Mobile dual arm robotic workers with embedded cognition for hybrid and dynamically reconfigurable manufacturing systems

Grant Agreement No	**:**	**723616**
Project Acronym	**:**	**THOMAS**
Project Start Date	**:**	**1st October, 2016**
Consortium	**:**	**UNIVERSITY OF PATRAS (LMS)**
		PEUGEOT CITROEN AUTOMOBILES S.A. (PSA)
		SICK AG (SICK)
		FUNDATION TECHNALIA RESEARCH & INNOVATION (TECNALIA)
		ROBOCEPTION GMBH (ROBOCEPTION)
		DGH ROBOTICA, AUTOMATIZACION Y MANTENIMIENTO INDUSTRIAL SA (DGH)
		AERNNOVA AEROSPACE S.A.U. (AERNNOVA)
		INTRASOFT INTERNATIONAL SA (INTRASOFT)

Title	**:**	THOMAS Generalized and robust skills – Final Version
Reference	**:**	D4.5
Availability	**:**	Public
Date	**:**	30/09/2019
Author/s	**:**	TECNALIA, ROBOCEPTION
Circulation	**:**	EU, consortium

Summary:

This document contains the definition and implementation details of the necessary skills for solving different automation problematic of automotive/aeronautic use cases.

Table of Contents

1. LIST OF FIGURES

2. LIST OF TABLES

3. EXECUTIVE SUMMARY

Skills for the execution of THOMAS actions inside both use cases and their storage in the skill library is the main target of this deliverable. Already initialized skills may be found inside the skill library. Executable files for THOMAS robot may include either pre-existing skills or new skills which will be initialized by the operator as documented in deliverable D4.4. Skills' parameterization based on a user-friendly GUI developed by TECNALIA. This GUI has already be presented in deliverable D4.4. This document focuses on this GUI's usage and how the operator can use it in order to export executable files for the robot.

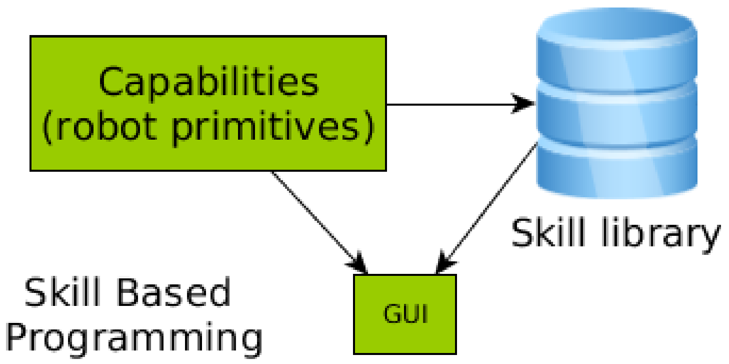


Figure 1: Skill library representation and its interaction with the GUI

WP4 and WP5 partners have developed a dedicated communication interface for the integration between the skill engine provided by TECNALIA and the station controller developed by INTRASOFT and LMS. Some tests for the connection and information exchange between these two platforms have been done and their results are presented in the following sections

Skills for the execution of all actions inside both use cases of THOMAS project have been initialized but they have been tested only in the Aeronautic one.

4. INTRODUCTION

The aim of this document is to present the implementation of skills in both THOMAS use cases. The skill library including all the pre-initialized skills and primitives of THOMAS project is available to the operator during the executable files' designing stage. The underline idea of the robot programming technique that is being investigating in THOMAS project is reducing the necessary time and effort for programming and adapting the automation of industrial processes. To achieve it, a deep analysis of the use cases is required, trying to find common elements and generalizable behaviours. The expected result consists in allowing the automatic programming of these generic operations and focusing the operator expertise and cognitive effort for parametrization of the specific details of each process. This approach is known as Skill Based Programming (SBP).

In THOMAS project, the SBP approach allows re-using generic skills for applying in similar contexts of Automotive and Aeronautics industries, that apparently have nothing in common. The combination of Skill Based Programming and CAD Programming is a step forward beyond the state of the art and allows taking advantage of the designer and operator expertise for robot programming.

In the following sections, a review of the state of the art, the architecture, morphology and application of Skill Based Programming is described.

5. THOMAS SKILL ENGINE

5.1. State of the art

The state of the art of simplified programming and execution has been performed at D1.4, however an updated analysis and review has been elaborated in this document (D4.5),

One of the key factors for increasing flexibility of robotics solutions is reducing complexity and required expertise for robot programming [1], [2], [3], [4], [5]. The classical way of programming robots, e.g. using Teach Pendants and vendor-specific robot programming languages, requires high qualified staff and increases the costs of process automation, especially in complex tasks where high flexibility is required. The need for easier programming techniques has led to the elaboration of several alternatives that reduce programming time and required expertise. Skill based programming is one approach to alleviate this drawback: allows easy, simple and intuitive robot programming [1], [2], [6].

Regarding the CAD Programming approach that THOMAS project integrates for simplified programming and execution, Computer-aided assembly or disassembly sequence planning (ASP/DSP) can be very interesting approach in order to obtain a sequence of operation to execute. The obtained sequence planning can be used as roadmap for generating a robot program automatically. This is a problem in which many authors have previously worked [7], [8], [9] obtaining better and improved assembly sequences. The obtained sequence planning can take into account the possible collisions between the elements [10], very important issue when the assembly will be automatized.

In THOMAS, a way of combining the ASP/DSP approaches with skill based programming and automatic program generation will be investigated. Taking advantage of recent progresses in this area [11] the information obtained from a CAD model is completer and more reliable.

5.1.1. State of the art – available technologies

The Skill Based Programming is based in the premise that a human behavior or action can be decomposed into three layers [12], [13]: the primitive, skill, and task layer. At the lowest level the idea is to model system capabilities in simple and intuitive symbolic units. Examples of these symbolic units (or primitives) can be the robot movement capabilities. Any capability of the system which can act atomically can be termed primitive. In the adjacent layer the skills are defined as a combination of primitives. The skills can be seen as human-like cognitive abilities [14]: can combine perception for decision taking increasing their autonomy, or by contrast, can perform simple pick and place ability with provided object and place pose. Skills that can be re-used in different contexts thanks to their parametrization. An appropriate set of parameters and interface must be selected in order to be both flexible and easily usable. In this point is where the complexity of Skill Based Programming lies, the key parameters that are available for the users must allow the skill being generic enough for adapting to different scenarios, as long as the number of parameters or their grade of complexity does not convert the skill configuration in an expert activity. Finally, the skills are organized into tasks. The task layer has a more global perspective, they are composed of the required skills to achieve the objective. The approach of task-level programming using skills is an alternative that many authors have followed [15], [16], [17]. A task can contain a sequence of skills, primitive or even another tasks.

Recently, the progress in the literature follows two main lines, the human robot interface (HRI) techniques and the efforts for achieving an industrial level for SBP. The interface between the robot and the human operator is a crucial part of SBP approach, thus, some authors has proposed physical elements that the robot system can understand [18], other efforts are focused on GUI [19], the integration of constraints can leverage the skill configuration process as well [20]. The examples of industrial applications implemented through SBP reinforce the thesis that progress is being made in this line [21], [22].

Focused on the skill implementation approach, several alternatives of skill engines have been developed in different EU projects, instead of summing up efforts to get one more mature and more applicable to industrial use. Research on flexible production lines using skills implementation is centralized in universities and research institutes with a large capacity to research in basic sciences. It is an indication of the low level of current developments. The next institutions lead a manner to implement flexible systems using semantics and skills.

