2022 I-RIM Conference
October 7-9, Rome, Italy
ISBN: 9788894580532
DOI:10.5281/zenodo.7531358

Task-oriented programming for industry: a comparison with robot-oriented programming

Michele Delledonne
University of Brescia
Brescia, Italy
https://orcid.org/0000-0001-5236-2706

Enrico Villagrossi
National Research Council of Italy
Milan, Italy
https://orcid.org/0000-0002-9493-4175

Marco Faroni
National Research Council of Italy
Milan, Italy
https://orcid.org/0000-0001-9633-4301

Manuel Beschi
University of Brescia
Brescia, Italy
https://orcid.org/0000-0002-8845-2313

Nicola Pedrocchi
National Research Council of Italy
Milan, Italy
https://orcid.org/0000-0002-1610-001X

***Abstract*—The ease of use of robot programming interfaces represents a barrier to robot adoption in several manufacturing sectors because of the lack of expertise of the end-users. Current robot programming methods are mostly the past heritage, with robot programmers reluctant to adopt new programming paradigms. This work aims to evaluate the impact on non-expert users of introducing a new task-oriented programming interface that hides the complexity of a programming framework based on ROS. The paper compares the programming performance of such an interface with a classic robot-oriented programming method based on a state-of-the-art robot teach pendant. An experimental campaign involved 22 non-expert users working on the programming of two industrial tasks demonstrating a high acceptance level of the task-oriented interface with not significant difference in the learning time compared to a standard interface.**

Keywords—Intuitive robot programming, Task-oriented programming, Human-machine interaction, End-user robot programming, End-user development

I. INTRODUCTION

Despite the increasing complexity of robotic applications, the approach to robot programming has barely changed over the years: the robot program remains a rigid list of instructions coded and saved into the robot memory [1]. The robot programmers are reluctant to adopt new programming paradigms; simultaneously, the current robot programming approach is a barrier for end-users (operators without programming and robotics experience) to the spread of industrial robots in SMEs [2]. The penetration of advanced robot programming techniques, such as visual programming or programming by demonstration, is facing barriers in the industrial context [3]. The main obstacles are the robustness of the advanced programming algorithms, the complexity of the programming interfaces, and robot programmers' technical heritage. Improvements to programming interfaces are required to attract new users unfamiliar with GPL bringing together the advanced features provided by ROS with the ease of use of classical *robot-oriented* programming languages and a design that enables *task-oriented* programming.

This work compares the acceptance level, the ease of use and the effectiveness of programming industrial tasks with the GUI, developed over the framework described in [4], MFI hereafter. The comparison was with a classic *lead-through* programming approach made with the robot Teach Pendant (TP). The testers were non-expert robot programmers and end-users. The goal is to analyze if using a *task-oriented* framework, as the MFI (designed to hide the complexity), can introduce slightly longer learning time but bring several benefits compared to standard programming techniques when it is necessary to deal with tasks repetition, robot reprogramming and collision-free motion planning. An experimental campaign based on many heterogeneous users will support the results.

II. MATERIALS AND METHODS

A. *Experimental setup*

The setup used for the test was a Universal Robot UR10e with its TP, which is nowadays considered the state-of-the-art of industrial robot TPs in terms of intuitiveness and ease of use. The robot gripper is a Robotiq 2F-85. An external PC controls the robot at a frequency of 500[Hz]. A force-torque sensor was mounted between the robot flange and the gripper.

B. *Programming interface*

This work compares a *robot-oriented* and a *task-oriented* programming interface.

Robot-oriented programming focuses on primitive robot movements that the robot can perform. The user combines these primitive actions into a sequence to obtain the desired program. The robot-oriented programming interface used is the UR10e TP. This one is a highly intuitive programming interface based on a sequence of *move* instructions by teaching the starting and the ending robot configuration to be interpolated. The teaching of robot position can be done by *lead-through* programming moving the robot with the so-called manual guidance mode. To guarantee collision-free trajectories, the programmer must add intermediate robot configurations (also called via-points). The TP provides specific functions to manage the gripper activation.

