariate.

5 Results

Analysis of **behavioural data** revealed that no differences were observed between the two groups in the guided tutorial phase regarding time spent on the tutorial, total number of trials and total number of modifications (see Table 1 for descriptive and inferential

statistics). During the test session, no differences were observed between the two experimental groups for the first, simpler task. However, for the second, more complex task, significant differences were observed (see Test 2 in Table 1, completion time with $p = 0.044$ and modifications with $p = 0.046$), with participants who received an introductory session completing the task faster and making a lower number of modifications compared to the group not receiving the introductory session.

Table 1. Descriptive and inferential statistics.

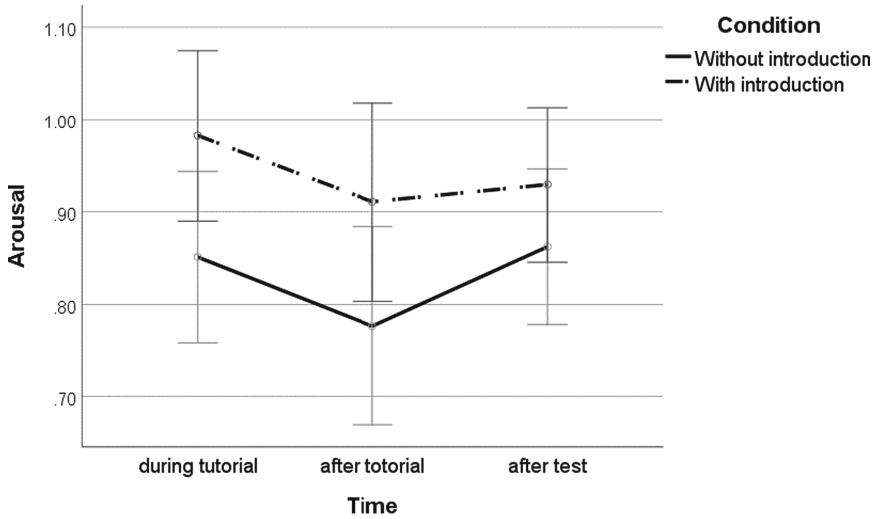
	Without introduction M (SD)	With introduction M (SD)	Test statistics t(N), p, 95% CI
Tutorials			
Completion time	563 (166)	639 (258)	t(22)=0.86, p=0.40 [-260, 107]
Number of trials	6.1 (3.2)	8.1 (4.23)	t(22)=1.31, p=0.21 [-5.18, 1.18]
Modifications	55.67 (23.46)	60.67 (36.69)	t(22)=0.40, p=0.70 [-31.07, 21.07]
Test 1			
Completion time	359 (238)	386 (307)	t(23)=0.25, p=0.80 [-254, 199]
Number of trials	4 (3.51)	4.5 (4.38)	t(23)=-0.32, p=0.76 [-3.77, 2.77]
Modifications	43.85 (30.3)	46.83 (40.39)	t(23)=-0.21, p=0.84 [-32.73, 26.40]
Test 2			
Completion time	378 (129)	238 (37)	t(9)=2.34, p=0.044 [4.46, 276.18]
Number of trials	4.17 (3.76)	2.40 (1.34)	t(9)=0.99, p=0.35 [-2.27, 5.80]
Modifications	51 (28.23)	20.6 (4.29)	t(5.27)=2.60, p=0.046 [0.48, 59.97]

Regarding skill acquisition, most users (90%) were able to program a simple robotic tasks by themselves despite the short duration of the tutorials (20 min). Less than half of the users (38%) were however able to program the most complex task, potentially because of the time limitation.