Table 1: Involved institutions in skill based programming and different implementations

University or research center	EU projects and main publications
Fraunhofer (IPA)	[23], followed its development in SMErobotics [24] and LIAA [25]
Tampere University of Technology (TUT)	[26]
Aalborg University	[27,28]
Danish Technological Institute (DTI)	[29] and LIAA [25]
Technical University Munich (TUM) and University of Bremen	RoboEarth [30] and Robohow [31] projects
German Aerospace Center (DLR)	[15]
Catholic University of Leuven (KUL)	PickNPack [32] and Sherpa [33]
Fraunhofer IOSB, Karlsruhe Institute of Technology (KIT) and the Research Center for Information Technology (FZI)	[34]

5.1.2. Progress beyond the state of the art of THOMAS skill engine

Due to the variety of skill formalisms and process descriptions in the literature, and above all, the absence of a standard, it is complicated to choose and follow an existing alternative. TECNALIA, after analyzing existing definitions and formalization of the skills, has decided to focus into a pragmatic approach which allows not only taking advantage of SBP, but also being open and flexible enough for adapting current Execution Engine to possible future standardization. The key aspects of THOMAS skill engine can be summarized as follows:

- Simplified skill definition
- Integration of CAD Programming for skill configuration
- Versatility for advanced programming. XML/Python/Script mixture (abstract definition – low level implementation)

As will be presented in the following sections, a simple skill definition has been used in THOMAS Skill engine, containing basically: id, parameters and result. The possible combinations of tasks and skills are infinite, and the user can define new tasks or modify existing one anytime. For executing them in the MRP the sequence of primitive, skill and tasks are stored in a XML format.

The CAD programming integration has benefited of the mentioned simplified definition being able to extract interesting information from CAD models and for parametrizing the skills. The CAD programming combined with Skill Based Programming in one strength point of THOMAS Skill Engine.

Another of the key points of the THOMAS Skill Engine in its versatility for integrating different sources of actions. On the one hand an XML of tasks can be executed, as a *pure* SBP, however, on the other hand, Python scripts, Bash scripts, ROS actions/services, Universal Robots URScripts, etc. can be executed in the developed architecture. These modules act a black-boxes, but in some industrial environments allow re-using existing developments. These integration capabilities permit combining an easy-programing approach that provides SBP with low level, specific, advance programming that probably only can be implemented in other ways.

5.2. Skill library

Skills are mechanisms to promote the re-usability, thus a place for storing skills for further use have been designed. In the presented implementation each skill is stored in an XML file, with a unique name that allows identifying unequivocally.

This package contains XML files with skill definition. The skills stored in this package do not contain any implementation values, only definition. The process specific values are stored in each application's process files.

The skill library is queried by the execution engine and by the easy programming GUI which presents all available skills; if more skills are added to the library they are automatically discovered by them (Figure 1).

From the developed intuitive programming GUI the user can create new skills using the available robot capabilities and taking advantage of the existing skills. In the D4.4 the process of new skill creation is presented.

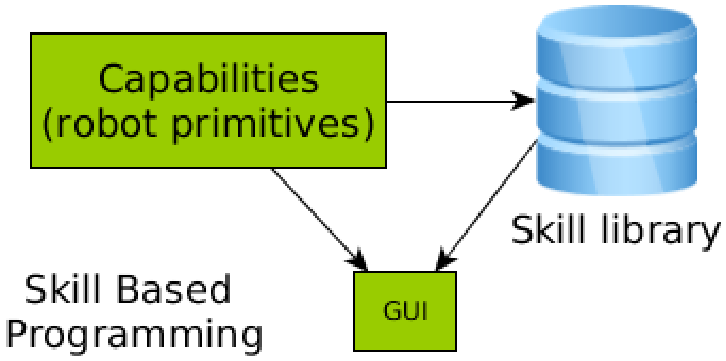
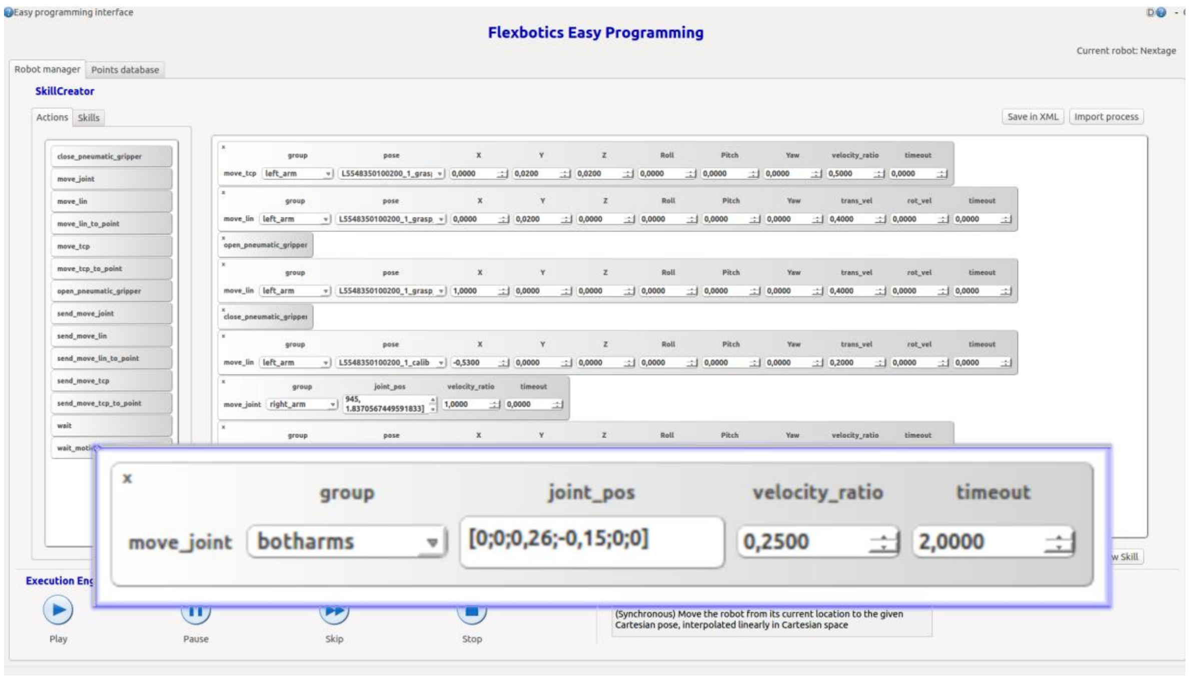


Figure 2: Skill library representation and its interaction with the GUI

5.3. Skill parametrization

The skill parametrization is performed with an easy-to-use GUI that allows building and configuring applications. Simply using the drag & drop feature new processes can be composed, and then, thanks to the information stored in the XML files or with the user provided information, the skills can be easily parametrized. Figure 2 shows how the skills and primitives can be parametrized using a GUI that currently is being developed in the WP4 – T4.1. Please refer to D4.4 for more details.

In this step the skill parameters are read taking the skill definition as schema; that means, if additional parameters are needed, the user must return to skill definition step for selecting the key parameters that should be configured. In the current implementation it is not possible changing skill definition at configuration step. The idea behind this decision is no other than keeping the skill parametrization and configuration as simpler as possible.



	group	pose	X	Y	Z	Roll	Pitch	Yaw	velocity_ratio	timeout
move_tcp	left_arm	L5548350100200_1_gras	0,0000	0,0200	0,0200	0,0000	0,0000	0,0000	0,5000	0,0000

	group	pose	X	Y	Z	Roll	Pitch	Yaw	trans_vel	rot_vel	timeout
move_lin	left_arm	L5548350100200_1_grasp	0,0000	0,0200	0,0000	0,0000	0,0000	0,0000	0,4000	0,0000	0,0000

open_pneumatic_gripper

	group	pose	X	Y	Z	Roll	Pitch	Yaw	trans_vel	rot_vel	timeout
move_lin	left_arm	L5548350100200_1_grasp	1,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,4000	0,0000	0,0000

close_pneumatic_gripper

	group	pose	X	Y	Z	Roll	Pitch	Yaw	trans_vel	rot_vel	timeout
move_lin	left_arm	L5548350100200_1_calib	-0,5300	0,0000	0,0000	0,0000	0,0000	0,0000	0,2000	0,0000	0,0000

	group	joint_pos	velocity_ratio	timeout
move_joint	right_arm	945, 1.83705674495918331]	1,0000	0,0000