Task-oriented programming focuses on the task. The user combines high-level actions by setting the parameters required by the process operation rather than the robot motion. The user does not define the primitive action from scratch, as the framework programmer previously defined the task structure. The user codes in an intuitive language. The *task-oriented* programming interface used is the MFI. Allowing the programming with high-level actions relieving the programmer from the management and the execution of single movement. MFI use a framework that can generates collision-free trajectories for a given planning scene and a given robotic system in the planning environment that can dynamically change, the user must set only the start and goal pose. The predefined actions are available *Pick, Place,* and *Go To.*

C. *Method*

The study involved a heterogeneous group of people made by university students of multiple STEM faculties and machine tool operators as real end-users from a SME. Only a

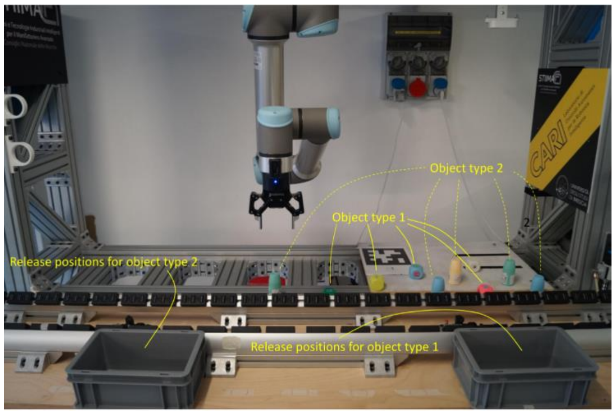

Object type 2
Object type 1
Release positions for object type 2
Release positions for object type 1

Figure 1. Setup used for Task 1.

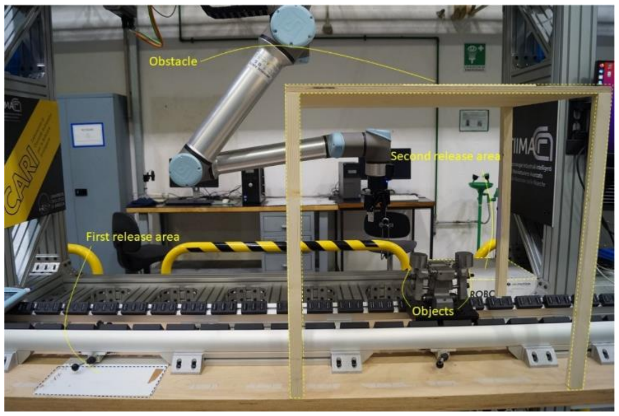


Figure 2. Setup used for Task 2.

few testers have prior experience in robot programming, not a professional one, but no one has ever seen the MFI and the TP used for the experiments.

The experiments consist of five phases:

- introduction: the user is informed about the experiment and the test phases.
- Teaching: the user watches a video that describes the interfaces (i.e., the robot TPI or the MFI) and their usage. Then, an expert operator supports the user in assisted training, where the goal is to perform a single pick and place. In this phase, the user is free to ask questions to the trainer. The training continues until the user declares he/she can program a task autonomously. Finally, the expert operator describes the user's task to program; the programming phase can start.
- Autonomous programming: the user programs the robot without the help of an expert. This phase ends when the user declares finished the task programming. The user can ask questions if he/she cannot proceed in task programming.
- Testing: testing the program developed in the previous phase. In case the task is correctly performed, this phase ends. On the contrary, the user must correct the program and test it until the task is performed completely. The task's success determines the end of this phase.
- Questionnaire: the user fills in a questionnaire regarding the intuitiveness and complexity of the interface.

Two robotics tasks were defined:

- Task 1: requires a simple Pick&Place task where 10 objects need to be picked and placed in predefined boxes, without constraints in the planning environment. This task represents a frequent application for robots in the manufacturing sector.
- Task 2: requires the manipulation of 2 objects from a constrained environment characterized by several obstacles to be picked and placed in a predefined area. This task simulates a machine tending application where the robot has to place or withdraw an object from the working area of a machine tool.

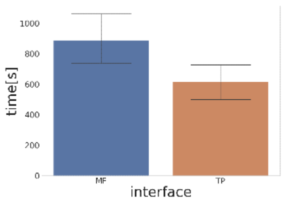
Task 1 was defined to analyze and measure the programming time of a simple but highly repetitive Pick&Place task where the adoption of a *robot-oriented* programming approach imposes the repetition of the same instructions multiple times. Task 2 was defined to analyze and measure the programming time of a complex task where the presence of obstacles and the use of a *robot-oriented* programming approach impose the definition of multiple via-points to avoid collisions. After the programming of Task 2, it was asked to the users to reprogram the task by changing the release position of the object. The testers implemented Task 1 and 2 by programming the robot in two different ways: in the first case, by writing the robot program through the TP as a collection of *"move to"* instructions teaching the trajectories via-points. In the second case, the testers implemented the same programs exploiting the MFI.

