robotics

MDPI

Communication

A Methodology for Flexible Implementation of Collaborative Robots in Smart Manufacturing Systems

Hermes Giberti [1,*], Tommaso Abbattista [2], Marco Carnevale [1], Luca Giagu [3] and Fabio Cristini [3]

[1] Dipartimento di Ingegneria Industriale e dell'Informazione, Università degli Studi di Pavia, Via A. Ferrata 5, 27100 Pavia, Italy; marco.carnevale@unipv.it
[2] ABB Striebel und John GmbH, 77880 Sasbach, Baden-Württemberg, Germany; tommaso.abbattista@de.abb.com
[3] Dipartimento di Meccanica, Politecnico di Milano, Via La Masa 1, 20156 Milano, Italy; luca.giagu@mail.polimi.it (L.G.); fabio1.cristini@mail.polimi.it (F.C.)
* Correspondence: hermes.giberti@unipv.it

Abstract: Small-scale production is relying more and more on personalization and flexibility as an innovation key for success in response to market needs such as diversification of consumer preferences and/or greater regulatory pressure. This can be possible thanks to assembly lines dynamically adaptable to new production requirements, easily reconfigurable and reprogrammable to any change in the production line. In such new automated production lines, where traditional automation is not applicable, human and robot collaboration can be established, giving birth to a kind of industrial craftsmanship. The idea at the base of this work is to take advantage of collaborative robotics by using the robots as other generic industrial tools. To overcome the need of complex programming, identified in the literature as one of the main issues preventing cobot diffusion into industrial environments, the paper proposes an approach for simplifying the programming process while still maintaining high flexibility through a pyramidal parametrized approach exploiting cobot collaborative features. An Interactive Refinement Programming procedure is described and validated through a real test case performed as a pilot in the Building Automation department of ABB in Vittuone (Milan, Italy). The key novel ingredients in this approach are a first translation phase, carried out by engineers of production processes who convert the sequence of assembly operations into a preliminary code built as a sequence of robot operations, followed by an on-line correction carried out by non-expert users who can interact with the machine to define the input parameters to make the robotic code runnable. The users in this second step do not need any competence in programming robotic code. Moreover, from an economic point of view, a standardized way of assessing the convenience of the robotic investment is proposed. Both economic and technical results highlight improvements in comparison to the traditional automation approach, demonstrating the possibility to open new further opportunities for collaborative robots when small/medium batch sizes are involved.

Keywords: collaborative robots; small-scale production; skill-based programming

Citation: Giberti, H.; Abbattista, T.; Carnevale, M.; Giagu, L.; Cristini, F. A Methodology for Flexible Implementation of Collaborative Robots in Smart Manufacturing Systems. *Robotics* **2022**, *11*, 9. https://doi.org/10.3390/robotics11010009

Academic Editor: Oscar Reinoso Garcia

Received: 11 November 2021
Accepted: 31 December 2021
Published: 4 January 2022

1. Introduction

The increasing diversification of customer needs as well as greater regulatory pressure [1] add further costs and complexity to industrial operations. Companies throughout the world have embraced mass customization in an attempt to avoid these pitfalls and provide unique value to their customer in an efficient manner. Readily available information technology and flexible work processes permit the customization of goods or services for individual customers in high volumes and at a relatively low cost, but mass customization can still produce unnecessary cost and complexity. Customers can no longer be thought of as members of a homogeneous market grouping. In fact, the concept of markets also needs to be redefined as customization becomes more commonplace.

Instead of focusing on homogeneous markets and average offerings, mass customizers have identified the dimensions along which their customers differ in their needs. These points of common uniqueness reveal where every customer is not the same. In addition, it is at these points that traditional offerings, designed for average requirements, create gaps in customer satisfaction: the difference between a company offering and what each customer truly desires [2].

Due to this, automation is currently undergoing a strong process of evolution, keeping the pace with an economic context characterized by dynamic manufacturing systems: demand becomes more and more unstable, with batches of production shrunk and highly customized, requiring companies to be incredibly flexible in changing from one production to another quickly and cheaply. In such production systems, as well as in artisan enterprises, robotics penetration is rather low due to the limiting characteristics of traditional automation: rigid, fixed to a single process and demanding steady and high volumes, to repay the large investment costs and risks.

To meet these needs, scientists around the world are trying to implement modern solutions to traditional robot controllers aimed at defining control procedures to allow a safe interaction between human and robot [3] and advanced programming methods based on the physical interaction between human and robot (e.g., learning by demonstration [4], gesture and voice communication [5,6]).

Some experimental user interfaces are also being proposed for traditional industrial robots, since, due to the specificity of operations they can carry out, the need of adapting their control programs to the changing assortment is an important issue [7–9]. However, collaborative robots have certainly a great potential to satisfy the needs of this variable customer demand [10,11]. They are often used as traditional automation, i.e., fixed to a single process with the only advantage of not requiring safety cages and therefore lowering investment costs [12]. On the other hand, they could be exploited on assembly lines: dynamically adaptive to new production requirements and easily reconfigured and reprogrammed to the changes in production line [13]. The idea previously investigated by researchers, and further explored in this paper, is to transform these robots into tools that can be applied by non-expert robotics users into the desired production process when they are actually needed, according to the daily production agenda, in a fast and easy way.

