# 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)**<br>**PEUGEOT CITROEN AUTOMOBILES S.A. (PSA)**<br>**SICK AG (SICK)**<br>**FUNDATION TECHNALIA RESEARCH & INNOVATION (TECNALIA)**<br>**ROBOCEPTION GMBH (ROBOCEPTION)**<br>**DGH ROBOTICA, AUTOMATIZACION Y MANTENIMIENTO INDUSTRIAL SA (DGH)**<br>**AERNNOVA AEROSPACE S.A.U. (AERNNOVA)**<br>**INTRASOFT INTERNATIONAL SA (INTRASOFT)** |



| | | |
|---|---|---|
| **Title** | : | THOMAS skills implementation |
| **Reference** | : | 4.3 |
| **Availability** | : | Public version |
| **Date** | : | 31.03.2018 |
| **Author/s** | : | TECNALIA, ROBOCEPTION |
| **Circulation** | : | EU, consortium |


**Summary:**

*This document contains the skill implementation prototypes for the first demonstrator [month 18]*

## Table of Contents

LIST OF FIGURES ..... 3
LIST OF TABLES ..... 4
1. EXECUTIVE SUMMARY ..... 5
2. INTRODUCTION ..... 6
3. THOMAS SKILL ENGINE ..... 7
3.1. Skill library ..... 7
3.2. Skill parametrization ..... 7
3.3. Skill engine interface ..... 8
4. AERONAUTICS USE CASE REQUIRED SKILL IMPLEMENTATION ..... 13
4.1. Description of the involved skills for drilling operation ..... 13
4.2. Implementation of the involved skills for drilling operation ..... 14
4.2.1. Cell to cell navigation ..... 14
4.2.2. In cell navigation ..... 15
4.2.3. Template detection ..... 15
4.2.4. Hole detection ..... 15
4.2.5. Drilling ..... 17
5. AUTOMOTIVE USE CASE SKILL IMPLEMENTATION ..... 18
5.1. Description of the involved skills for front axle damper assembly ..... 18
5.2. Implementation of the involved skills for front axle damper assembly ..... 20
6. CONCLUSIONS ..... 21
7. GLOSSARY ..... 22

## LIST OF FIGURES

Figure 1: Skill library ..... 7
Figure 2: Skill based programming GUI. Skill parametrization detail ..... 8
Figure 3: Parameter linking between skills and primitives ..... 8
Figure 4: Station Controller – Skill Engine Interface ..... 9
Figure 5: Station Controller Program XML File ..... 10
Figure 6: Skill Engine – Navigation Process, XML File ..... 12
Figure 7: Drilling Process, XML File ..... 12
Figure 8: Navigation skill allow MRP navigates providing a target area ..... 14
Figure 9: Precise navigation allows the MRP docking into a compressed-air station ..... 15
Figure 10: First template pose estimation ..... 16
Figure 11: Precise detection of the template holes ..... 16
Figure 12: Drilling simulation with a mock-up that simulates the ADU ..... 17

## LIST OF TABLES

Table 1: Description and topic types required for the communication ..... 9
Table 2: Goal Message Definition ..... 9
Table 3: Result Message Definition ..... 10
Table 4: Feedback Message Definition ..... 10
Table 5: Updated description of the required skills for drilling operation ..... 13
Table 6: Updated description of the required skills for the front axle damper assembly operation ..... 18

# 1. EXECUTIVE SUMMARY

The content of this document summarizes the progress in the implementation of skills for both THOMAS pilot cases. The main purpose of this document is to update the Skill Engine status. Using the developed programming GUI and skills as inputs, the user is able to create new skills. This process has been already introduced in Deliverable D4.2. The skill parametrization process is being completed through a GUI that is being developed under WP4 – T4.1 keeping the parametrization and configuration as simple as possible. The expected result from the parameterization process is being saved in an XML file for further execution or re-use.

For the communication between the Skill Engine [TECNALIA] and the Station Controller [LMS] (see deliverable D5.3), one dedicated communication interface has been developed. This interface has been tested in a physical workshop organized in TECNALIA for that purpose. Based on ROS actionlib library, the Skill Engine is waiting to accept goals to execute from the Station Controller and gives feedback about the execution process of each goal.