The following key points were monitored during the experimental tests of Task 1: the learning time (LeT), the programming time (PrT), the number of questions made during the programming (PrQ), the testing time (TeT), the execution time (ExT), the number of tests executed (TeN), the number of questions made during the testing (TeQ). In addition, in Tasks 2 were monitored: the reprogramming time (ReT), the reprogramming testing time (ReTeT), the reprogramming execution time (ReExT), the questions made during the reprogramming (ReQ) and retesting (ReTeQ), and the number of tests made during the testing (ReTeN). The training of the users was made by showing videos to avoid bias between subjects.

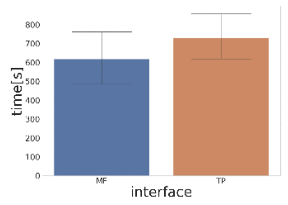
III. Results and Discussion

The experiments of Task 1 enlisted two groups of students: Group A composed of 8 people, they programmed the application using the UR10e TP. Group B composed by 9 people, they programmed the same application using the MFI. Task 2 was tested in a real shop floor of a SME with machine tool operators. In total 5 people made the experiments, due to the limited number of people, they used both the UR10e TP and the MFI.

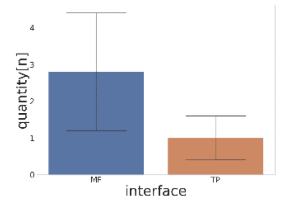
Task 1: The average learning time LeT for the MFI is 44.9% higher than the TPI. This result was expected as the MFI has more complex concepts than the TPI and the learning time tends to be higher. The average programming time PrT for the MFI is 51.7% lower than the TPI. High values of PrT for the TPI are directly related to the number of objects involved in the task because the operator needs to teach multiple positions to perform collision-free trajectories. On



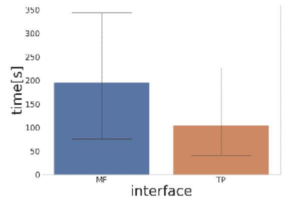
(a) Learning Time (LeT)



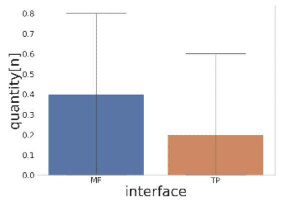
(b) Programminf Time (PrT)



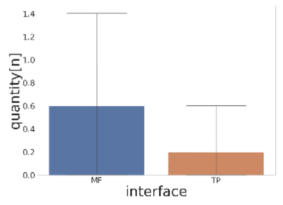
(c) Programming Questions (PrQs)



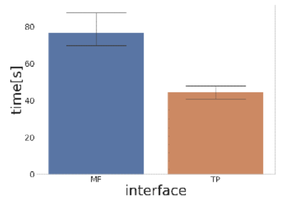
(d) Test Time (TeT)



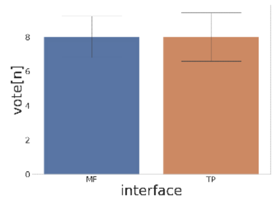
(e) Test Numbers (TeNs)



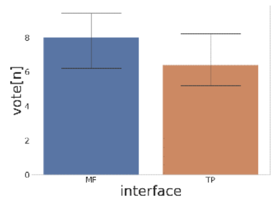
(f) Test Questions (TeQs)



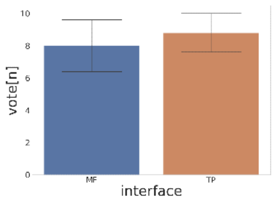
(g) Execution Time (ExT)



(h) Interface ease of use



(i) Interface intuitiveness

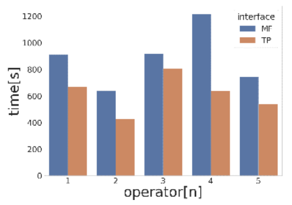


(l) Interface learning speed

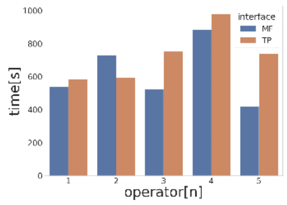
Figure 3. Task 1 experiments' results.
TP: UR10e teach pendant interface.
MF: manipulation framework interface interfacequestions

the contrary, the MFI allows to define a Pick&Place Task using a few positions. The average number of programming questions PrQs and test TeQs is low for both the interfaces. These low values represent a good operator learning rate that reflects comparable ease of use for both the interfaces. The number of tests TeNs and the test time TeT are low. Most of the experiments do not present mistakes during the programming. The low presence of mistakes avoids corrections in many experiments; when required, the