Four main challenges arise when trying to implement a methodology to exploit cobots in the above-mentioned conditions; these are the starting points of this project:

- Expert engineers are needed for designing, programming and testing the robotic application.
- Programmers need to know the workstation requirements for developing the precise production process.
- The robot adoption in such production process conditions has to be technically justified, since small and varied batches are involved.
- Investment costs are certainly significant, and the economic return has to be assured.

Robot programming is recently evolving from the traditional written language to new more accessible directions. For example, kinesthetic teaching applied to a walk by demonstration approach simplifies a lot the workload required for programming a machine, and its path towards industry is on track, being regulated by ISO/TS 15066.

An interesting approach in the field of easy-programming research is the task-level layered framework, as described in [14]. A productive process, referred to as a task, is constituted by a sequence of standard, modular and parametric blocks named skills [15], arranged as a state machine architecture [16]. Skills may have different structures, implying a motion and a trajectory component as in [13], or have a slightly modified template as in [17–21]. In general, a skill is composed of a pre and post check phase, where preconditions and objectives are evaluated, and an execution phase in which elementary activities or component manipulation are performed. It must be remarked that most state-of-the-art task level programming still requires engineers or experts for creating a real industrial application, thus reducing the operative range of non-expert users. Some recent work is

aimed at developing a framework that alleviates the problem of a high programming effort, e.g., exploiting the advantages of Learning by Demonstration, Learning by Programming and Learning by Interaction, combined [22].

Another relevant issue is the impossibility for programmers to standardize, in advance, the workstation, which would be needed for a reliable utilization of robots in the production process. Indeed, centering and positioning must be considered to guarantee the typical precision recognized in robotic applications. In [23] the robot can find the workstation center by means of its vision system, recognizing a QR code and using it for the correct positioning of the machine's Cartesian origin. Other calibration methods have been investigated by researchers, for example in [24] by applying hand-eye calibration or laser triangulation aided by image processing.

The last issues to be considered are related to the economic feasibility of the robotization process. Investing in robotics and automation requires great capital, usually repayable only if high and constant volumes are involved. The economic investment return can be guaranteed only by transforming robotics, typically rigidly utilized as a stable workstation, into a flexible tool that can be applied to different processes without involving significant costs or times.

The methodology proposed in this paper, named Interactive Refinement Programming (IRP), allows non-expert users to newly develop a production process by refining, step by step, built-in standard and parametric tasks in an easy way. The elements of novelty in this approach is a first translation phase carried out by engineers of production processes (in which the process to be carried out is translated into a robotic metalanguage based on skills), followed by an on-line setup and correction carried out by the non-expert user. Moreover, an economic analysis to generate a standardized way of assessing the convenience of the robotic investment is proposed. This also allows the production department to decide lot by lot the convenience of carrying out the production in a robotic or manual way.

First, a company department skilled in robotics, e.g., engineering of production processes, is responsible for a first translation of the production process from the manual activities, typically assigned to the operators, to a list of skills, thus creating a robotic metalanguage. This code structure is built by adopting pre-built skill blocks (which is a topic widely studied in the literature [17,18]) developed in our case by using the ABB code robot studio. At this stage the code is still not executable by a robot, since the created list of skills still needs an initialization and to be characterized by the actual process data. The process is then refined and finalized by a non-expert user into the real production process, by inserting the required parameters and correcting possible errors that may arise during its execution. From the point of view of industrial operation, this approach allows the shift of some operations typical of the production line department to the design stage. Each product can be conceived directly with its assembling procedure, and the robotic code structure can be directly given as an output of the design stage.

In production lines, non-skilled operators in the use of robots are enabled to provide input data to the pre-built code. The robot itself already has an inbuilt behavior that allows skill initialization. Moreover, the machine is able to drive a correction phase: even if each skill is standardly programmed, and the robot thus knows how the required activities or manipulations must be carried out, the specificities of each different workstation may cause some errors that make the task not properly executable. Built-in correction strategies are then preliminarily implemented, so that the machine can guide a non-expert user in this correction.

The paper is organized as follows: Section 2 gives an overview of the proposed methodology, of the exploited hardware and of the developed software. Section 3 details the Interactive Refinement Programming procedure. In Section 4 an economic analysis to evaluate a standardized way of assessing the robotic investment is proposed. The industrial test case and results are described in Section 5 and, finally, conclusions are drawn in Section 6.

2. Overview of the Proposed Methodology

The intrinsic properties of interaction and security of collaborative robots permit the flexibility needed in the reference context for this work, i.e., the possibility of being relocated and reused in an easy way. A widespread solution of cobots available on the market is the single-arm configuration with 6 or more Degrees of Freedom (DoF) and a payload on the order of 5–10 kg. An exception is made by the ABB YuMi, a double-arm, 7 + 7 DoF robot with a payload of 0.5 kg per arm. All the mentioned cobots present useful features that are exploitable to achieve the target of flexible implementation, such as the hand-guiding modality and the multi-modal interaction system (composed of a vision system, voice recognition and force/torque sensing), which are both very useful to interact with humans.

To obtain the required high level of flexibility, the workstation and all the related equipment (grippers, pliers or input and output systems) must be as general and standard as possible. A suitable solution consists in developing a general frame and some specific dowels per product, allowing for variation in as few components as possible when changing the production process. Obviously, the possibility of developing some specific components as well is still valid, but it would increase the implementation and setup times.

