Traceable Robotic Handling of Sterile Medical Products

D6.1 ROS Middleware Mock-Up

Deliverable 6.1

Deliverable Title	D6.1 ROS Middleware Mock-Up
Deliverable Lead:	ASTECH PROJECTS LIMITED
Related Work Package:	WP6: Integration & Evaluation
Related Task(s):	T1.1 Use Case Specification T1.2: System Architecture T6.1: Demonstration Hardware Integration T6.2: Software Integration
Author(s):	Syafiq Rosidi (AST) Ben Gordon (AST)
Dissemination Level:	Public
Due Submission Date:	26/11/2021
Actual Submission:	29/11/2021
Project Number	101017089
Instrument:	Research and innovation action
Start Date of Project:	01.01.2021
Duration:	51 months
Abstract	ROS middle-ware mock-up, demonstrating UR arms control using ROS simulated environment.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089), 2021-2025

Versioning and Contribution History

Version	Date	Modified by	Modification reason
v.01	10.11.2021	Syafiq Rosidi (AST)	First Issue
v.02	11.11.2021	Anthony Remazeilles (TECN)	Technical Additions
v.03	12.11.2021	Syafiq Rosidi (AST)	Ready for internal Review
v.04	16.11.2021	Emily Dixon (AST)	Proof Checking
v.05	16.11.2021	Patrick Mania (UOB)	Internal Review
v.06	17.11.2021	Jean-Baptiste Weibel (TUW)	Internal Review
v.07	26.11.2021	Syafiq Rosidi, Ben Gordon (AST)	Amendments based off Internal Review Comments
v.08	26.11.2021	Emily Dixon (AST)	Final Proof Checking
V1.0	29.11.2021	Syafiq Rosidi (AST)	Final Version Ready for submission

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Table of Contents

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

1 Executive Summary

The Key Objective of this deliverable was to demonstrate initial functionality of the ROS framework to control elements of the system to be developed over the course of the TraceBOT project. The methodology intended to achieve this involved the development of system architecture (linked into Task 1.2) to identify the different elements and interfaces within the system; the creation of a CAD Mock-Up of the lab environment and; finally, the development of the ROS interface for programmable elements to be integrated. We present results demonstrating the control of the selected UR arms and beyond that, an attempt of the system being able to implement the Simplified Use Case as defined in Task 1.1.

This deliverable demonstrated successful functionality of control of simulated elements using the ROS middleware development – the next steps from here will be to substitute the simulated environment with ROS drivers for the physical hardware.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

2 Introduction

The TraceBOT project has been structured to allow work to develop in parallel across a number of partners. Parallel development can open up a risk to the project as there can be issues interfacing between the respective systems that may only be identified once the different building blocks are integrated together – this tends to be towards the testing phase at the end of the project, when timelines are tight.

In order to minimise this interfacing risk, the project took two decisions. The first was to utilise a ROS middleware framework. The primary advantage of using ROS is its ease of integration – partners within the TraceBOT project will be able to carry out focused development on their respective work packages and the ROS framework will allow these packages to easily interface together.

The second decision was to begin an initial integration effort which drove discussions on the interface and connectivity of the different components. At this early stage of the project, we intended to connect component mock-ups, which would then be progressively replaced by real implementations. Nevertheless, these discussions will reduce the risk of incompatibilities between developments in the next year of the project.

Leading up to this deliverable, significant work has been completed across all the different Work Packages. Of particular note to this report are the links to Work Package 1 [WP1] – in particular around work on Tasks 1.1 (Use Case Specification) and 1.2 (System Architecture). As a group, we have developed and agreed upon a 'Simplified Use Case' [Figure 1] as a reduced set of tasks to test initial control of the system. The scope of this initial testing includes control of simulated UR arms; communications between a simulation of the testing environment and the Digital Twin; and an attempt of a successful cycle of the defined Simplified Use Case.

The Simplified Use Case presented in Figure 1 concerns the insertion of the canister into the tray. This is composed of four main steps: detect the canister, grab the canister, move towards the tray, and finally insert the canister. We recognised the need to detail these steps to identify more atomic operations that would be required, and this would also allow us to highlight the expected necessary contribution of the WP/partners to its implementation. This is why the figure also presents a distribution of the steps per partner.

In this first analysis, we do not expect a completely functional system to be set up at this stage, but rather setup the interfaces of them, to ensure successful interaction and a common understanding of the overall architecture. The simulation tools available in ROS are used to provide a visualization of the system actions but the resulting simulation of physical interactions (in particular for grasping objects) may not be sufficiently mature yet therefore the use of "workaround" packages (such as that used for the gripper) in order to simulate real-world interactions.