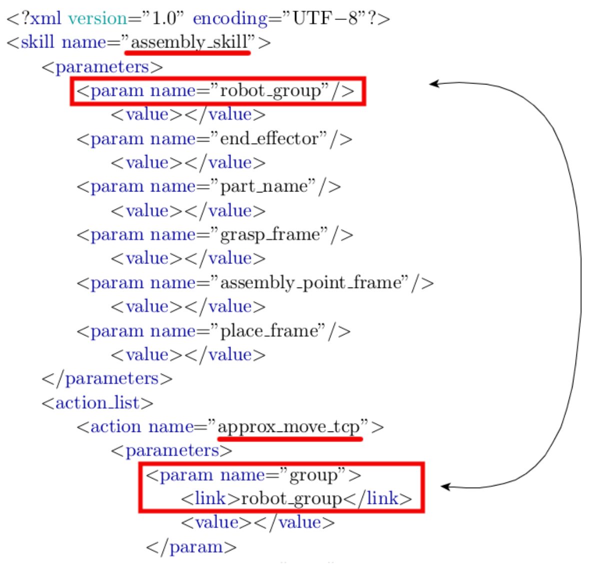
	group	pose	X	Y	Z	Roll	Pitch	Yaw	velocity_ratio	timeout

	group	joint_pos	velocity_ratio	timeout
move_joint	botharms	[0;0;0,26;-0,15;0;0]	0,2500	2,0000

(Synchronous) Move the robot from its current location to the given Cartesian pose, interpolated linearly in Cartesian space

Figure 3: Skill based programming GUI. Skill parametrization detail

After the appropriate skill parametrization and linking the expected result is an XML file that contains this information for a further execution or re-use. Figure 3 shows how the skill parameters are linked (and subsequently their values are translated) to the primitive parameters.



```
<?xml version=”1.0” encoding=”UTF−8”?>
<skill name=”assembly_skill”>
    <parameters>
        <param name=”robot_group”/>
            <value></value>
        <param name=”end_effector”/>
            <value></value>
        <param name=”part_name”/>
            <value></value>
        <param name=”grasp_frame”/>
            <value></value>
        <param name=”assembly_point_frame”/>
            <value></value>
        <param name=”place_frame”/>
            <value></value>
    </parameters>
    <action_list>
        <action name=”approx_move_tcp”>
            <parameters>
                <param name=”group”>
                    <link>robot_group</link>
                    <value></value>
                </param>
```

Figure 4: Parameter linking between skills and primitives

5.4. Skill engine interface

For the integration between the Skill Engine [TECNALIA] and Station Controller [LMS], we developed a dedicated communication interface to establish control over the network and ROS. To achieve the integration between the systems, we organized a physical workshop (hosted by TECNALIA) where we had the opportunity to fine-tune the interface and physically test the connection and information exchange.

The interface is implemented based on ROS *actionlib* library whereas Skill Engine has the role of the server, which is waiting to accept goals to execute from the station controller, and on the other hand, the Station Controller has the role of the client that interacts with the skill engine and awaits feedback on a given goal from the Skill Engine.

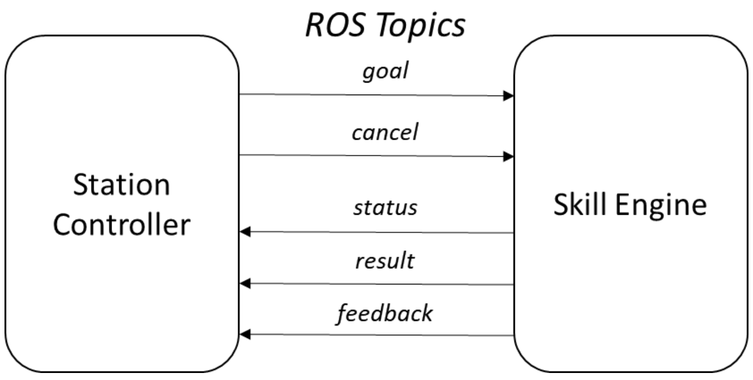


Figure 5: Station Controller - Skill Engine Interface

The above diagram depicts the Station Controller/Skill Engine interface with the individual topics including the direction of communication. For instance, the Station Controller will either send a Goal for execution or cancel an existing goal whereas the Skill Engine notifies the Station Controller with the *status* topic of each goal, the *result* of goal and the *feedback* topic for the currently executing goal. Further description is summarized in Table 2 with included description and topic types required for the communication. Additional details of the task goal, the obtained result and the feedback are summarized in the tables that can be found below (

Table **3**, Table 4 and Table 5 respectively).

Table 2: Description and type of the available topics for communication

Topic	**Description**	**Type**
/execute_task	Parent namespace for the Execution Action	flexbotics_msgs/TaskAction
/execute_task/goal	Used to send a new Process for execution by the Skill Engine	flexbotics_msgs/TaskGoal
/execute_task/cancel	Used to send cancel request to Skill Engine	flexbotics_msgs/TaskCancel
/execute_task/status	Used to notify Station Controller on the state of every goal in Skill Engine	actionlib::SimpleClientStatus
/execute_task/result	Used to notify Station Controller with information about a goal periodically	flexbotics_msgs/TaskResult
/execute_task/feedback	Used to notify Station Controller upon completion of a Goal	flexbotics_msgs/TaskFeedback

Table 3: Goal Message Definition

TaskGoal	**Description**	**Type**
process_xml	Holds the XML process	std_msgs/String

Table 4: Result Message Definition

TaskResult	**Description**	**Type**
result	Holds the Goal’s result	std_msgs/String

Table 5: Feedback Message Definition

TaskFeedback	**Description**	**Type**
status	Holds the current result of the goal	std_msgs/String

The designed architecture can be summarized in the Figure 6. In the diagram, the WP5 module represents the Station Controller.

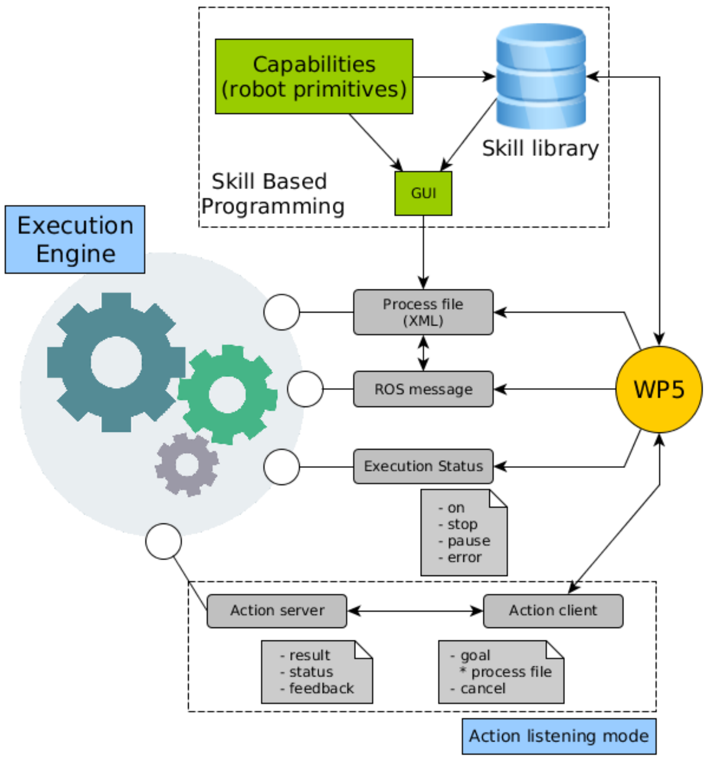


Figure 6: Summary of Skill Engine and Station Controller designed interface

In the following lines the XML files used during the workshop for executing a simple scenario using Station Controller & Skill Engine can be found. The first XML (Figure 7) contains the program description required by the Station Controller for executing the correct sequence whereas the last 2 XML files (Figure 8 and Figure 9) are the individual Process files required by the Skill Engine that describe the drilling skill and the navigation skill respectively.

```xml
<?xml version='1.0' encoding='UTF-8'?>
<program name="Workshop Demo"
        description="DEMO program for testing intergration between station-controller and
flexbotics">
        <action name="HOME" description="Goto home position" type="TASK">
                <actions>
                        <action name="navigate" type="NAVIGATE">
                                <inputs>
                                        <input name="xml" value="data/navigate.process" />
                                </inputs>
                        </action>
                        <action name="drill" type="OPERATION">
                                <inputs>
                                        <input name="xml" value="data/drill.process" />
                                </inputs>
                        </action>
                </actions>
        </action>
</program>
```