To increase system mobility, it could be useful to mount the robot on a cart, which can be manual or automatic, and to organize the shop floor into different fixed workstations which can be used both by human operators and robots independently. Each station should be of course provided with a proper blockage body and with a calibration system useful to correctly setting up the workstation itself. (This last part of activity is not carried out in the present work.)

As for the software, in order to obtain a tool that can be completely handled by a common robot-non-expert worker, the solution adopted is a pyramidal parametrized programming approach: a layered pyramidal framework composed of Primitives, Skills and Tasks, depicted in Figure 1:

- Primitives are the basic robot code command lines.
- Skills are blocks of code, composed by Primitives, that allow the robot to execute a specific operation.
- Tasks are logical and precise sequence of Skills used to allow the robot to complete an entire production process.

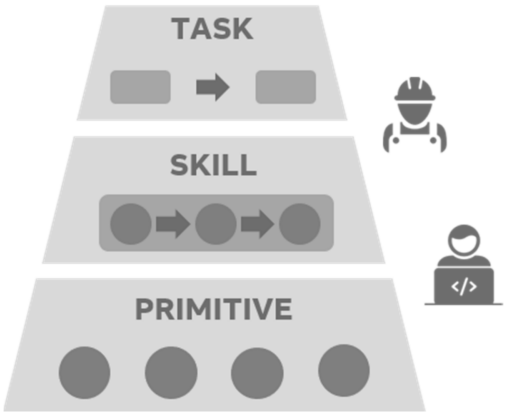


Figure 1. Layered pyramidal framework.

Each block of code, named Skill, represents a real activity or operation performed by the robot on the product to be manufactured. It is composed of a series of basilar command lines, called Primitives. The choice of creating a Skill equivalent to a production step is driven by robotics experts, in charge of developing Primitives and Skill libraries according to what the operator is used to doing during manual operations.

The structure of the considered Skills is divided into three main parts, reported in Figure 2.

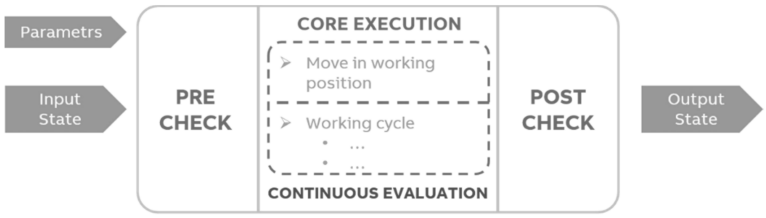


Figure 2. Structure of a standard and parametric skill.

- Pre-Check: the preliminary verification that has to be done before starting the execution of operations; the effective Execution can start only if all the Input States of the system have been verified.
- Core Execution: the parts that effectively execute the working cycle; the Core is always preceded by an initial movement to the working position, before starting the actual operation on the workpiece. In parallel to the Execution, a Continuous evaluation of the process in the workstation is carried out, and, consequently, certain methods to handle occurring errors are implemented.
- Post Check is performed when the working cycle is completed, in order to verify if the Core has correctly been executed and if the Output States are equal to the expected ones.

When a sequence of skills is created, each block, therefore, has Input and Output States which represents all the variables necessary for the execution of the current working cycle and for the execution of the following one.

Tasks, at the top of the layered pyramidal framework of Figure 1, are composed by connecting different Skills in sequence. The sequence can also present different branches, in a sort of state machine in which each block is executed when the value of the input state is correct. Different branches of the sequence can be thus executed on the basis of what has happened before.

The chosen Skill architecture is then very useful in order to keep what is happening under control and to avoid inconsistencies or mismatching errors. Input and Output States of each Skill, and thus Pre and Post Checks, are fundamental for the correct and congruent creation of a complete robot running code.

In the above-described scenario, the competencies required for creating a robotic application are split among different players. Robotics experts write the code for primitives and skill libraries, which are created after an analysis of all the involved production processes in the company, to make them as standardized as possible. The job of creating the application—generating a list of skills and thus creating a robotic metalanguage—is then assigned to the production process engineering department, who already possess the required competencies. Finally, shop floor non-expert users merely have to set input parameters for each skill during a Teaching Phase (which specifically relates each skill to the particular process), and to carry out, as a second step, a Correction phase to fix occurring errors. Non-expert users are helped in this phase by a software managing the entire teaching and correction procedure through a Graphical User Interface. These two phases will be described in detail in the following section.

Skill based programming has been extensively studied in the literature [17,18]. In this work, it has been adopted as an instrument to define and test an overall procedure accounting for every step in the flow of operations in an industrial environment, ultimately enabling the fine-tuning of a robotic cell by non-skilled operators.

3. Interactive Refinement Programming Procedure

This section describes the Interactive Refinement Programming (IRP) procedure, through which non-expert users can carry out the teaching and correction phases, ultimately leading to task execution. The main steps of the IRP are represented in Figure 3.

Figure 3. Steps of the Interactive Refinement Procedure (IRP).