All robot actions of THOMAS Aeronautics and Automotive pilot case will be executed through this communication interface. The skills, which will be implemented for the front axle damper assembly operation, have not been finalized yet. However, implemented skills for the drilling operation are:

- Cell to cell navigation (navigation)
- In cell navigation (precise navigation)
- Template detection
- Hole detection
- Drilling

Some of these skills will be translated to the automotive pilot case, and the lessons learned will solve expected problems.

# 2. INTRODUCTION

In this document a developed communication interface for the message exchange between the Station Controller that is being developed from LMS and the Skill Engine of TECNALIA will be introduced. Using the same GUI and methodology as described in Deliverable D4.2, each skill can be created from a group of primitives and other skills. In the first section of this document, big emphasis is being given to the outcome of the parameterization process. The output of this process can be used for further execution or re-use.

In this communication interface, Station Controller and Skill Engine exchange messages between each other about accepted goals for execution and the status of each task. In section 3, results from the test of this communication interface are being presented. This interface has been tested in a physical workspace hosted by TECNALIA. In the next two sections, all involved skills for the execution of all THOMAS tasks in both Aeronautics and Automotive use case are going to be described.

# 3. THOMAS SKILL ENGINE

## 3.1. 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 a 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).

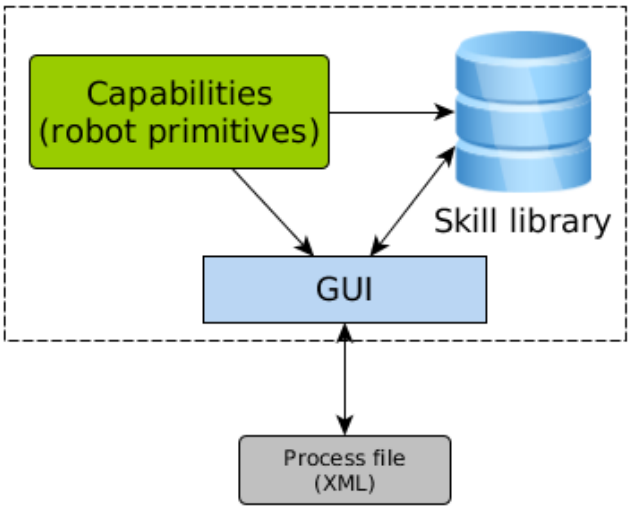


**Figure 1: Skill library**

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 D4.2 the process of new skill creation is presented.

## 3.2. 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 D4.2 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.