Figure 7: Station Controller Program XML File

```xml
<?xml version='1.0' encoding='UTF-8'?>
<process version="1.0">
  <action name="go_initial_position">
    <parameters />
    <result />
  </action>
  <action name="drill">
    <parameters>
      <param name="robot_group">
        <value type="data">right_arm</value>
      </param>
      <param name="drill_hole_frame_id">
        <value type="data">C02134000-001-AJMA01-0830_1_drill_1</value>
      </param>
      <param name="tool_frame_id">
        <value type="data">rarm_drilling_tool</value>
      </param>
    </parameters>
    <result />
  </action>
  <action name="drill">
    <parameters>
      <param name="robot_group">
        <value type="data">right_arm</value>
      </param>
      <param name="drill_hole_frame_id">
        <value type="data">C02134000-001-AJMA01-0830_1_drill_3</value>
      </param>
      <param name="tool_frame_id">
        <value type="data">rarm_drilling_tool</value>
      </param>
    </parameters>
    <result />
  </action>
  <action name="drill">
    <parameters>
      <param name="robot_group">
        <value type="data">right_arm</value>
      </param>
      <param name="drill_hole_frame_id">
        <value type="data">C02134000-001-AJMA01-0830_1_drill_5</value>
      </param>
      <param name="tool_frame_id">
        <value type="data">rarm_drilling_tool</value>
      </param>
    </parameters>
    <result />
  </action>
  <action name="drill">
    <parameters>
      <param name="robot_group">
        <value type="data">right_arm</value>
      </param>
      <param name="drill_hole_frame_id">
        <value type="data">C02134000-001-AJMA01-0830_1_drill_7</value>
      </param>
      <param name="tool_frame_id">
        <value type="data">rarm_drilling_tool</value>
      </param>
    </parameters>
    <result />
  </action>
  <action name="go_initial_position">
    <parameters />
    <result />
  </action>
</process>
```

Figure 8: Skill Engine – Drilling Process, XML File

```
<?xml version='1.0' encoding='utf-8'?>
<process version="1.0">
  <action name="navigate">
    <parameters>
      <param name="goal">
        <value type="data">[0.78, -0.95, 0.0, 0.0, 0.0, 3.14]</value>
      </param>
    </parameters>
    <result/>
  </action>
</process>
```

Figure 9: Skill Engine –Navigation Process, XML File

6. AERONAUTICS USE CASE REQUIRED SKILL IMPLEMENTATION

6.1. Description of the common skills for aeronautic use case

The Table 6 contains the description of the common skills that are involved in the aeronautic use case.

Table 6: Description of the common skills for aeronautics use case

Skill	Description	Key parameters
Cell to cell navigation	The MRP navigates within the areas. This navigation implies obstacle detection and avoidance.	- Target area
In cell navigation	The MRP navigates along the area in order to accomplish the required operation. This navigation implies obstacle detection and avoidance. Through the sensors located in the head an AR-marker is detected and the MRP navigates precisely to a provided relative position.	- Distance with respect to the reference (AR-marker)
Human detection	In order to guarantee safety of navigation, the MRP is continuously detecting the human presence. If a human is detected his/her position estimation is obtained.	- Sensor - Operating range
End effector exchange	The MRP exchange the end-effectors depending of the needs of the operation. The available end-effectors can be stored in different stations (usually near to the workspace of each operation). Additionally, the equipped tools can be released anytime.	- MRP arm - Required end-effector
AprilTag localization	The MRP start searching and localizing AprilTags using a sensor. Obtains an estimation of the pose of the tag.	- Tag (is necessary to provide tag id or type?) - Sensor - Theoretical position (searching area)

6.2. Description of the involved skills for drilling operation

In the Table 7 the description of the specific skills for the drilling operation can be found.

Table 7: Description of the required skills for drilling operation

Skill	Description	Key parameters
Element localization	One of the arms of MRP (or using an static sensor) moves to a theoretical position for localizing the required element using ROBOCEPTION rc_visard 160 sensor.	- Element (drilling template) - Sensor (MAC address) - Theoretical position (searching area)

Skill	Description	Key parameters
Drilling detection	The MRP detects the drillings of the template. Obtains the position and direction estimation of the drillings. After first object detection, the other MRP arm moves close to the template for a precise detection and refinement of the holes. This operation is carried out with ROBOCEPTION rc_visard 65 sensor	- Element (drilling template) - Sensor (MAC address) - Result of the element localization
Drilling	The MRP drills the localized template. This operation implies: 1. Correct localization and drilling detection 2. Appropriate end-effector for the required template 3. Collision free trajectories of the arms to the tool pre-inserting position 4. Tool inserting in the template's hole 5. Drilling-machine activation 6. Tool extraction from the template 7. Repeat 3-6 until completes all template holes.	- Detected hole pose - Specific strategy (if necessary) - Required end-effectors (can be inferred from the template model) - MRP arm

6.3. Description of the involved skills for sanding operation

In the Table 8 the description of the specific skills for the sanding operation can be found.

Table 8: Description of the required skills for sanding operation

Skill	Description	Key parameters
Sanding	The MRP sands required zone applying pre-programmed paths. Sanding program can be provided teaching by demonstration or using off-line programming tools.	- Element (part id) - MRP arm - Required end-effector

6.4. Description of the involved skills for rivet inspection operation

In the Table 9 the description of the required skills for the rivet inspection operation can be found.

Table 9: Description of the required skills for rivet inspection operation

Skill	Description	Key parameters
Rivet inspection	The MRP performs a rivet installation quality inspection. For that purpose, the MRP arm moves to an inspection zone and starts a vision or sensor-based inspection routines.	- Element (part id) - MRP arm - Required end-effector

6.5. Implementation of the involved skills for the aeronautic use case

In the following sections the current implementation of the skills is presented. Each skill has been tested using the MRP in Tecnalia’s workshop, replicating the pilot cell designed by Aernnova. As can be appreciated in Figure 10 an actual wing is divided into three sections, and each section will be used for demonstrating different operations of the use case, namely, drilling, sanding and rivet inspection.

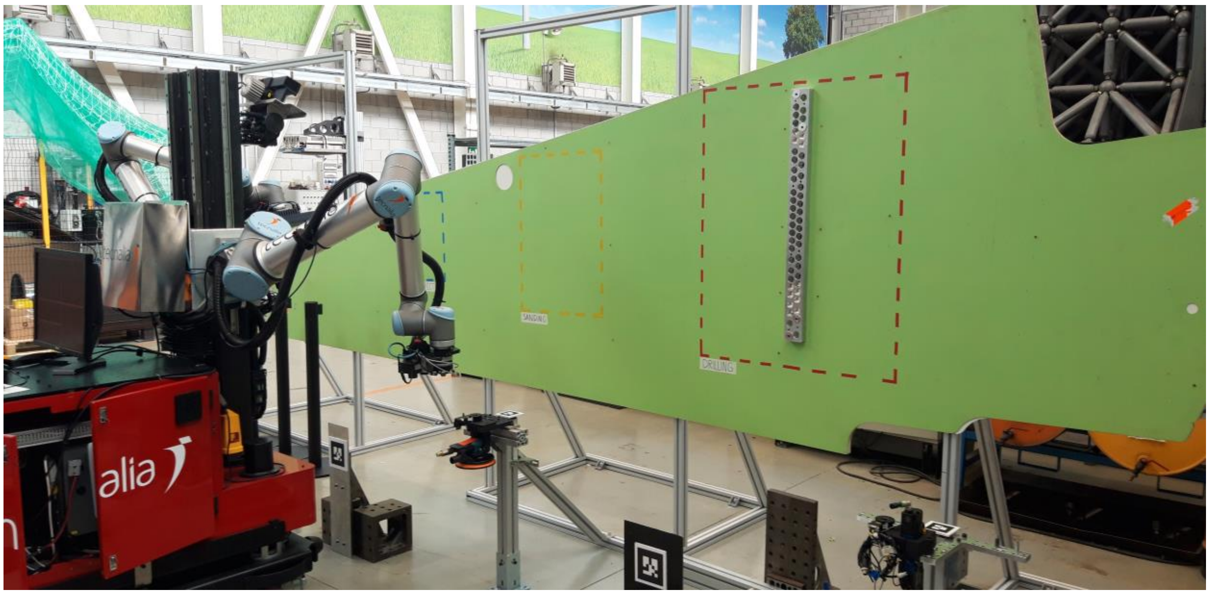


Figure 10: Current pilot cell for aeronautics use case in Tecnalia's facilities

6.5.1. Cell to cell navigation

The MRP navigates within the areas. This navigation implies obstacle detection and avoidance. The skill receives a target pose as a goal; the rest of configuration is tuned by navigation experts (see Figure 11). The precision of the cell to cell navigation is in the range of 10 cm, therefore after cell to cell navigation, in cell navigation skill is necessary to execute for achieving the required localization precision.