3.1. Setup

Engineers in charge of designing and developing new products and manufacturing processes have to define the skill tree being executed based on operations traditionally done by hand, generating a list of skills in a robotic metalanguage. This sequence, at this stage, contains general purpose items that can still take shape accordingly to the context of the considered process. The task is not yet ready to be executed by the robot, since the inputs of each skill are still to be defined. This phase, named the Translation Phase, also includes the design and development of specific hardware or software components, in case specific adjustment of hardware or software, different from the general bundle, is required.

After the Translation Phase is completed, the operator in the production department receives the list with the sequence of Skills and the instructions to manage production (i.e., instructions and information for the operators on how to handle the workstation). The workstation can be set up and all the equipment mounted and verified according to the prescriptions in the list. When the station is ready, the setup phase is over. The operator uploads the sequence of Skills into a manager software in charge of driving the beginning of the Teaching phase for each Skill.

3.2. Teaching

If the manager software detects it is the first time a specific production process is implemented on a robot, a procedure to insert the precise input parameters for each skill is run. During this teaching phase, the user, guided by the manager software, exploits the interaction capabilities of the cobot to set the correct parameters needed for the execution of each skill. This procedure can be done in different ways, for example by using the keyboard and moving the robot into the correct positions using the teaching pendant (as in the present work), or by exploiting kinesthetic teaching (and thus moving manually the robot in the necessary positions) or, finally, by exploiting gestures [5] and voice recognition systems [6,25] or even more advanced interaction methods [26].

3.3. Correction

Once all the input parameters have been assigned, the robot contains an effectively runnable program. However, the execution could still find errors that should be detected in advance and corrected. Thus, a Test and Correction phase is performed: each operation is executed slower than the actual executing speed, so that the operator can supervise and

intervene if any error or problem occurs. In such a circumstance, the program stops and the issue can be recovered through proper editing. Some typical problems are related to the arm movements (e.g., singularity points, wrong configurations, out of reach points or physical interference with other objects in the working area) or to the gripping/contact forces.

Some predefined corrective packages have been developed and preinstalled into the manager software to drive error recovering: this is the fastest and most straightforward procedure, which, however, requires all the possible occurring errors to be known in advance to permit the development of a pertinent correction procedure for each one. The list of correction packages developed is the following:

- Automatic Trajectory Generator.
- Waypoints Addition: the movement of the robot can be stopped and some passing points can be added to the path.
- Avoid Singularity.
- Lead by Demonstration: the operator drags the robot along the desired path between two points and the robot records the trajectory in order to re-execute it.
- Force routine: the robot closes the gripper repeatedly around an object, at each iteration increasing the force until the operator confirms that the object has been grabbed correctly.

When the Correction phase has been completed for each Skill, a low-speed test is performed in order to verify the correctness of the overall task.

3.4. Execution

At this point, the Task is ready for the Execution. The complete IRP procedure is obviously necessary only the first time that the process is implemented: the parameters are then saved and made available for the next time the same product will have to be manufactured. Nevertheless, each time a task has to be redeployed, the station must be recalibrated, and thus it is appropriate to initially perform a Test Execution and, eventually, again the Teaching and Correction steps. When everything is correct, the production cycle can finally start.

In manufacturing factories whose production is highly variable and whose products are customizable, it may occur that a production process for a customized product is slightly different from the process of the basic product. Thus, this solution allows for slight modification of an existing Task and for saving this as new one by adding some other skill branches to the state machine tree. This would maintain the desired grade of flexibility required by these kinds of production systems.

4. Economic Analysis

Investing in robotics requires high costs that should be paid back by the possibility of reducing labor hours, saving annual workforce costs or increasing production capacity.

In order to assess the economic convenience of installing a robotic cell, a deeper focus on the production activities of the department is needed. Every process should be analyzed to First evaluate its "robotic feasibility" (i.e., complexity of the operations to be carried out by the robot) and then its economic impact. The threshold value N_{thr} for the lot size that makes the adoption of robots convenient is computed by comparing the total robot setup time $T_{robot\ setup}$ (from off-state to application finalized) with the time that would be required to the operator if production activities were performed manually. These two times are equaled in Equation (1):

$$T_{robot\ setup} = N_{thr} \times t_{robot\ act.} + T_{man\ setup\ for\ robot\ activities} \quad (1)$$

On the left side of the equation, the term $T_{robot\ setup}$ represents the robot setup time, consisting of the sum of:

- Robot positioning in the specific workstation (automatically or manually) and anchorage through a mechanical or magnetic braking system.

- Robot controller startup routine.
- Setup of the hardware components required for a correct execution of the task and product feeding.
- Teaching, correction and low speed test phases for every skill composing the Task.

On the right side of Equation (1), $t_{robot\ act}$ represents the manual cycle time, per unit, of the activities that should be instead allocated to the robot. When multiplied by the number of pieces in the lot (N_{thr}) this gives the time that an operator would need to manually perform all the batch operations. Finally, $T_{man\ setup\ for\ robot\ activities}$ is an additional setup time of those activities that should be made by the workforce in standard manual operation, and that instead can be allocated to the robot in the automatic procedure. The lot threshold value N_{thr} obtained by Equation (1) can finally be multiplied by a safety coefficient to adopt the robot in a more conservative way, only when the saved hours are a relevant percentage of the required workforce time.