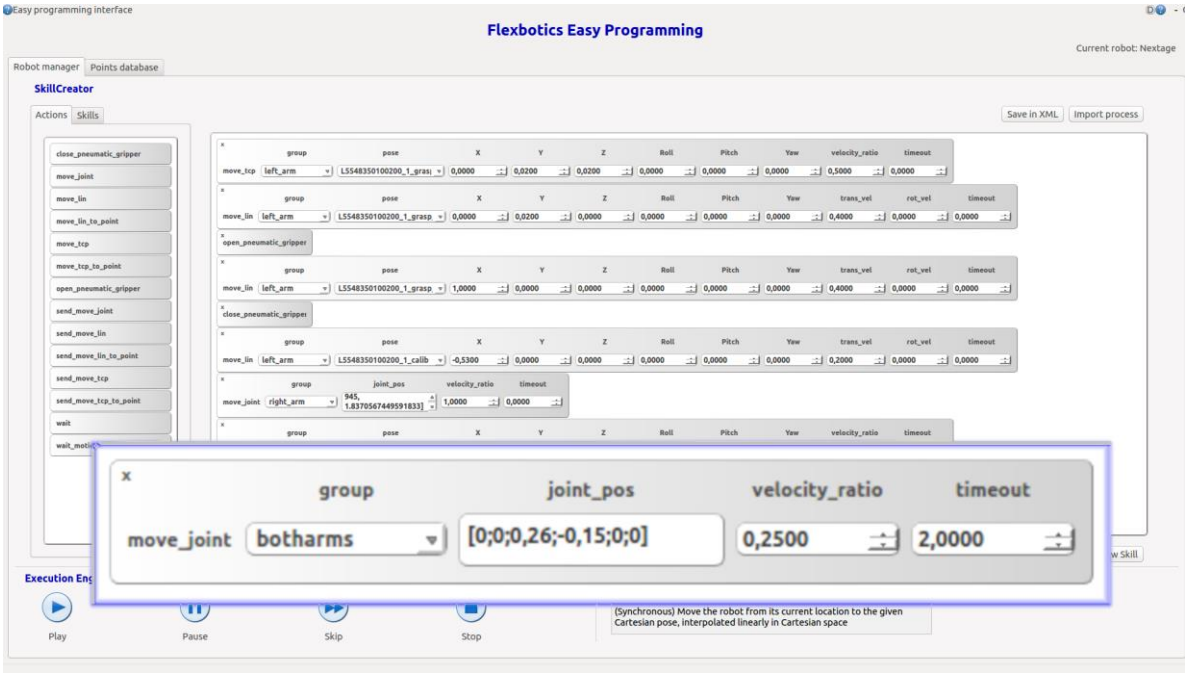


**Figure 2: Skill based programming GUI. Skill parametrization detail**

After the appropriate skill parametrization and linking, the expected result is a 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 3: Parameter linking between skills and primitives**

## 3.3. 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.

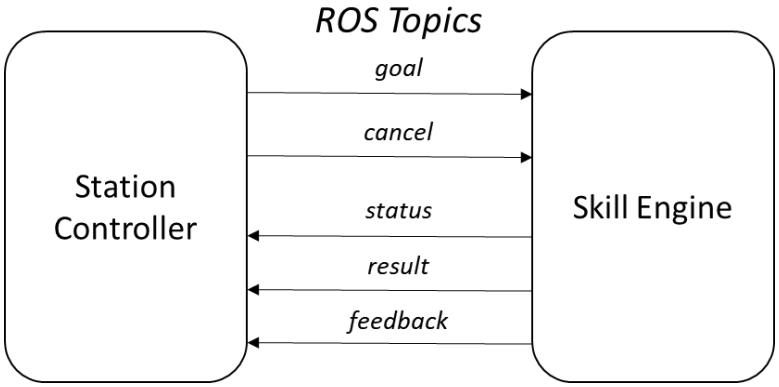


**Figure 4: Station Controller – Skill Engine Interface**

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 1 below with included description and topic types required for the communication.

**Table 1: Description and topic types required for the 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 2: Goal Message Definition**

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

**Table 3: Result Message Definition**

| TaskResult | Description | Type |
|---|---|---|
| **result** | Holds the Goal's result | std_msgs/String |

**Table 4: Feedback Message Definition**

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

Below are XML files used during the workshop for executing a simple scenario using Station Controller & Skill Engine. The first XML contains the program description required by the Station Controller for executing the correct sequence whereas the last 2 XML files are the individual Process files required by the Skill Engine that describe the Navigation Skill and the Drilling skill respectively.

```
<?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 5: Station Controller Program XML File**

```
<?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 6: Skill Engine – Navigation 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 7: Drilling Process, XML File**

# 4. AERONAUTICS USE CASE REQUIRED SKILL IMPLEMENTATION

## 4.1. Description of the involved skills for drilling operation

In Table 5 the updated description of the required skills for the drilling operation can be seen.