6.5.2. In cell navigation

The MRP navigates along the area to position precisely in the requested location. This navigation implies obstacle detection and avoidance. Through the camera located in the front of the MRP an AR-marker is detected and the MRP navigates precisely to a provided relative position. The in-cell navigation skill precision range is less than a millimetre, which allows performing repetitive actions with the manipulators of the MRP. In the aeronautic use-case the MRP is docked into a compressed-air station (see Figure 12).

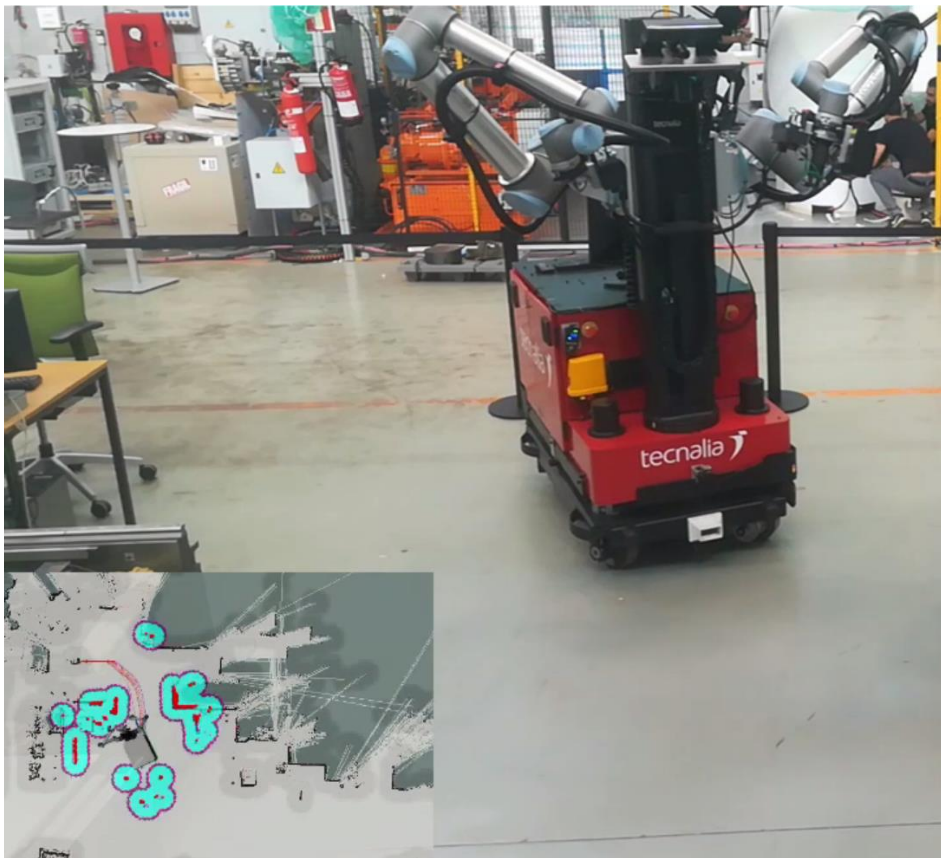


Figure 11: Navigation skill allow MRP navigates providing a target pose

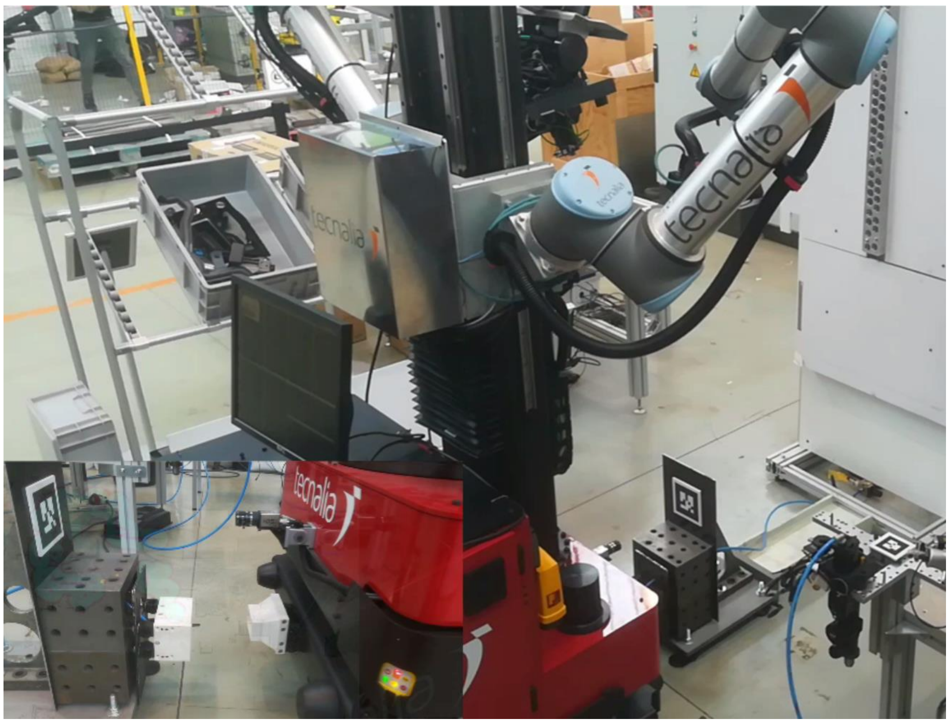


Figure 12: Precise navigation allows the MRP docking into a compressed-air station

6.5.3. End-effector exchange

The MRP can exchange the end-effectors depending the needs of the operation. The required end-effectors are stored in different stations near the workspace of each operation. When an end-effector request is received, the MRP's arm moves to an approach position from where a marker is visible. The available tools are in a fixed position respect this marker, allowing the required precision for the Schunk tool exchanger operation. The exchange skill is designed to manage all the required collision free trajectories, AR-marker detection and available devices; only providing the MRP's arm and desired end-effector (none in case of release) is enough for perform an end-effector exchange.

The Figure 13, Figure 14 and Figure 15 show the end-effectors that the MRP can exchange depending of the operation.

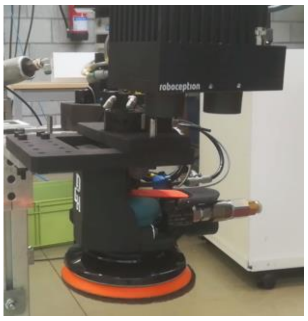


Figure 14: Sanding tool

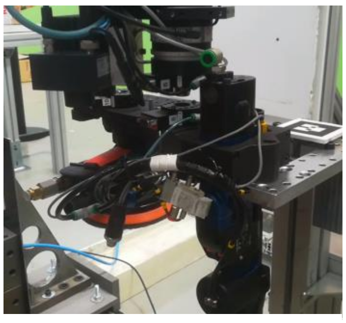


Figure 15: Setitec ADU

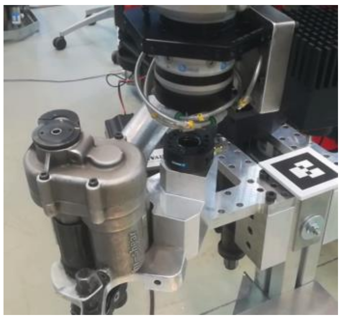


Figure 13: Hi-shear ADU

6.5.4. Template detection

One of the arms of MRP moves to a theoretical position for localizing the required element using ROBOCEPTION rc_visard 160 sensor (Figure 6). This operation obtains an estimation of the position and orientation of the drilling template. The obtained precision is not enough for the drilling requirements; therefore, additional hole detection operation is required (see Section 6.5.5). For more details about the sensor and vision algorithm please refer to D3.3.