The annual working hours saved for the specific process j ($h_{saved\ j}$) are then computed as in Equation (2) only taking into account the batches i bigger than the defined threshold value N_{thr}, which can therefore be carried out through a robotic cell. In the Equation, the saved time is evaluated by summing up, for each batch i, the manual cycle time of the activities to be automatized ($N_{batch_{i,j}} \times t_{robot\ act_j}$) and the setup time that would be necessary to the operator if the activities were performed manually and by finally subtracting the time needed to setup the robot.

$$h_{saved\ j} = \sum_{i}^{suitable\ batches} \left(N_{batch_{i,j}} \times t_{robot\ act_j} + T_{man\ setup\ for\ robot\ act_j} - T_{robot\ setup_j} \right) \quad (2)$$

Dealing with the overall investment, monetary benefit is computed in the easiest scenario by summing up, process by process, the freed person-hours multiplied by the hourly workforce cost. On the other side, the investment costs are the robot, its moveable station, and all the general and process-specific hardware and software costs for developing the required components or software modifications. Comparing these fixed costs and the annual savings, it is possible to compute the payback time and to assess the convenience for the case of interest.

Since the target companies of the proposed work deal with a truly dynamic market request, in order to introduce variability in the study, a Monte Carlo analysis has been used to simulate the demand planning and the consequent outcome on the production process.

Starting from the historical data for every demanded product of the portfolio, the probability density functions of orders' date/size and yearly frequency have been simulated, and random values have been extracted. The constraint of robot production capacity has also been introduced, and the computation of annual savings has been repeated 150 times to get a distribution of the investment benefits, whose average has been used for computing the payback time.

A specific performance indicator on an N-year basis, named *Implementation Efficiency*, has been created for benchmarking different processes in light of robotic implementation. This indicator is computed, as in Equation (3), as the ratio between an equivalent net saving (i.e., savings S_j minus implementation costs in HW and SW) and the equivalent workforce cost (computed by multiplying the hourly cost by the total man cycle time necessary to manually perform all the activities).

$$\eta_N = \frac{S_j - \frac{C_{impl}}{N}}{C_{€/h} \times t_{cycle\ tot/y}} \quad (3)$$

In the equation, N is the year horizon upon which the indicator is computed. The equation thus computes the ratio between the yearly net savings if automation was applied, and the production costs if the operator performed manually all the activities.

5. Industrial Test Case

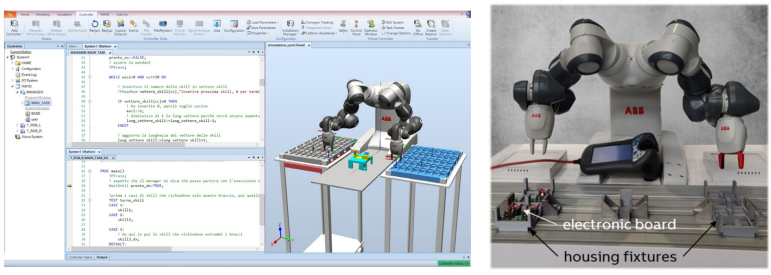
The proposed methodology has been tested in an industrial environment, in collaboration with the ABB Building Automation department in Vittuone (Milan, Italy). This factory unit, dealing with the manufacturing of personalized electronic components, is aligned with the project target, since its production is highly unstable and customized and it involves small batches and large variety and variability.

The test case is carried out on the production process regarding firmware updates in motherboards for security sensors. This working procedure is rather simple, but it contains crucial ingredients for assessing the feasibility of the proposed method. The board has to be picked out from an input tray, placed on a cable fixture connected to the PC, updated and then picked out and placed on the output tray. The ABB YuMi two-arm robot was used.

For the test case execution, the following required hardware and software was developed:

- *Hardware:* a multi-board 4 × 4 tray module, ad-hoc fixture and specific pliers (3D printed).
- *Skills:* pick and place are the two most important skills developed, completely standard and parametric, allowing for the performing of activities on a large variety of components in all the possible directions. They are developed in the ABB *Robot Studio* Software.
- *Program manager:* the program manager is a no-motion additional software uploaded on the robot controller to coordinate the entire process based on the sequence of skills generated by the engineering office. It allows the management of every phase (i.e., Teaching, Correction or Execution) for every skill, by transferring instructions to each of the two arms and displaying on a screen all the instructions the operator has to follow to define the input of each skill (as an example in the picking operation the operator has to drive the robot in the desired position and set the pose which is saved by the manager software). The communication interface of the developed program is made available on the teach-pendant screen, so that the operator can directly follow the instructions.
- *Correction packages:* lead by demonstration and waypoints addition are two routines useful for solving motion problems, by respectively recording a trajectory directly from the operator dragging the robot, or by adding waypoints interactively to avoid obstacles or force the path to be followed. Two-hands calibration is instead used for improving the precision of a place operation, autonomously calibrating the actual position of the component hold by the robot gripper by using the position of the free arm gripper as a reference (This procedure has to be activated by the operator when a precise insertion is needed). Force calibration and angle rounding routines are instead designed for easing the operator during the teaching phase and for improving the accuracy of teaching-phase parameters.

Figure 4 reports an example of the code through which the program manager code was developed (Figure 4a), and a frame of the developed robotic procedure, in which it is visible that the two arms can operate at the same time (Figure 4b). In the figure the fixture specifically designed to house the electronic boards is visible, for each of the two arms.