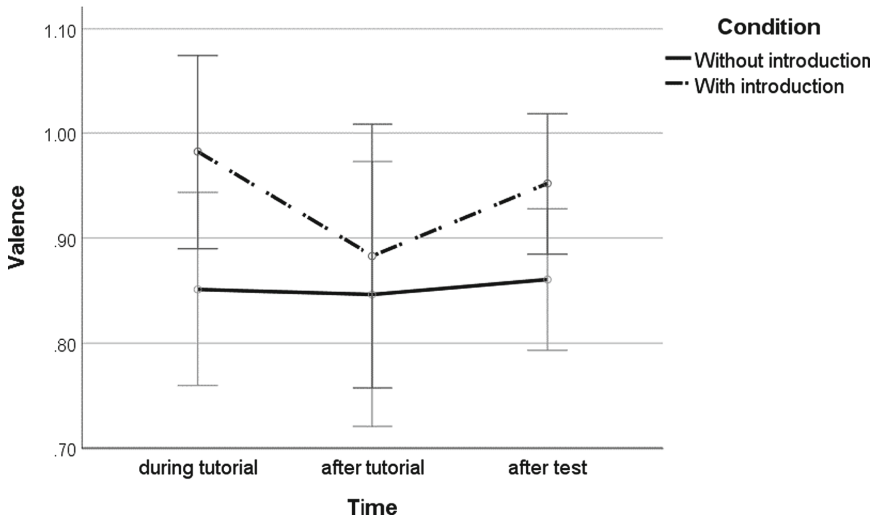
The **usability** assessment of the system resulted in a SUS-score [5] of 71.5 (SD=12.3) which can be considered to be between good and excellent in terms of adjective ratings [2]. Statistical analysis comparing the two experimental groups indicates very small mean-differences with no effect of significance ($t(40) = 0.31$, $p = 0.76$, $d = 0.1$).

Analysis of participants' affective state over the different phases of the experimental study revealed higher levels of **arousal** for participants in the introductory session group compared to participants not receiving an introduction ($F(1, 20) = 4.76$, $p = 0.041$, $\eta^2 = 0.19$; see Fig. 7a). The evolution of the valence over time ($F(2, 40) = 2.82$, $p = 0.072$, $\eta^2 = 0.12$) and the interaction (introductory session x time) did not reach significance level ($F < 1$).

With regard to **valence**, data analysis revealed that the effect of the experimental manipulation was not significant ($F(1, 20) = 2.42$, $p = 0.14$, $\eta^2 = 0.11$; see Fig. 7b) while the effect of time was significant ($F(2, 40) = 5.74$, $p = 0.006$, $\eta^2 = 0.22$). Sidak-corrected pairwise comparisons (statistical procedure for comparing multiple groups, considering the issue of multiple comparisons) however indicated no significant differences between the three measurement points. Furthermore, the interaction (introductory session x time) did not reach significance level ($F(2, 40)=1.05$, $p=0.36$, $\eta^2=0.05$).



(a) Arousal



(b) Valence

Fig. 7. Evolution of affective states over time and experimental conditions.

6 Discussion

The findings indicate that the majority of participants (90%) effectively programmed the simple task (Fig. 4e), demonstrating the interface's user-friendliness and the successful acquisition of necessary programming skills through the tutorials by the users.

An important findings from this study relate to the effect of the experimental manipulation on the performance data in the test tasks (Fig. 4e and 4f). While no effect of the introductory session on the (easier) first task was observed, participants performed better in the (difficult) second task if they participated in the introductory session. In the first test task, the goal of the exercise was to palletize several cubes on different pallets. In the second test task, an obstacle had to be avoided by the robot arm while palletizing cubes. Most participants successfully programmed the robot to avoid the added obstacle between pick-up and placement areas using a waypoint. However, a significant amount of them had difficulties realizing that this avoidance procedure was also needed on the way back of the robot arm to the pick-up area because of the repetition of the pick-up and placement behavior inherent to the palletizing task. The percentage of participant who had the time to finish this second test task being relatively low (38%), exploring the performances without time restriction could bring even more information about this finding.

The observation that the group who followed the introductory session displayed better performance in this test may be a consequence of a better understanding of the way how robots work. Being machines, robots follow ‘blindly’ a list of actions defined (programmed) by a human and are not able to ‘understand’ contextual information. If one considers the robot endowed with information contextualization capacity, one would expect the robot to avoid the obstacle on both ways; while a clearer understanding that the robot blindly follows a list of instructions leads to correctly solving the task. This indicates that following the introductory session influences the mental model [17] of participants, which allows to predict or explain a system’s behavior. In other words, the results suggest that the user’s mental model of the robot plays an important role in the learning performance of robot programming.

Participants receiving the introductory session showed a higher level of arousal compared to the control group. Although the origin of this effect can only be speculated, it is possible that experiencing a real robot arm in action is related to this increased arousal, which might be linked with an increased level of experienced stress. It could be that possible negative consequences of this stressful experience were compensated by the positive effects of better understanding of the way how robots work. Further research is needed to replicate this result and determine its origin.