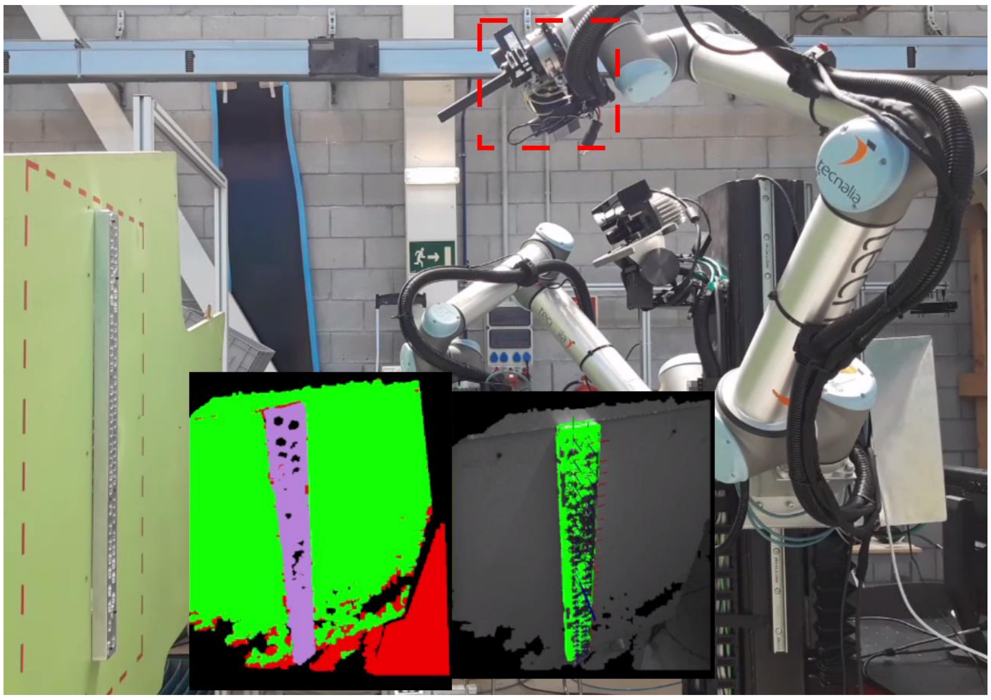


Figure 16: First template pose estimation

6.5.5. Hole detection

After the preliminary template detection operation (see Section 6.5.4), the position and orientation of the drilling template is known. Since the required precision for drilling operation is high additional vision operation is mandatory. The drilling template is divided into smaller sections that contains at most 10 holes, and then the other MRP arm moves close to these template sections for a precise detection and refinement of the holes. This operation is carried out with ROBOCEPTION rc_visard 65 sensor, as can be seen in Figure 17. For more details about the sensor and vision algorithm please refer to D3.3.

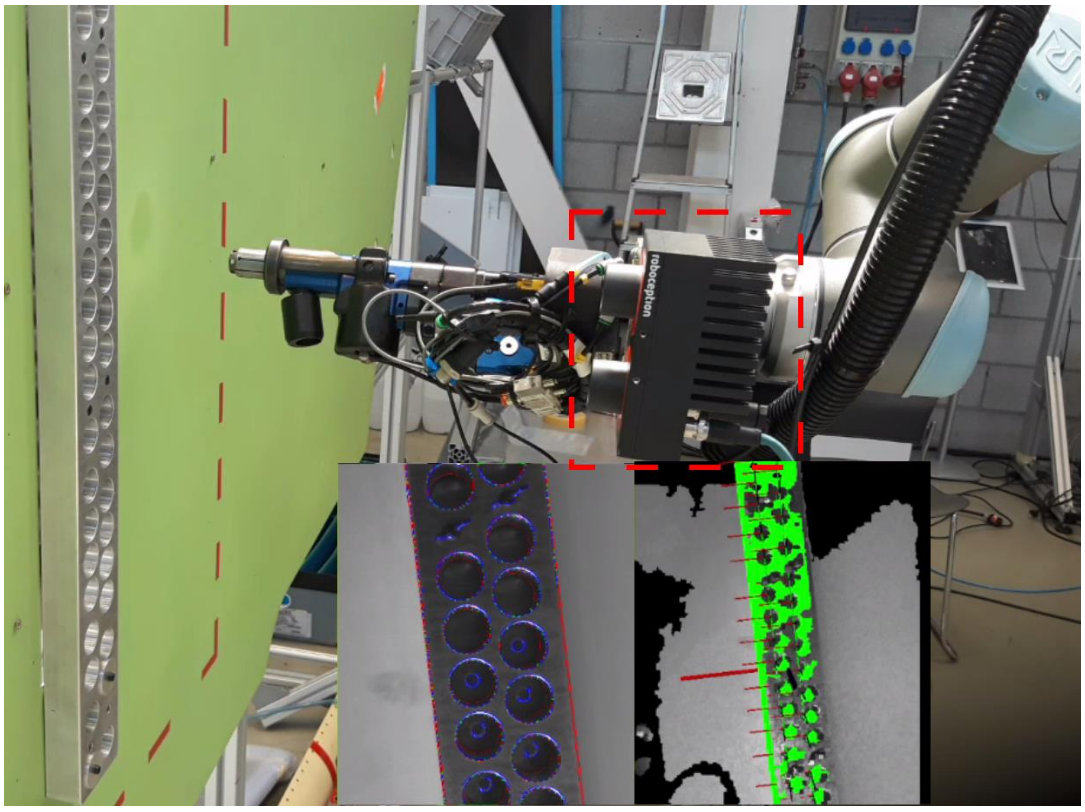


Figure 17: Precise detection of the template holes

6.5.6. Drilling

The MRP drills the located template holes using a semiautomatic Setitec ADU. In the previous phases of the project a manual Hi-shear driller had been used, nevertheless, the drilling skill has been updated for adapting to a more modern and lighter device as Setitec ADU. This operation implies the following:

1. Correct template localization and precise drilling detection.
2. Appropriate end-effector for the required template.
3. Collision free trajectories of the arms to the tool pre-inserting position.
4. Tool inserting in the template holes.
5. Drilling-machine activation.
6. Tool extraction from the template.
7. Repeat 3-6 until completes all template holes.

The mock-up of the cell prepared in Tecnalia's facilities is designed for performing drilling trials in aluminium test sheets. On this way, fast experiments with different strategies for the drilling process can be performed, being able to analyse the quality of the produced drills. In the Figure 19 the first drilling results can be seen.

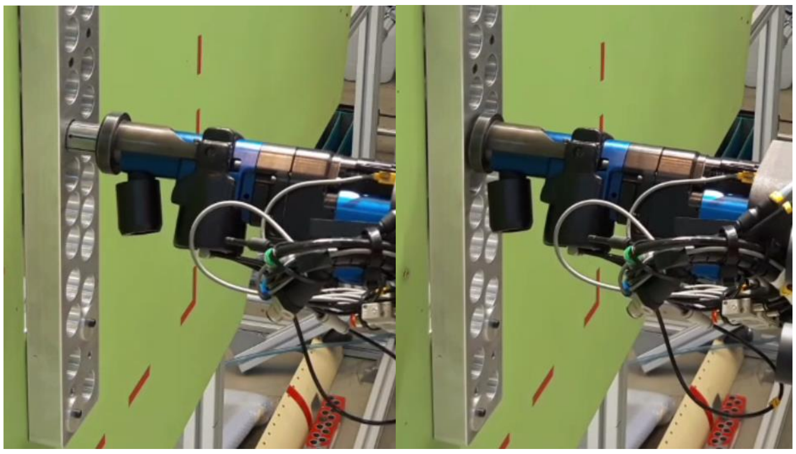


Figure 18: Drilling with Setitec ADU

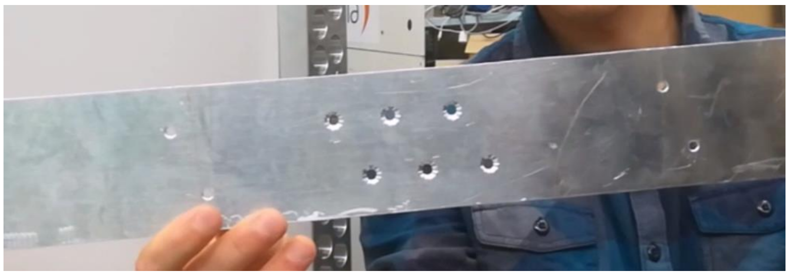


Figure 19: First drilling results into aluminium test sheet

6.5.7. Sanding

The MRP sands required zone applying pre-programmed paths. Sanding program can be provided teaching by demonstration or using off-line programming tools. To guarantee the applied force and maintain the correct perpendicularity, an OnRobot HEX 6-axis force/torque sensor has been integrated in the MRP's arm. Thanks to this element the sanding tool can follow a manually taught path applying a constant force and orientation. Figure 20 and Figure 21 presents an example of sanding in a curved aeronautic part and in a flat wing part respectively