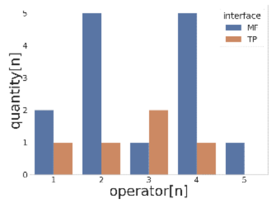
correction time is short, strengthening the result already given by PrT values. The low number of mistakes leads to overlapping the TeTs and the ExTs. The average execution time ExT is 13.5% lower for the TPI. The robot motion



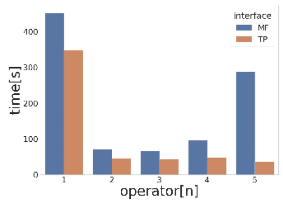
(a) Learning Time (LeT)



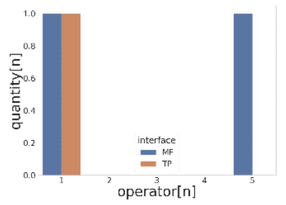
(b) Programming Time (PrT)



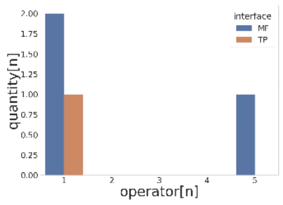
(c) Programming Questions (PrQs)



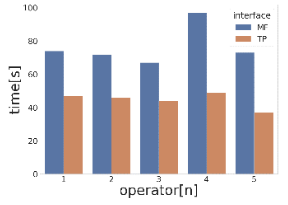
(d) Test Time (TeT)



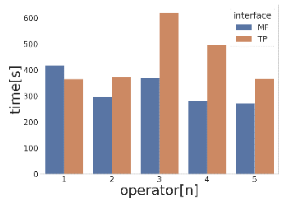
(e) Test Numbers (TeNs)



(f) Test Questions (TeQs)



(g) Execution Time (ExT)

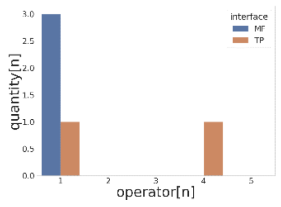


(h) Reprogramming Time (ReT)

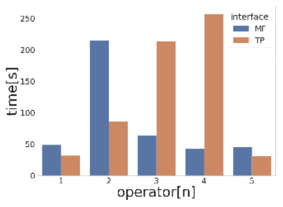
Figure 4. Task 2 experiments' results.
TP: UR10e teach pendant interface.
MF: manipulation framework interface interfacequestions

planner interpolates the trajectory via-points taught through the TPI. The time to compute the trajectories is short, and the via-points are linearly interpolated. Instead, the MFI interpolates the starting and the goal position with optimal collision-free trajectories. The computational time to evaluate the planning scene and generates the collision-free trajectories can vary. This difference explains the differences in the execution times. Despite this, the difference between the results is insignificant; furthermore, TPI ExT present a larger standard deviation than the MFI. The large standard deviation of the ExT of TPI highlights a high dependency on the user's skills. The MFI presents a small ExT standard deviation because the execution is independent of the operator capacity. At the end a questionnaire was proposed to the users. There are identical results between the MFI and the TPI regarding ease of use, intuitiveness and learning speed.

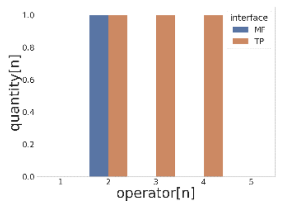
Task 2: The average learning time LeT is 43.7% higher for MFI. The result is similar to the LeT of Task 1 experiments. The average programming time PrT is 15.2% lower for the MFI. The PrT is similar for both the interfaces because Task 2 is composed of only two objects, the number of repetitive Actions is reduced tending to provide similar results. The programming questions PrQs is higher for the MFI. The users



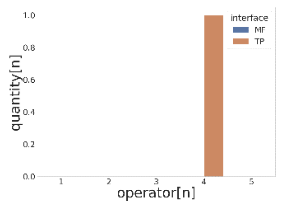
(a) Reprog. Questions (ReQs)



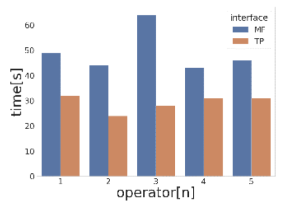
(b) Reprog. Test Time (ReTeT)



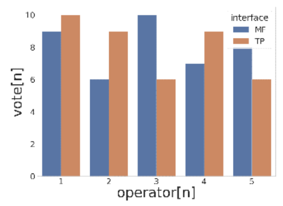
(c) Rep. Test Numbers (ReTeN)