**Table 5: Updated description of the required skills for drilling 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<br>- Operating range |
| 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<br>- Required end-effector |
| AprilTag localization | The MRP start searching and localizing AprilTags using a sensor. Obtains a estimation of the pose of the tag. | - Tag (is necessary to provide tag id or type?)<br>- Sensor<br>- 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)<br>- Sensor (MAC address)<br>- Theoretical position (searching area) |
| 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)<br>- Sensor (MAC address)<br>- Result of the element localization |
| Drilling | The MRP drills the localized template. This operation implies: | - Detected hole pose<br>- Specific strategy (if |

| Skill | Description | Key parameters |
|---|---|---|
| | 1. Correct localization and drilling detection<br>2. Appropriate end-effector for the required template<br>3. Collision free trajectories of the arms to the tool pre-inserting position<br>4. Tool inserting in the template's hole<br>5. Drilling-machine activation<br>6. Tool extraction from the template<br>7. Repeat 3-6 until completes all template holes. | necessary)<br>- Required end-effectors (can be inferred from the template model)<br>- MRP arm |

## 4.2. Implementation of the involved skills for drilling operation

In the current phase of the project the following skills have been implemented:

- Cell to cell navigation (navigation)
- In cell navigation (precise navigation)
- Template detection
- Hole detection
- Drilling

### 4.2.1. Cell to cell navigation

The MRP navigates within the areas. This navigation implies obstacle detection and avoidance. The skill receives a target area as a goal, the rest of configuration is tuned by navigation experts (Figure 8).

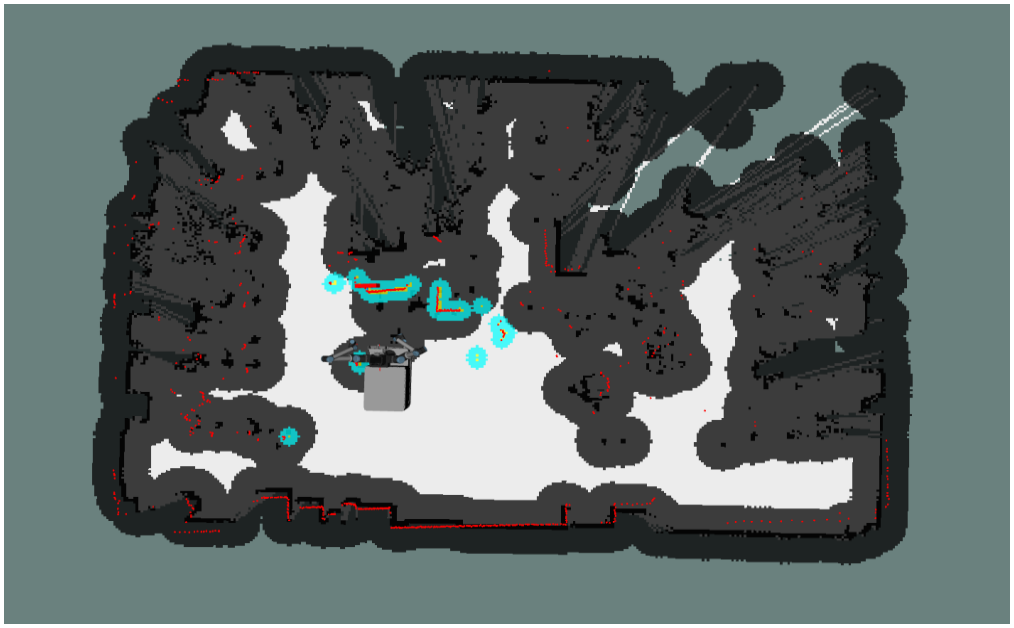

**Figure 8: Navigation skill allow MRP navigates providing a target area**

### 4.2.2. 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. In the aeronautic use-case the MRP is docked into a compressed-air station (Figure 9).