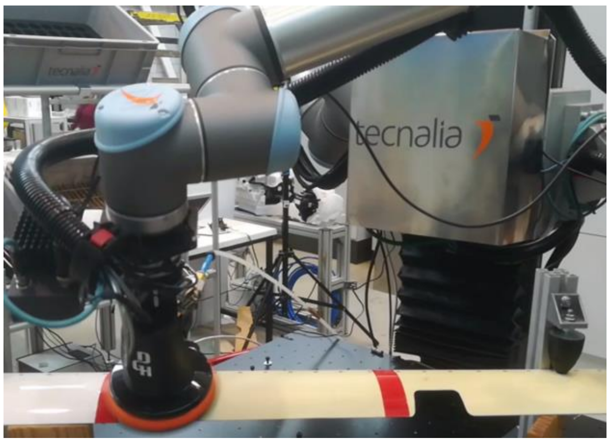


Figure 20: Sanding example in a curved part

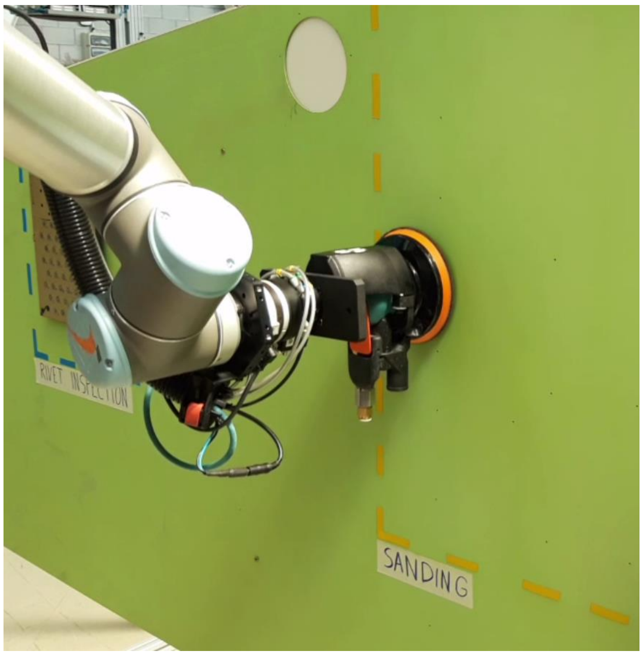


Figure 21: Sanding example in a flat part

6.5.8. Rivet inspection

The MRP performs a rivet installation quality inspection. For that purpose, the MRP arm moves to an inspection zone and starts a vision or sensor-based inspection routines (Figure 22).

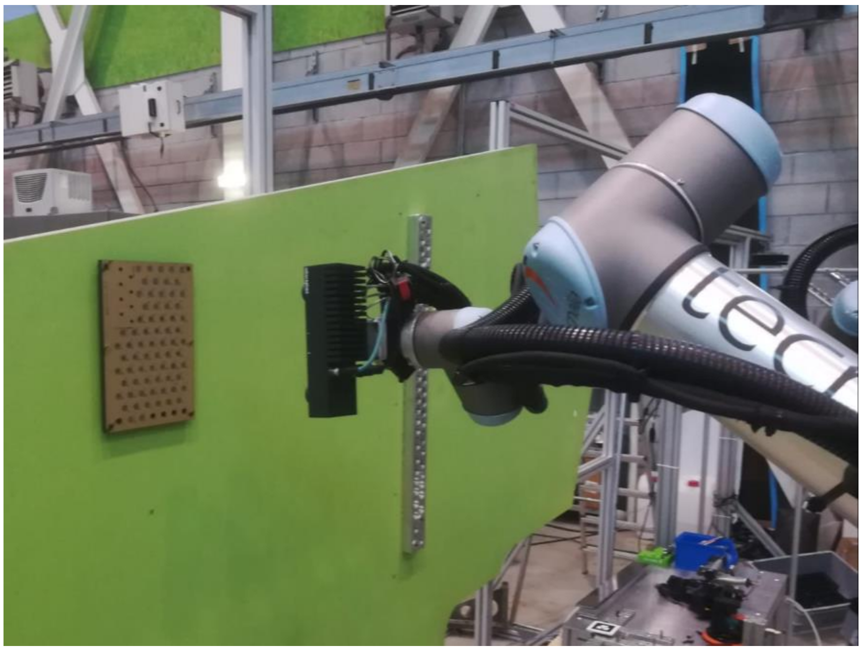


Figure 22: Rivet inspection example

7. AUTOMOTIVE USE CASE SKILL IMPLEMENTATION

7.1. Description of the involved skills for front axle damper assembly

In the Table 5 the updated description of the required skills for the front axle damper assembly operation can be seen.

Table 10: Updated description of the required skills for the front axle damper assembly operation

Skill	Description	Key parameters
Cell to cell navigation	The MRP navigates within the areas. This navigation implies obstacle detection and avoidance.	- Target area
In cell navigation	The MRP navigates along the area in order to accomplish the required operation. This navigation implies obstacle detection and avoidance. Through the sensors located in the head an AR-marker is detected and the MRP navigates precisely to a provided relative position.	- Distance with respect to the reference (AR-marker)
Human detection	In order to guarantee safety of navigation, the MRP is continuously detecting the human presence. If a human is detected his/her position estimation is obtained.	- Sensor - Operating range
Docking to the MPP	The MRP docks to the AGV (MPP). This operation requires the synchronization of movement of the MRP with the AGV. Also implies the detection of the AGV for correct positioning.	- AGV id - AGV path - AGV speed - Required sensors
End effector exchange	The MRP exchange the end-effectors depending of the needs of the operation. Some of the end effectors are stored in the back of the MRP; if the required end-effectors are located in another area in cell navigation or cell to cell navigation will be required.	- MRP arm - Required end-effector
AprilTag localization	The MRP start searching and localizing AprilTags using a sensor. Obtains an estimation of the pose of the tag.	- Tag (is necessary to provide tag id or type?) - Sensor - Theoretical position (searching area)
Element localization	One of the arms of MRP (or using an static sensor) moves to a theoretical position for localizing the required element using ROBOCEPTION rc_visard 160 sensor.	- Element (drilling template) - Sensor (MAC address) - Theoretical position (searching area)
Pick element	The MRP picks the required element with the specified arm. This operation implies the	- Element

Skill	Description	Key parameters
	following: 1. Collision free trajectories for approximating to theoretical position. 2. Element localization skill 3. MRP arm movement to element position 4. Gripper activation 5. Return to approximate position	- MRP arm - Required gripper (inferred from the element) - Theoretical position - Localization skill
Pick and place	The MRP picks the required element with the specified arm. This operation implies the following: 1. Pick element skill 2. Collision free trajectories to target position approximate position 3. Place the object in target position 4. Target place localization 5. Gripper opening. 6. Return to a safe position Note: some elements may require a specific pick and place skill due to their particularities. In further phases of the project these needs will be analysed.	- Element - MRP arm - Required gripper (inferred from the element) - Theoretical position - Localization skill - Target position
Assembly	The MRP assembly previously picked element. This operation is a special case for pick and place skill because specific strategy could be necessary for assembling the element. This operation implies: 1. Collision free trajectories to target position approximate position 2. Target place localization 3. Assembly the object in target position 4. Gripper opening. 5. Return to a safe position Note: some elements will probably require a specific assembly skill due to their particularities. In further phases of the project these needs will be analysed	- Element - MRP arm - Required gripper (inferred from the element) - Theoretical position - Assembly strategy - Localization skill - Target position
Screw	The MRP screws required elements using an specific screwdriver. This operation implies the following: 1. Collision free trajectories to element position approximate position	- Element - MRP arm - Required end-effector - Theoretical position

Skill	Description	Key parameters
	2. Element localization	- Localization skill
	3. Screwdriver insertion	
	4. Screwdriver activation	
	5. Return to a safe position	

7.2. Implementation of the involved skills for front axle damper assembly

Until this phase of the project the developments have been focused in the aeronautic use case. This does not mean that in the automotive use case has not progressed, the workflow and the common issues have been improved and matured. Even more, there are some skills that can be re-used in both use cases, namely, cell to cell navigation, in cell navigation, tool exchange, element localization, and AprilTag localization.