As required, we have simplified the emulation of some operations with our intention being focused on the overall architecture. The simplified use case provides focus on development of some initial tasks. This development and learning can then be extended to other tasks on the complete use case.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

Finally, details of both the hardware specification and software specification of the system will be discussed to provide an overview of all elements within the system. Having all the elements mapped out within an interconnected system will help to identify potential risks and challenges around interfacing and provide the opportunity to develop proactive solutions to overcome these.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

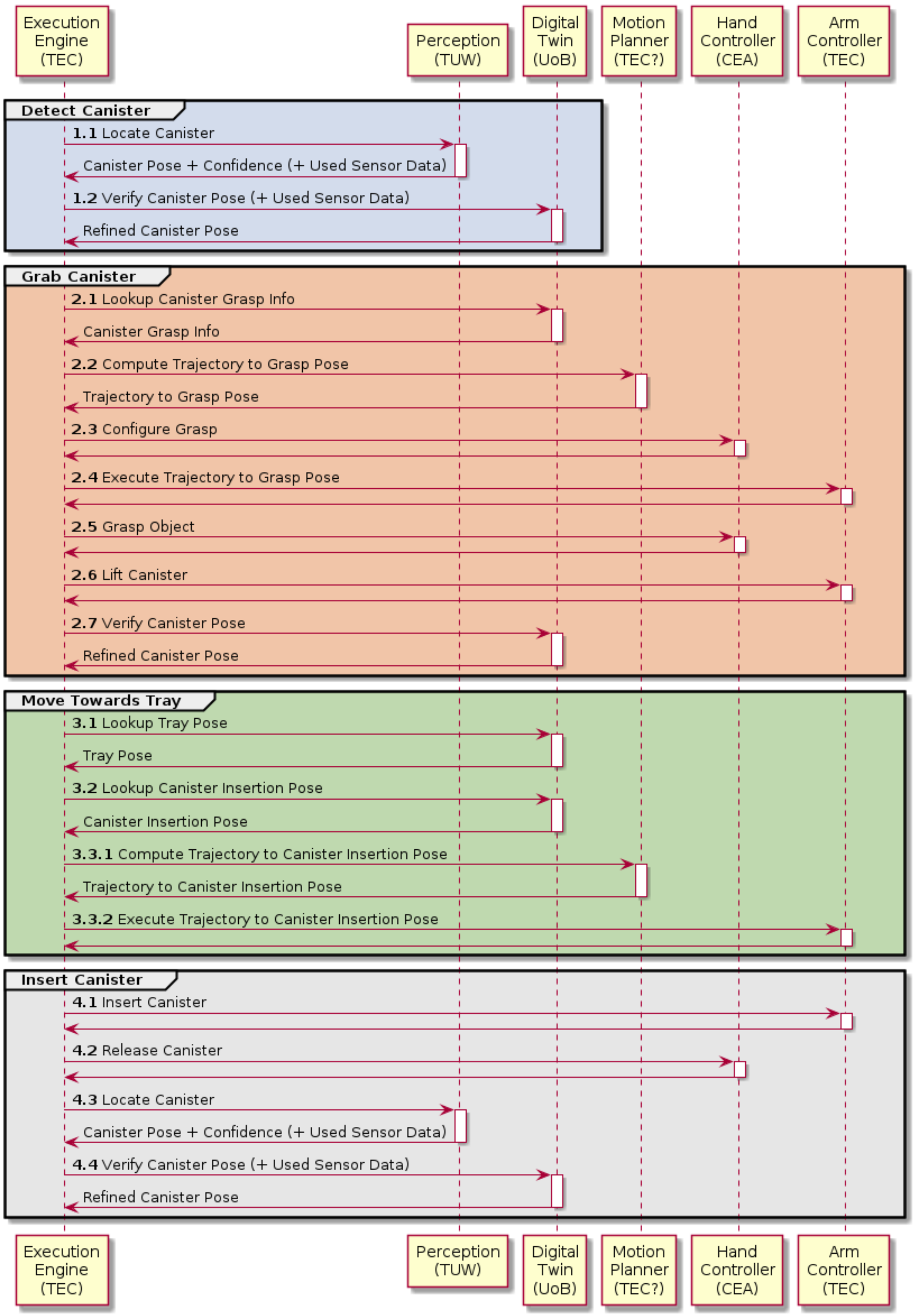


Figure 1: Graphical Representation of the Simplified Use and the respective responsibilities for each Partner

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

3 Description of Work & Main Achievements

Over the past 11 months, the focus of the work described within this deliverable has been focused on creating the fundamental building blocks required to integrate the respective TraceBOT systems from Work Packages 2-5. With Milestone 1 being focused around integration with the Digital Twin, the key focus was on building a ROS framework to allow all the systems to interface.