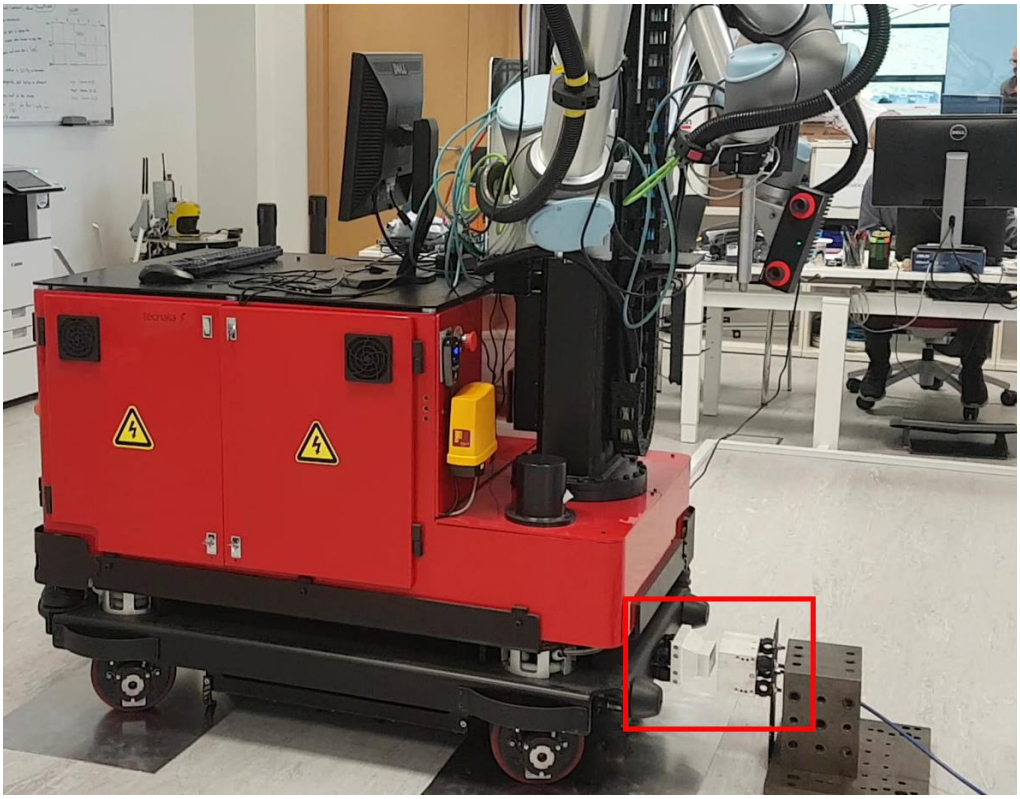

**Figure 9: Precise navigation allows the MRP docking into a compressed-air station**

### 4.2.3. 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 10). For more details please refer to D3.3.

### 4.2.4. Hole 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 (Figure 11). For more details please refer to D3.3.