(a) (b)

Figure 4. Implemented Firmware up-date process for security sensor electronic boards. (**a**) Software development for manager program. (**b**) View of the two Yumi arms operating at the same time.

The robotic process has been created and set up by a non-expert user, revealing the great potential of the methodology. Two series of tests have been carried out: a first one with a non-expert user who, however, is up to date on robotic programming. The second series of tests have been carried out by a person who had never used a robot before. In both cases, the operator's learning was rather fast, since the entire process is intuitive and guided by a basic and practical graphical interface.

The results achieved by the two different users were not systematically different in terms of average values and standard deviations, so that the presented results do not differ for the single operator. This fact demonstrates that having skills in robotic does not help to improve usage skills, and the process can be therefore easily set up by every kind of non-expert user.

Table 1 reports in the second column the average teaching and correction times of the two non-expert users after several trials (time of 2nd, 3rd and 4th attempts), together with the maximum and minimum times. In the third column the times of the first trial are reported, which are around 25% higher than the following attempts in which teaching and correction times are stabilized.

Table 1. Times spent on teaching and correction phase by non-expert robotic users.

Skills	**Teaching and Correction Time** (average of 2nd, 3rd and 4th attempts for both the operators)		**1st Attempt Time** (average time on two operators)
Pick a Piece from the multi-board 4×4 tray module	1′41″	max 1′48″ / min 1′38″	2′06″
Place a piece on the fixture	2′30″	max 2′39″ / min 2′21″	3′01″
Connect to PC	-		
Pick a piece from the fixture (after SW updating)	1′49″	max 1′58″ / min 1′41″	2′11″
Place a piece on the output tray (after SW updating)	1′54″	max 2′02″ / min 1′43″	2′19″

Table 2 reports the time needed for the slow speed tests, which are carried out as a test after the teaching and correction phases for the skills of the entire task are completed, and the final cycle time per piece. By summing up the times of Tables 1 and 2 it can be observed that the complete application has been created in less than 10 min.

Table 2. Slow speed execution time and final cycle time.

Slow-speed test (i.e., test of the complete task executed after defining the input parameter for each skill)	53″
Cycle time per piece	37″

The process ended up not being as optimized in terms of trajectory and cycle time as one created by a specialist in production process engineering, but the flexibility expressed by this approach is by far more relevant than process optimization, considering the peculiar production conditions where robots can be continuously allocated to new processes. The robot could perform the task correctly even without any optimization.

From an economic point of view, a Monte Carlo simulation has been conducted on all the processes which could be feasibly carried out by robots in the considered department, in order to assess if, for the reference factory, a robotic implementation is economically valuable. These processes, in addition to the firmware update which is the considered test case from a technical point of view, are related to the following products: USB Charger (wall

charger for USB cable, involving both assemblies and customization activities), Crystal Touch Sensor (CTS electronic boards to be assembled and tested, and frontal motherboard to be tested only), Remote Controller for security sensors (sharing part of the process with other products, as firmware update and laser marking) and product customization. These are only the production processes which involve at least one activity that can be allocated to YuMi, respecting technical feasibility and robot performance. All the processes of interest have been simulated in the Monte Carlo simulation based on probability functions evaluated from historical data and forecast demand.

Table 3 reports a synthesis, for each of the mentioned processes, of the main input data (i.e., time allocated to YuMi, YuMi setup time, average lot size, pieces produced in a year) and output data obtained from the Equations (1)–(3) described previously in Section 4 (i.e., Lot Size threshold, Saved person-hours, implementation efficiency). It can be noted that the assembly of the top and bottom parts of the Crystal Touch Sensor (CTS) accounts for the great majority of saved person-hours, given the high quantities produced per year and the large percentage of activities which can be allocated to the robot.

Table 3. Single process data synthesis.

Process	Firmware Update	Remote Controller	USB Charger	CTS (Bottom)	CTS (Top)	Custom
YuMi allocated time	24 s (39%)	67 s (60%)	34.5 s (81%)	93 s (95%)	165 s (94%)	24 s (75%)
YuMi setup time	57 min	65 min	69 min	89 min	53 min	45 min
average lot size	150 pcs	50 pcs	135 pcs	280 pcs	280 pcs	200 pcs
pieces per year	3899 pcs	576 pcs	2440 pcs	9112 pcs	9880 pcs	2039 pcs
lot size threshold	139 pcs	56 pcs	117 pcs	56 pcs	19 pcs	108 pcs
saved person-hours	10.5 h	5.6 h	10 h	195.6 h	426.9 h	7.7 h
efficiency (N = 2)	8.19%	−1.31%	17.37%	75.16%	86.20%	19.51%

Investment costs are about 50,000 € (i.e., robot 40,000 €, mobile station and general hardware 2000 €, general software bundle 3000 €, operator training 3000 €, process implementation costs 2000 €), whereas saved person-hours are quantified to be in total 656 h/y, which, multiplied for an average the operator hourly cost, leads to an annual savings of 19,688 €. The payback time is therefore around 2.5 years, oscillating with ±0.5 years depending on a positive or negative scenario. In addition to the standard case, indeed, two further scenarios have been analyzed, repeating the simulation with an increased or decreased demand and batch size of 10%.