In the following phases of the project the focus will be translated to the automotive use case, and the lessons learned will allow solving the expected issues quickly and more effectively.

8. CONCLUSIONS

This document presents the status of the skill-based programming modules for industrial robotic applications. Moreover, how the Skill Engine interacts with the rest of the framework is introduced. Other topics covered in the document are the skill library, parametrization and interaction. Finally, the set of required skills for each pilot case is presented and a description of implemented skills can be found.

9. GLOSSARY

XML	eXtensible markup language
GUI	Graphical user interface
ROS	Robot operating system
MRP	Mobile robot platform
AR-Marker	Augmented reality marker
MPP	Mobile product platform
SBP	Skill Based Programming

10. REFERENCES

1. Krüger, N., Geib, C., Piater, J., Petrick, R., Steedman, M., Wörgötter, F., ... & Agostini, A. (2011). Object–action complexes: Grounded abstractions of sensory–motor processes. Robotics and Autonomous Systems, 59(10), 740-757.
2. Abbas, T., & MacDonald, B. A. (2011, May). Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes. In Robotics and Automation (ICRA), 2011 IEEE International Conference on (pp. 3816-3821). IEEE.
3. Biggs, G., & MacDonald, B. (2003, December). A survey of robot programming systems. In Proceedings of the Australasian conference on robotics and automation (pp. 1-3).
4. Pan, Z., Polden, J., Larkin, N., Van Duin, S., & Norrish, J. (2012). Recent progress on programming methods for industrial robots. Robotics and Computer-Integrated Manufacturing, 28(2), 87-94.
5. Lemme, A., Freire, A., Barreto, G., & Steil, J. (2013). Kinesthetic teaching of visuomotor coordination for pointing by the humanoid robot icub. Neurocomputing, 112, 179-188.
6. Pedersen, M. R., Nalpantidis, L., Andersen, R. S., Schou, C., Bøgh, S., Krüger, V., & Madsen, O. (2016). Robot skills for manufacturing: From concept to industrial deployment. Robotics and Computer-Integrated Manufacturing, 37, 282-291.
7. Shpitalni, M., Elber, G., & Lenz, E. (1989). Automatic assembly of three-dimensional structures via connectivity graphs. CIRP Annals-Manufacturing Technology, 38(1), 25-28.
8. Ben-Arieh, D., & Kramer, B. (1994). Computer-aided process planning for assembly: generation of assembly operations sequence. The international journal of production research, 32(3), 643-656.
9. Neto, P., Pires, J. N., & Moreira, A. P. (2010, June). CAD-based off-line robot programming. In 2010 IEEE Conference on Robotics, Automation and Mechatronics (pp. 516-521). IEEE.
10. Ben Hadj, R., Trigui, M., & Aifaoui, N. (2015). Toward an integrated CAD assembly sequence planning solution. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 229(16), 2987-3001
11. Belhadj, I., Trigui, M., & Benamara, A. (2016). Subassembly generation algorithm from a CAD model. The International Journal of Advanced Manufacturing Technology, 87(9-12), 2829-2840
12. Gat, E. (1998). On three-layer architectures. Artificial intelligence and mobile robots, 195, 210.
13. Björkelund, A., Edström, L., Haage, M., Malec, J., Nilsson, K., Nugues, P., ... & Linderoth, M. (2011, May). On the integration of skilled robot motions for productivity in manufacturing. In Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on (pp. 1-9). IEEE.
14. Moshkina, L., Trickett, S., & Trafton, J. G. (2014, March). Social engagement in public places: a tale of one robot. In Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction (pp. 382-389). ACM
15. Thomas, U., Hirzinger, G., Rumpe, B., Schulze, C., & Wortmann, A. (2013, May). A new skill based robot programming language using UML/P Statecharts. In Robotics and Automation (ICRA), 2013 IEEE International Conference on (pp. 461-466). IEEE.
16. Sen, S., Sherrick, G., Ruiken, D., & Grupen, R. A. (2011, August). Hierarchical Skills and Skill-based Representation. In Lifelong learning.
17. Zhou, J., Ding, X., & Yue Qing, Y. (2011). Automatic planning and coordinated control for redundant dual-arm space robot system. Industrial Robot: An International Journal, 38(1), 27-37.
18. Sefidgar, Y., Elliott, S., & Cakmak, M. (2017). A System for Situated Tangible Programming of Robot Skills. In IROS Workshop on Learning for Collaborative Robotics: Enabling Flexible, Redeployable and Agile Industrial Applications.
19. Steinmetz, F., Wollschläger, A., & Weitschat, R. (2018). RAZER—A HRI for Visual Task-Level Programming and Intuitive Skill Parameterization. IEEE Robotics and Automation Letters, 3(3), 1362-1369.
20. Polverini, M. P., Zanchettin, A. M., & Rocco, P. (2019). A constraint-based programming approach for robotic assembly skills implementation. Robotics and Computer-Integrated Manufacturing, 59, 69-81.

21. Schou, C., Andersen, R. S., Chrysostomou, D., Bøgh, S., & Madsen, O. (2018). Skill-based instruction of collaborative robots in industrial settings. Robotics and Computer-Integrated Manufacturing, 53, 72-80.
22. Huang, P. C., Hsieh, Y. H., & Mok, A. K. (2018, July). A Skill-Based Programming System for Robotic Furniture Assembly. In 2018 IEEE 16th International Conference on Industrial Informatics (INDIN) (pp. 355-361). IEEE.
23. VFF: Holistic, extensible, scalable and standard virtual factory framework. http://www.itia.cnr.it/sitiprogetti/vff/
24. SMERobotics: The European Robotics Initiative for Strengthening the Competitiveness of SMEs in Manufacturing by integrating aspects of cognitive systems. http://www.smerobotics.org/.
25. LIAA aims at the development of low-cost, low-complexity robot systems. http://www.projectleanautomation.eu/.
26. Minna Lanz, Eeva Jarvenpaa, F. G. P. L. and Tuokko, R. (2012). Towards adaptive manufacturing systems- knowledge and knowledge management systems. Manufacturing System, ISBN: 978-953-51-0530- 5, DOI: 10.5772/36015.
27. Simon Bøgh, Mads Hvilshøj, M. K. and Madsen., O (2011). Autonomous industrial mobile manipulation (aimm): From research to industry. In Proceedings of the 42nd International Symposium on Robotics.
28. Simon Bøgh, Oluf Skov Nielsen, M. R. P. V. K. and Madsen., O. Does your robot have skills? In The 43rd Intl Symp. on Robotics (ISR2012).
29. Andersen, R. H., Solund, T., and Hallam, J. (2014). Definition and initial case-based evaluation of hardwareindependent robot skills for industrial robotic coworkers. In ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of, pages 1–7.
30. Tenorth, M., Perzylo, A. C., Lafrenz, R., & Beetz, M. (2012, May). The roboearth language: Representing and exchanging knowledge about actions, objects, and environments. In 2012 IEEE International Conference on Robotics and Automation (pp. 1284-1289). IEEE.
31. Robohow: Web-enabled and Experience-based Cognitive Robots that Learn Complex Everyday Manipulation Tasks.
32. PickNpack: Flexible robotic systems for automated adaptive packaging of fresh and processed food products. http://www.picknpack.eu/.
33. Sherpa: Smart collaboration between Humans and ground-aErial Robots for imProving rescuing activities in Alpine environments. http://www.sherpaproject.eu/sherpa/.
34. Skillpro: Skill-based Propagation of ’Plug and Produce’ Devices in Reconfigurable Production Systems by AML. http://www.skillproproject.eu/.