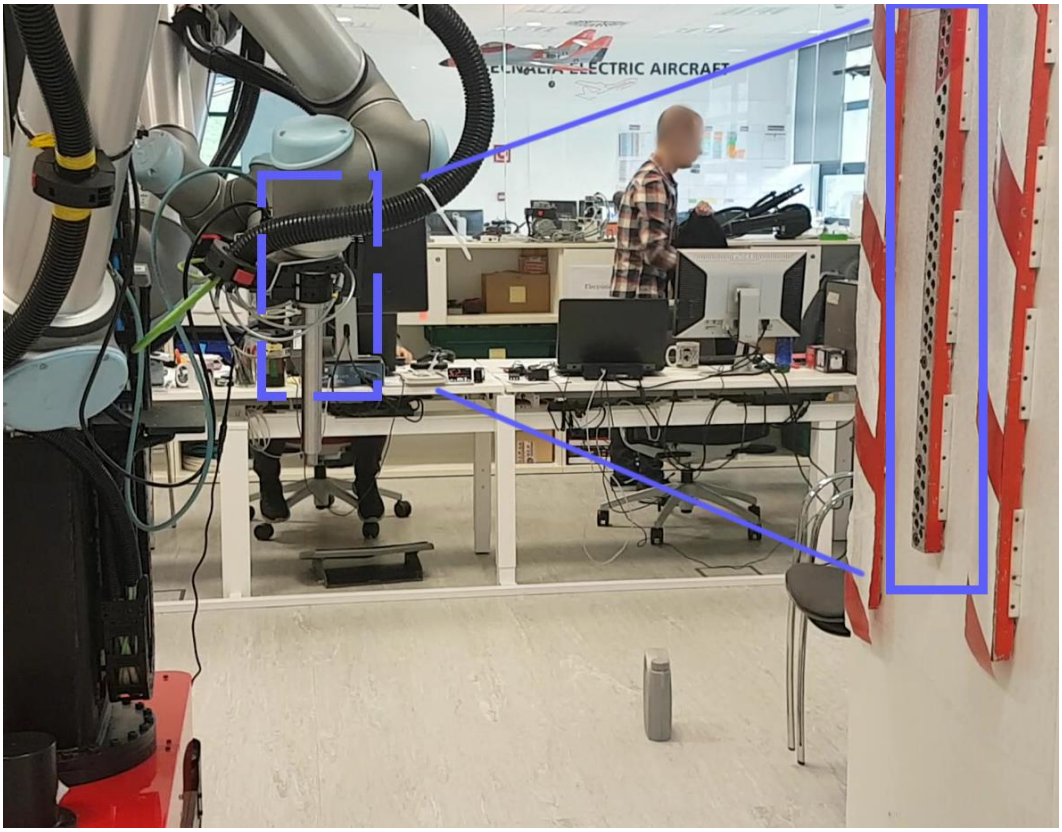

**Figure 10: First template pose estimation**

**Figure 11: Precise detection of the template holes**

### 4.2.5. Drilling

The MRP drills the localized template holes (Figure 12). 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.

**Figure 12: Drilling simulation with a mock-up that simulates the ADU**

# 5. AUTOMOTIVE USE CASE SKILL IMPLEMENTATION

## 5.1. Description of the involved skills for front axle damper assembly

In Table 6 the updated description of the required skills for the front axle damper assembly operation can be seen.

**Table 6: 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<br>- 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<br>- AGV path<br>- AGV speed<br>- 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<br>- Required end-effector |
| AprilTag localization | The MRP start searching and localizing AprilTags using a sensor. Obtains a estimation of the pose of the tag. | - Tag (is necessary to provide tag id or type?)<br>- Sensor<br>- Theoretical position (searching area) |
| Element localization | One of the arms of MRP (or using a static sensor) moves to a theoretical position for localizing the required element using ROBOCEPTION rc_visard 160 sensor. | - Element (drilling template)<br>- Sensor (MAC address)<br>- 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:<br>1. Collision free trajectories for approximating to theoretical position.<br>2. Element localization skill<br>3. MRP arm movement to element position<br>4. Gripper activation<br>5. Return to approximate position | - MRP arm<br>- Required gripper (inferred from the element)<br>- Theoretical position<br>- Localization skill |
| Pick and place | The MRP picks the required element with the specified arm. This operation implies the following:<br>1. Pick element skill<br>2. Collision free trajectories to target position approximate position<br>3. Place the object in target position<br>4. Target place localization<br>5. Gripper opening.<br>6. Return to a safe position<br>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<br>- MRP arm<br>- Required gripper (inferred from the element)<br>- Theoretical position<br>- Localization skill<br>- 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:<br>1. Collision free trajectories to target position approximate position<br>2. Target place localization<br>3. Assembly the object in target position<br>4. Gripper opening.<br>5. Return to a safe position<br>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<br>- MRP arm<br>- Required gripper (inferred from the element)<br>- Theoretical position<br>- Assembly strategy<br>- Localization skill<br>- Target position |
| Screw | The MRP screws required elements using a specific screwdriver. This operation implies the following:<br>1. Collision free trajectories to element position approximate position | - Element<br>- MRP arm<br>- Required end-effector<br>- Theoretical position |

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

## 5.2. Implementation of the involved skills for front axle damper assembly

In 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 and in cell navigation.

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.

# 6. CONCLUSIONS

This document presents the current status skill-based programming prototypes for industrial robotic applications. Moreover, how the Skill Engine interacts with the Station Controller and the rest of the framework is presented. The communication interface that is going to be implemented in THOMAS project is being described in details and XML files represent its results are being presented. Other topics covered in the document are the skill library, parametrization and interaction. The results from the skill parametrization process and their future use are being described. Finally, the set of required skills for each pilot case is presented and a description of implemented skills for the Aeronautics use case can be found. In following phases of the project, the involved skills for the automotive pilot case will be finalized and presented also.

# 7. 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 |