In a similar vein, it is difficult to interpret the change in the valence measurement because of the non-significant post-hoc comparisons. Further studies with more power are needed here to allow for clearer statements regarding the affective consequences of such introductory sessions.

Limitations of this study include the rather small sample size and the programming being conducted in groups of two. Although group evaluations are a sometimes used method in the field of interface development [3, 8], group effects may influence the results in this study. A replication of these results in a single study with more power is therefore planned.

A current limitation of the study is its focus on assessing the system interface’s immediate intuitiveness and the effect of user training on efficient utilization. Indeed, this study involves novices recruited from a master’s program, while the target demographic for an end-user product based on this system is likely to be made up of blue-

collar workers who are heavily invested in using the system efficiently while at work for an extended period of time. These employees will likely take the time to become used to the technology and are mostly focused on workflow issues unique to their production activities. They place more focus on the system's medium to long term integration into the production cycle than on how intuitive or simple it is to use right now. However, the assessment of the usability of the system is still relevant to promote the acceptance of the technology, since a user-friendly and well-designed system reduces the learning curve, minimizes user errors, and ultimately enhances user confidence, which are all key factors that contribute to the successful adoption and utilization of the technology. A future experiment including our target users in industrial settings is already planned and will be the subject of a future publication.

In summary, this study showed interesting results indicating a positive impact of our newly developed interface on robotic learning. An initiation with real robots seems to enhance learning outcomes by fostering a deeper comprehension of the situation, including mental models and situation awareness. However, it is important to note that this immersive approach could also lead to increase arousal levels, potentially indicating an increase of stress. As predicted when developing the virtual environment of the robotic system (see Sect. 3.3), being exposed to the real robot can be associated with an increase in arousal and stress. It seems that the positive effect of increased situational understanding was able to compensate for the possible negative consequences of experiencing additional stress due to exposure to real robots. The question arises whether it is possible to facilitate the positive effects of the presentation of the robot without additional stress experience - a question that should be addressed in future studies.

Altogether, this study show encouraging results on our journey to make robotics programming more accessible for novices, although test with our target users are still needed.

7 Future Work

Using a virtual environment to learn robot programming has been favoured for better learning performances, especially allowing to bypass the fear of breaking the system or being hurt by the robotic arm [18], more especially with the close interaction allowed with cobots. The results presented here suggest that working with the real functional robot may instead bring more tangibility during the interaction and making programming easier to learn. Future studies are planned to verify this hypothesis.

It is important to highlight that the current focus of the tutorials is on comprehending programming concepts. However, based on our findings, it is evident that additional tutorials will be necessary to help users develop a more comprehensive understanding of how the robot functions. Developing effective programming interfaces requires a deep understanding of the factors that influence learning and performance, particularly in relation to novice programmers who often face barriers known as the Gulf of Execution and Evaluation (GEE) [22]. In programming the GEE can be seen as the challenge of converting intentions into code and interpreting input from the programming environment [1]. But utilizing robots as a learning tool appears to be an effective approach for beginners to grasp fundamental programming concepts [19]. Overcoming the

GEE is a common struggle for inexperienced programmers. Therefore, the construction of mental models becomes essential in surmounting these barriers, as learners' internal representations of programming concepts and environments can significantly assist them in bridging the GEE more effectively and honing their programming skills [17].

At the time of the study, the interface was still a prototype and we decided to test it first with students to gain insights on its potential and limitations before testing it with our target users. We are now working with a company to integrate our system to their shop floor and we will start testing it with the employees in the coming months. The goal of the project is to incrementally improve the interface based on their feedback.

8 Conclusion

Our results indicate that integrating a no-code, block-based interface with problem-oriented tutorials and practical demonstrations involving actual robots can empower inexperienced users to independently program a robot while enhancing their abilities.

We understand empowerment not only as skill acquisition, but also as the development of a feeling of control over the machine. This shift of paradigm led us to the development of a new block-based language integrated with robotic skills, enhanced with a collection of problem-based tutorials enabling the empowerment the user. While the results of the study are promising, further research is needed to better understand the key factors of worker empowerment. Indeed, if clear methodologies are available for usability and user experience, tools for designing empowering interfaces are still missing [11].