The mentioned payback scenarios are represented in Figure 5. The two additional cases reported in dashed lines make reference to a specificity of the fiscal regime that the factory department is subjected to, which further reduces the payback time of the investment.

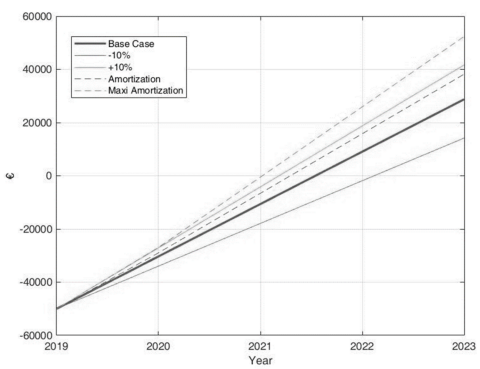


Figure 5. Comparison of Payback time scenarios.

6. Conclusions

The proposed Interactive Refinement Programming (IRP) is an approach aimed at simplifying the creation of a robotic application, allowing non-expert users to actively setup a robot guided by the machine itself, and to correct any occurring error with an iterative procedure. A pyramidal approach is based on primitives and general skills to be developed by expert engineering, which can then be connected in a tree structure to generate a specific task. The only thing that shop floor non-expert users have to do is to provide a set of input parameters for each skill during a Teaching Phase, which specifically relates each skill to the particular process, and a correction phase to fix occurring errors. These teaching and correction phases have been driven by a self-developed program manager software uploaded on the robot controller and able to display instructions on the teach-pendant screen to assist the non-skilled operator.

Ad hoc performance indicators have been created for the economic assessment of process automation benefits and for the daily management of production orders once the robotic application has been set up.

The proposed approach was validated through a real industrial case—in which the base software and hardware needed for the application were developed by expert engineers whereas the actual set-up of the robotic application was established by non-expert users in a very short time—and demonstrated the flexibility of the proposed method. The complexity in creating a robotic application has been drastically reduced in light of easy programming efforts. Collaborative features of cobots are exploited at a higher level, not only during execution but also for correcting and finalizing the application itself.

The main limitation regards the fact that the proposed method is intrinsically based on a trade-off between task optimization and flexibility: skills are designed as generally as possible to obtain the needed flexibility, which implies that the resulting tasks cannot be optimized for the specific process in terms of trajectories and cycle time.

The presented results are a further step toward the implementation of robotics in craft manufacturing, overcoming the limitations of traditional automation and current use of collaborative robots.

Author Contributions: Conceptualization, H.G. and T.A.; methodology, H.G., T.A. and M.C.; software, L.G. and F.C.; validation, T.A., L.G. and F.C.; formal analysis, H.G., T.A., L.G., F.C. and M.C.; investigation, L.G. and F.C.; resources, T.A.; data curation, L.G. and F.C.; writing—original draft preparation, M.C., L.G., F.C. and T.A.; writing—review and editing, H.G. and M.C.; visualization, T.A., L.G. and F.C.; supervision, H.G. and T.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data access is restricted due to the policy of the company involved in the work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cavadias, S.; Ladas, K.; Loch, C. The Transformative Business Model. *Har. Bus. Rev.* 2016. Available online: https://www.apdata.com/upload/file/Edition_October_2016.pdf (accessed on 30 December 2021).
2. Gilmore, J.H.; Pine, B.J., 2nd. The four faces of mass customization. *Harv. Bus Rev.* **1997**, *75*, 91–101. [PubMed]
3. Rosenstrauch, M.J.; Pannen, T.J.; Krüger, J. Human robot collaboration—Using Kinect v2 for ISO/TS 15066 speed and sepa-ration monitoring. *Procedia CIRP* **2018**, *76*, 183–186. [CrossRef]
4. Duque, D.A.; Prieto, F.A.; Hoyos, J.G. Trajectory generation for robotic assembly operations using learning by demonstration. *Robot. Comput. Manuf.* **2019**, *57*, 292–302. [CrossRef]
5. Popov, V.; Ahmed, S.; Shakev, N.; Topalov, A. Gesture-based Interface for Real-time Control of a Mitsubishi SCARA Robot Manipulator. *IFAC-PapersOnLine* **2019**, *52*, 180–185. [CrossRef]