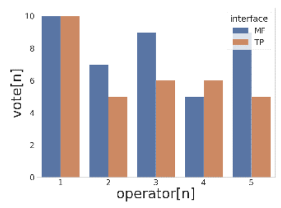
(d) Rep. Test Questions (ReTeQ)



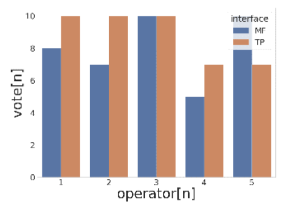
(e) Rep. Execution Time (ReExT)



(f) Interface ease of use



(g) Interface intuitiveness



(h) Interface learning speed

Figure 5. Task 2 experiments' results.
TP: UR10e teach pendant interface.
MF: manipulation framework interface interfacequestions

involved in Task 2 (i.e., shop floor machine tools operators and technicians) had less familiarity with the use of robots and, in general, less inclination to technologies compared to the users of Task 1 (i.e., STEM faculties students). This aspect is the possible cause why Task 2 PrQs values are higher than Task 1 in particular for the MFI. The more complex structure of MFI has amplified this phenomenon. Despite this problem, the PrQs values do not represent a real problem. The training of non-expert operator for the MFI is less than one hour (considering the video watching). The test time TeT shows similar TeT for both interfaces apart from the spike of user one that is anyway present for both interfaces; in particular, the time to correct the errors is comparable. The test numbers TeNs shows a not relevant number of trials, so the reduced number of errors made during the programming demonstrate that reduced knowledge does not affect the operator performance. The test questions TeQs shows a high autonomy of the user to correct the errors. The execution time ExT shows that the TPI ExT present lower values than the MFI. In this case, the computation time necessary for the MF to generate collision-free trajectories is higher than Task 1 because the robot workspace presents constrained spaces and more obstacles. The higher ExTs is reflected in the benefit of a collision-free trajectory guaranteed by the MF motion planners. On the contrary, with the TPI, the collision avoidance of the robot is in charge of the programmer. The average reprogramming time ReT shows that usually, the MFI ReT values are lower than the TPI. The MFI ReT average value is 26.1% lower than TPI. With the MFI the user has to modify the program to teach only two new positions. Instead, the TPI requires to add new via-points to the trajectories already defined. The reprogramming number of questions ReQs shows the low values; most of the users did not need any help during the reprogramming. The reprogramming test times ReTeTs, the reprogramming tests numbers ReTeN, the test questions ReTeQs, and the execution time ReExT show results similar to those obtained in the first testing phase and no significant differences emerged between the interfaces. As for Task 1, at the end of Task 2, the questionnaire was proposed to the users. There are similar results for both interfaces. The parameters have high values demonstrating a good appreciation by the machine tools operators highlighting the usability of the interfaces in the industrial world.

Conclusions

This work presents a comparison between two robot programming interfaces. The first is the UR10e TP interface that represents the state-of-the-art for intuitiveness and ease of use for industrial robots, the second is the MFI that provides advanced features such as *task-oriented* programming, state-of-the-art motion planners, collision-free motion planning, avoiding the use of textual programming. The results demonstrated that the use of an intuitive GUI that hides a complex framework can bring short learning times, anyway higher than the TP, but with the possibility of training an end-user in very little time. At the same time this instrument brings to reduced programming time and ease of use even for users without preliminary programming knowledge neither robotics expertise.

Acknowledgement

This work is supported by the ShareWork project (H2020, European Commission – G.A. 820807). Special thanks to Cembre S.p.A. for providing the opportunity to make tests on their shop floor with machine tool operators.

References

[1] Valeria Villani, Fabio Pini, Francesco Leali, Cristian Secchi, "Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications", *Mechatronics*, Volume 55, 2018, Pages 248-266, ISSN 0957-4158.

[2] Gopika Ajaykumar, Maureen Steele, and Chien-Ming Huang. "A Survey on End-User Robot Programming", *ACM Comput. Surv.*, 2021, 54, 8, Article 164 (November 2022), 36 pages.

[3] Panagiota Tsarouchi, Sotiris Makris and George Chryssolouris, "Human–robot interaction review and challenges on task planning and programming", *International Journal of Computer Integrated Manufacturing*, 2016, 29:8, 916-931.

[4] E. Villagrossi, N. Pedrocchi and M. Beschi, "Simplify the robot programming through an action-and-skill manipulation framework," *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 1-6.