We believe that designing interfaces that will improve worker independence will be key in the future. Indeed, there is a common agreement in the literature that humans will remain central to manufacturing in the next decades [21]. Contrarily to the common vision of future workers, referred to as *Operators 4.0*, who are empowered by technologies like artificial intelligence, augmented reality, and robotics to enhance their performance and efficiency, leading to the concept of a *Super Operator* [29], our focus in this work is on identifying the key factors necessary to enable workers to become active programmers, as coined in Taylor's work [36] as *Makers 1.0*. The goal is to give back the control of the production means to the workers so that they can fully leverage their knowledge of the customer needs, the processes and the tools to implement innovative automation solutions [36].

Acknowledgements. This study was financed by the Swiss National Science Foundation (SNSF) as part of the National Research Program NRP77 Digital Transformation, project no. 187298.

References

1. Andrew, J.K., Myers, B.A., Aung, H.H.: Six learning barriers in end-user programming systems. In: 2004 IEEE Symposium on Visual Languages - Human Centric Computing, pp. 199–206 (2004). <https://doi.org/10.1109/VLHCC.2004.47>
2. Bangor, A., Kortum, P., Miller, J.: Determining what individual SUS scores mean: Adding an adjective rating scale (2009). <https://slsp-bfh.primo.exlibrisgroup.com>

3. Battarbee, K.: Defining co-experience. In: Proceedings of the 2003 international conference on Designing pleasurable products and interfaces, pp. 109–113. DPPI 2003, Association for Computing Machinery, New York, NY, USA (2003). <https://doi.org/10.1145/782896.782923>
4. Bogue, R.: Europe continues to lead the way in the collaborative robot business. *Ind. Robot Int. J.* **43**(1), 6–11 (2016). <https://doi.org/10.1108/IR-10-2015-0195>
5. Brooke, J.: SUS: A ‘Quick and Dirty’ Usability Scale, pp. 207–212. CRC Press (1996). <https://doi.org/10.1201/9781498710411-35>, <https://www.taylorfrancis.com/>
6. Digmayer, C., Jakobs, E.M.: Employee empowerment in the context of domain-specific risks in Industry 4.0. In: 2018 IEEE International Professional Communication Conference (ProComm), pp. 125–133 (2018). <https://doi.org/10.1109/ProComm.2018.00034>, iSSN: 2158-1002
7. Fantini, P., Pinzone, M., Taisch, M.: Placing the operator at the centre of Industry 4.0 design: modelling and assessing human activities within cyber-physical systems. *Comput. Ind. Eng.* **139**, 105058 (2020). <https://doi.org/10.1016/j.cie.2018.01.025>, <https://www.sciencedirect.com/science/article/pii/S0360835218300329>
8. Forlizzi, J., Battarbee, K.: Understanding experience in interactive systems. In: Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques, pp. 261–268. DIS 2004. Association for Computing Machinery, New York, NY, USA (2004). <https://doi.org/10.1145/1013115.1013152>
9. Fraser, N.: Ten things we’ve learned from Blockly. In: 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond), pp. 49–50 (2015). <https://doi.org/10.1109/BLOCKS.2015.7369000>
10. Frohm, J., Lindström, V., Winroth, M., Stahre, J.: The industry’s view on automation in manufacturing. *IFAC Proc. Vol.* **39**(4), 453–458 (2006). <https://doi.org/10.3182/20060522-3-FR-2904.00073>, <https://www.sciencedirect.com/science/article/pii/S1474667015330925>
11. Gallula, D., Ronen, H., Shichel, I., Katz, A.: User empowering design: expanding the users’ hierarchy of needs. In: Proceedings of the 6th International Conference on Computer-Human Interaction Research and Applications, pp. 201–208. SCITEPRESS - Science and Technology Publications, Valletta, Malta (2022). <https://doi.org/10.5220/0011552500003323>, <https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0011552500003323>
12. Ginat, D.: On novice loop boundaries and range conceptions. *Comput. Sci. Educ.* **14**(3), 165–181 (2004). <https://doi.org/10.1080/0899340042000302709>
13. Grover, S., Basu, S.: Measuring student learning in introductory block-based programming: examining misconceptions of loops, variables, and boolean logic. In: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, pp. 267–272. SIGCSE 2017. Association for Computing Machinery, New York, NY, USA (2017). <https://doi.org/10.1145/3017680.3017723>
14. Jain, A., Jain, P., Chan, F.T., Singh, S.: A review on manufacturing flexibility. *Int. J. Prod. Res.* **51**(19), 5946–5970 (2013). <https://doi.org/10.1080/00207543.2013.824627>
15. Kaasinen, E., et al.: Empowering and engaging industrial workers with Operator 4.0 solutions. *Comput. Ind. Eng.* **139**, 105678 (2020). <https://doi.org/10.1016/j.cie.2019.01.052>, <https://www.sciencedirect.com/science/article/pii/S036083521930066X>
16. Kadir, B.A., Broberg, O.: Human well-being and system performance in the transition to industry 4.0. *Int. J. Ind. Ergon.* **76**, 102936 (2020)
17. Kieras, D.E., Bovair, S.: The role of a mental model in learning to operate a device. *Cogn. Sci.* **8**(3), 255–273 (1984). [https://doi.org/10.1016/S0364-0213\(84\)80003-8](https://doi.org/10.1016/S0364-0213(84)80003-8), <https://www.sciencedirect.com/science/article/pii/S0364021384800038>
18. Kulić, D., Croft, E.: Pre-collision safety strategies for human-robot interaction. *Auton. Robot.* **22**(2), 149–164 (2007). <https://doi.org/10.1007/s10514-006-9009-4>