6. Kaczmarek, W.; Panasiuk, J.; Borys, S.; Banach, P. Industrial Robot Control by Means of Gestures and Voice Commands in Off-Line and On-Line Mode. *Sensors* **2020**, *20*, 6358. [CrossRef] [PubMed]
7. Kaczmarek, W.; Lotys, B.; Borys, S.; Laskowski, D.; Lubkowski, P. Controlling an Industrial Robot Using a Graphic Tablet in Offline and Online Mode. *Sensors* **2021**, *21*, 2439. [CrossRef] [PubMed]
8. Benotsmane, R.; Dudás, L.; Kovács, G. Trajectory Optimization of Industrial Robot Arms Using a Newly Elaborated "Whip-Lashing" Method. *Appl. Sci.* **2020**, *10*, 8666. [CrossRef]
9. Urrea, C.; Jara, D. Design, Analysis, and Comparison of Control Strategies for an Industrial Robotic Arm Driven by a Multi-Level Inverter. *Symmetry* **2021**, *13*, 86. [CrossRef]
10. Gašpar, T.; Deniša, M.; Radanovič, P.; Ridge, B.; Savarimuthu, T.R.; Kramberger, A.; Priggemeyer, M.; Roßmann, J.; Wörgötter, F.; Ivanovska, T.; et al. Smart hardware integration with advanced robot programming technologies for efficient reconfiguration of robot workcells. *Robot. Comput. Manuf.* **2020**, *66*, 101979. [CrossRef]
11. Djuric, A.M.; Urbanic, R.; Rickli, J. A Framework for Collaborative Robot (CoBot) Integration in Advanced Manufacturing Systems. *SAE Int. J. Mater. Manuf.* **2016**, *9*, 457–464. [CrossRef]
12. Michalos, G.; Makris, S.; Tsarouchi, P.; Guasch, T.; Kontovrakis, D.; Chryssolouris, G. Design Considerations for Safe Human-robot Collaborative Workplaces. *Procedia CIRP* **2015**, *37*, 248–253. [CrossRef]
13. Lee, J.; Lapira, E.; Bagheri, B.; Kao, H.-A. Recent advances and trends in predictive manufacturing systems in big data environment. *Manuf. Lett.* **2013**, *1*, 38–41. [CrossRef]
14. Wahrburg, A.; Zeiss, S.; Matthias, B.; Peters, J.; Ding, H. Combined pose-wrench and state machine representation for modeling Robotic Assembly Skills. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 852–857. [CrossRef]
15. Stenmark, M.; Topp, E.A. From Demonstrations to Skills for High-Level Programming of Industrial Robots (2016) AAAI Fall Symposium; Technical Report; ELLIIT: The Linköping-Lund Initiative on IT and Mobile Communication Department of Computer Science Robotics and Semantic Systems FS-16-01–FS-16-05. pp. 75–78. Available online: https://portal.research.lu.se/en/publications/from-demonstrations-to-skills-for-high-level-programming-of-indus (accessed on 30 December 2021).
16. Herrero, H.; Moughlbay, A.A.; Outón, J.L.; Sallé, D.; de Ipiña, K.L. Skill based robot programming: Assembly, vision and Workspace Monitoring skill interaction. *Neurocomputing* **2017**, *255*, 61–70. [CrossRef]
17. Pedersen, M.R.; Nalpantidis, L.; Andersen, R.S.; Schou, C.; Bøgh, S.; Krüger, V.; Madsen, O. Robot skills for manufacturing: From concept to industrial deployment. *Robot. Comput. Manuf.* **2016**, *37*, 282–291. [CrossRef]
18. Sorensen, L.C.; Mathiesen, S.; Waspe, R.; Schlette, C. Towards Digital Twins for Industrial Assembly—Improving Robot Solutions by Intuitive User Guidance and Robot Programming. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA2020), Vienna, Austria, 8–11 September 2020; pp. 1480–1484. [CrossRef]
19. Schou, C.; Damgaard, J.; Bogh, S.; Madsen, O. Human-robot interface for instructing industrial tasks using kinesthetic teaching. *IEEE ISR* **2013**, *2013*, 1–6. [CrossRef]
20. Bøgh, S.; Nielsen, S.O.; Pedersen, R.M.; Krüger, V.; Madsen, O. Does your Robot have Skills? In Proceedings of the 43rd International Symposium on Robotics, Taipei, Taiwan, 29–31 August 2012.
21. Pedersen, M.R.; Nalpantidis, L.; Bobick, A.; Krüger, V. On the Integration of Hardware-Abstracted Robot Skills for use in In-dustrial Scenarios. In Proceedings of the 2nd International IROS Workshop on Cognitive Robotics Systems (CRS): Replicating Human Actions and Activities, Tokyo, Japan, 3–7 November 2013.
22. Akkaladevi, S.C.; Pichler, A.; Plasch, M.; Ikeda, M.; Hofmann, M. Skill-based programming of complex robotic assembly tasks for industrial application [Skill-basierte Programmierung von komplexen Roboter-Montageaufgaben für die industrielle Applikation]. *E I Elektrotechnik Und Inf.* **2019**, *136*, 326–333. [CrossRef]
23. Andersen, R.S.; Damgaard, J.S.; Madsen, O.; Moeslund, T.B. Fast calibration of industrial mobile robots to workstations using QR codes. *IEEE ISR* **2013**, *2013*, 1–6. [CrossRef]
24. Hvilshøj, M.; Bøgh, S.; Madsen, O.; Kristiansen, M. Calibration Techniques for Industrial Mobile Manipulators: Theoretical configurations and Best practices (2010) Joint. In Proceedings of the 41st International Symposium on Robotics and 6th German Conference on Robotics 2010 ISR/ROBOTIK, Munich, Germany, 7–9 June 2010; Volume 2, pp. 773–779.
25. Korayem, M.H.; Azargoshasb, S.; Tabibian, S. Design and Implementation of the Voice Command Recognition and the Sound Source Localization System for Human–Robot Interaction. *Robotics* **2021**, *39*, 1779–1790. [CrossRef]
26. Dmytriyev, Y.; Zaki, A.M.A.; Carnevale, M.; Insero, F.; Giberti, H. Brain computer interface for human-cobot interaction in industrial applications. In Proceedings of the 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 11–13 June 2021.