19. Major, L., Kyriacou, T., Brereton, O.P.: Systematic literature review: teaching novices programming using robots. *IET Softw.* **6**(6), 502–513 (2012). <https://doi.org/10.1049/iet-sen.2011.0125>, <https://digital-library.theiet.org/content/journals/10.1049/iet-sen.2011.0125>
20. Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., Resnick, M.: Scratch: a sneak preview [education]. In: Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004, pp. 104–109 (2004). <https://doi.org/10.1109/C5.2004.1314376>
21. Nelles, J., Kuz, S., Mertens, A., Schlick, C.M.: Human-centered design of assistance systems for production planning and control: the role of the human in Industry 4.0. In: 2016 IEEE International Conference on Industrial Technology (ICIT), pp. 2099–2104 (2016). <https://doi.org/10.1109/ICIT.2016.7475093>
22. Norman, D.A.: *The Design of Everyday Things*. Basic Books, New York, revised and expanded edition edn (2013)
23. Noroozi, A., Khakzad, N., Khan, F., MacKinnon, S., Abbassi, R.: The role of human error in risk analysis: application to pre- and post-maintenance procedures of process facilities. *Reliab. Eng. Syst. Safety* **119**, 251–258 (2013). <https://doi.org/10.1016/j.res.2013.06.038>, <https://www.sciencedirect.com/science/article/pii/S0951832013002032>
24. Pedersen, M.R., et al.: Robot skills for manufacturing: From concept to industrial deployment. *Robot. Comput. Integrated Manuf.* **37**, 282–291 (2016). <https://doi.org/10.1016/j.rcim.2015.04.002>, <https://www.sciencedirect.com/science/article/pii/S0736584515000575>
25. Ras, E., Wild, F., Stahl, C., Baudet, A.: Bridging the skills gap of workers in Industry 4.0 by human performance augmentation tools: challenges and roadmap. In: Proceedings of the 10th International Conference on Pervasive Technologies Related to Assistive Environments, pp. 428–432. PETRA 2017. Association for Computing Machinery, New York, NY, USA (2017). <https://doi.org/10.1145/3056540.3076192>, <https://dl.acm.org/doi/10.1145/3056540.3076192>
26. Resnick, M., et al.: Scratch: programming for all. *Commun. ACM* **52**(11), 60–67 (2009). <https://doi.org/10.1145/1592761.1592779>
27. Robins, A., Rountree, J., Rountree, N.: Learning and teaching programming: a review and discussion. *Comp. Sci. Educ.* **13**(2), 137–172 (2003). <https://doi.org/10.1076/csed.13.2.137.14200>
28. Rokis, K., Kirikova, M.: Challenges of low-code/no-code software development: a literature review. In: Nazaruka, E., Sandkuhl, K., Seigerroth, U. (eds.) *Perspectives in Business Informatics Research*. LNBIP, pp. 3–17. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-16947-2_1
29. Romero, D., Bernus, P., Noran, O., Stahre, J., Fast-Berglund, A.: The Operator 4.0: human cyber-physical systems & adaptive automation towards human-automation symbiosis work systems. In: Nääs, I., Vendrametto, O., Mendes Reis, J., Gonçalves, R.F., Silva, M.T., von Cieminski, G., Kiritsis, D. (eds.) *Advances in Production Management Systems. Initiatives for a Sustainable World*. IFIP Advances in Information and Communication Technology, pp. 677–686. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-51133-7_80
30. Romero, D., Stahre, J., Taisch, M.: The Operator 4.0: towards socially sustainable factories of the future. *Comp. Ind. Eng.* **139**, 106128 (2020). <https://doi.org/10.1016/j.cie.2019.106128>, <https://www.sciencedirect.com/science/article/pii/S0360835219305972>
31. Russell, J.A.: A circumplex model of affect. *J. Pers. Soc. Psychol.* **39**(6), 1161–1178 (1980). <https://doi.org/10.1037/h0077714>. place: US Publisher: American Psychological Association
32. Schou, C., Andersen, R.S., Chrysostomou, D., Bøgh, S., Madsen, O.: Skill-based instruction of collaborative robots in industrial settings. *Robot. Comput. Integrated Manuf.* **53**, 72–80 (2018). <https://doi.org/10.1016/j.rcim.2018.03.008>, <https://linkinghub.elsevier.com/retrieve/pii/S0736584516301910>

33. Sonderegger, A., Heyden, K., Chavaillaz, A., Sauer, J.: AniSAM & AniAvatar: animated visualizations of affective states. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, p. 4837 (2016). <https://doi.org/10.1145/2858036.2858365>
34. Sweller, J.: Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educ. Psychol. Rev.* **22**(2), 123–138 (2010). <https://doi.org/10.1007/s10648-010-9128-5>
35. Tan, Q., Tong, Y., Wu, S., Li, D.: Anthropocentric approach for smart assembly: integration and collaboration. *J. Robot.* **2019**, e3146782 (2019). <https://doi.org/10.1155/2019/3146782>, <https://www.hindawi.com/journals/jr/2019/3146782/>
36. Taylor, M.P., Boxall, P., Chen, J.J.J., Xu, X., Liew, A., Adeniji, A.: Operator 4.0 or Maker 1.0? Exploring the implications of Industrie 4.0 for innovation, safety and quality of work in small economies and enterprises. *Comp. Ind. Eng.* **139**, 105486 (2020). <https://doi.org/10.1016/j.cie.2018.10.047>, <https://www.sciencedirect.com/science/article/pii/S0360835218305278>
37. Terveen, L.G.: Overview of human-computer collaboration. *Knowl. Based Syst.* **8**(2), 67–81 (1995). [https://doi.org/10.1016/0950-7051\(95\)98369-H](https://doi.org/10.1016/0950-7051(95)98369-H), <https://www.sciencedirect.com/science/article/pii/095070519598369H>
38. Villani, V., Pini, F., Leali, F., Secchi, C.: Survey on human-robot collaboration in industrial settings: safety, intuitive interfaces and applications. *Mechatronics* **55**, 248–266 (2018). <https://doi.org/10.1016/j.mechatronics.2018.02.009>, <https://linkinghub.elsevier.com/retrieve/pii/S0957415818300321>
39. Weintrop, D., Afzal, A., Salac, J., Francis, P., Li, B., Shepherd, D.C., Franklin, D.: Evaluating CoBlox: a comparative study of robotics programming environments for adult novices. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, pp. 1–12. CHI 2018. Association for Computing Machinery, New York, NY, USA (2018). <https://doi.org/10.1145/3173574.3173940>
40. Weintrop, D., Shepherd, D.C., Francis, P., Franklin, D.: Blockly goes to work: Block-based programming for industrial robots. In: 2017 IEEE Blocks and Beyond Workshop (B&B), pp. 29–36 (2017). <https://doi.org/10.1109/BLOCKS.2017.8120406>
41. Wilson, H.J., Daugherty, P.R.: Collaborative Intelligence: Humans and AI Are Joining Forces. *Harvard Business Review* (2018)
42. Yanco, H.A., Drury, J.L., Scholtz, J.: Beyond usability evaluation: analysis of human-robot interaction at a major robotics competition. *Hum.-Comp. Interact.* **19**(1–2), 117–149 (2004). <https://doi.org/10.1080/07370024.2004.9667342>, <https://www.tandfonline.com/doi/abs/10.1080/07370024.2004.9667342>