3.1 System Architecture

The development of software architecture provides a structural approach to understanding all software elements and their interactions within a given system. This will ensure that all required elements for the system are considered and that provisions are made for anything that has been identified as missing during the generation of this software architecture.

The approach for software development is to identified the key hardware involved, as the software will be used to control or receive inputs from this hardware. The core hardware involved in the system is listed below:

- UR Arms
- Gripper
- Camera
- Linux Controller
- Dedicated Digital Twin & Visual Processing Unit

Interfacing of the UR Arms is greatly helped by the provision of proprietary Universal Robots software compatible with use in a ROS environment. These arms will be controlled via the programmable logic developed as part of Work Package 3. The programmable logic will give the UR arms the information required for path optimisation for each of the tasks to be executed. Considerations for collision avoidance will also be required.

The Gripper will act as both an actuating manipulator and to provide tactile feedback to the system. The final gripper hand to be developed by CEA will not be delivered until at least after Month 25 of the project (after Milestone 2 Initial Integration) and so an interim gripper is required for initial testing. We have had discussions to consider interim grippers (e.g. Robotiq 2F85) but the general interim gripper concept would expect to have only 2 signal controls (open & close). Discussions on the specific control on the bespoke gripper hand have been deferred to later in the project once it is more defined.

The Linux Controller will manage all of the skills & actions of the system, as well as hosting all of the drivers for the hardware, and provide a visual interface to allow operator interaction with the system. A secondary system in conjunction, equipped with a GPU, will provide higher processing power for the Digital Twin and Visual Processing.

The final piece of major system hardware is the camera used. This has been agreed to be an Intel RealSense D435 and will be used as a vision sensor, with the input to be matched against the Digital Twin for verification and traceability purposes.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

All other hardware in the system will be either used to build and emulate the lab environment (e.g. worktop & frame) or to comply with safety requirements for use with a collaborative robot (e.g. safety zoning scanners/light curtains).

The collection of control and perception items involved in TraceBOT is significant, and it is therefore complex to provide an exhaustive and useful description of all of them. Nevertheless we rely on the implementation choice in TraceBOT that is to use a skill-based framework, in which each behaviour is encapsulated into a skill (behaviour involving robot motions, perception, gripper actuation, tactile sensing, etc…). The skill framework provides a programmer an intuitive structure via a standardised description of each behaviour. In this framework, the following core principles are considered:

- Each behaviour is represented as a skill
- A skill may itself rely on a set of skills
- Skill gets combined into a process.

In the use case presented on Figure 1, the complete process is defined as the combination of 4 skills, which are themselves composed of subprocess containing various skills. For now, the hierarchical structure is of level 2, but the framework is scalable. At execution time, the Execution engine is responsible for launching each of the required skills according to the plan.

The integration effort consists then to implement each behaviour as skills. Taking advantage of the ROS functionalities, the Skill framework is provided with the possibility to implement "atomic skills" as ROS actions. As a starting point, it has been thus decided to implement each of the leaf skills (that are distributed per partner) as ROS actions, following the specificities of the skill framework.

At the integration level, the advantage of using a framework is that it encapsulates all behaviours at the highest control level and provides a common communication interface. Then the implementation of each component may require more complex structure, but that is the responsibility of each partner provider. For instance, the implementation of the object detection functionality requires a ROS camera driver, a visual perception component to launch required image processing techniques on an image received from the camera. But at the integration level it is represented as a detect object skill which provides the estimated pose of the required object. The complexity of the implementation is hidden from the execution engine that only handles the management of skill launch, monitoring, and parameter transmission.

Figure 2 presents a snapshot of the current architecture displaying few skills implemented as ROS actions (in purple), which are progressively triggered by the Execution engine according to the process description (in green). The skills are connected to the ROS components they require to perform their duty. For instance, the Verify Canister internally interacts with the Digital Twin to perform that operation. As an aside, the tracer is used to collect operational details to be converted into the audit trail, which gets progressively filled. To get a systematic tracing, a derivation of the skill concept has been prepared to get "Traceable Skills", in which the input, feedback (at a given frequency) and outputs (actions result) gets automatically collected, packed and transmitted to the Tracer through regular topic communication. The implementation currently focuses on skills implemented as ROS actions, but it will be progressively extended to consider other skill types, or more detailed trace information.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

On an integration perspective, all skills involved within the simplified use case have been defined as an action interface, and are available in the TraceBOT shared code repository (Annex 1). All skills have been implemented where possible. Where full implementation is not yet possible (i.e. lack of functionality) mock-ups have been setup to allow so that the complete process can be already launched (Video 1).

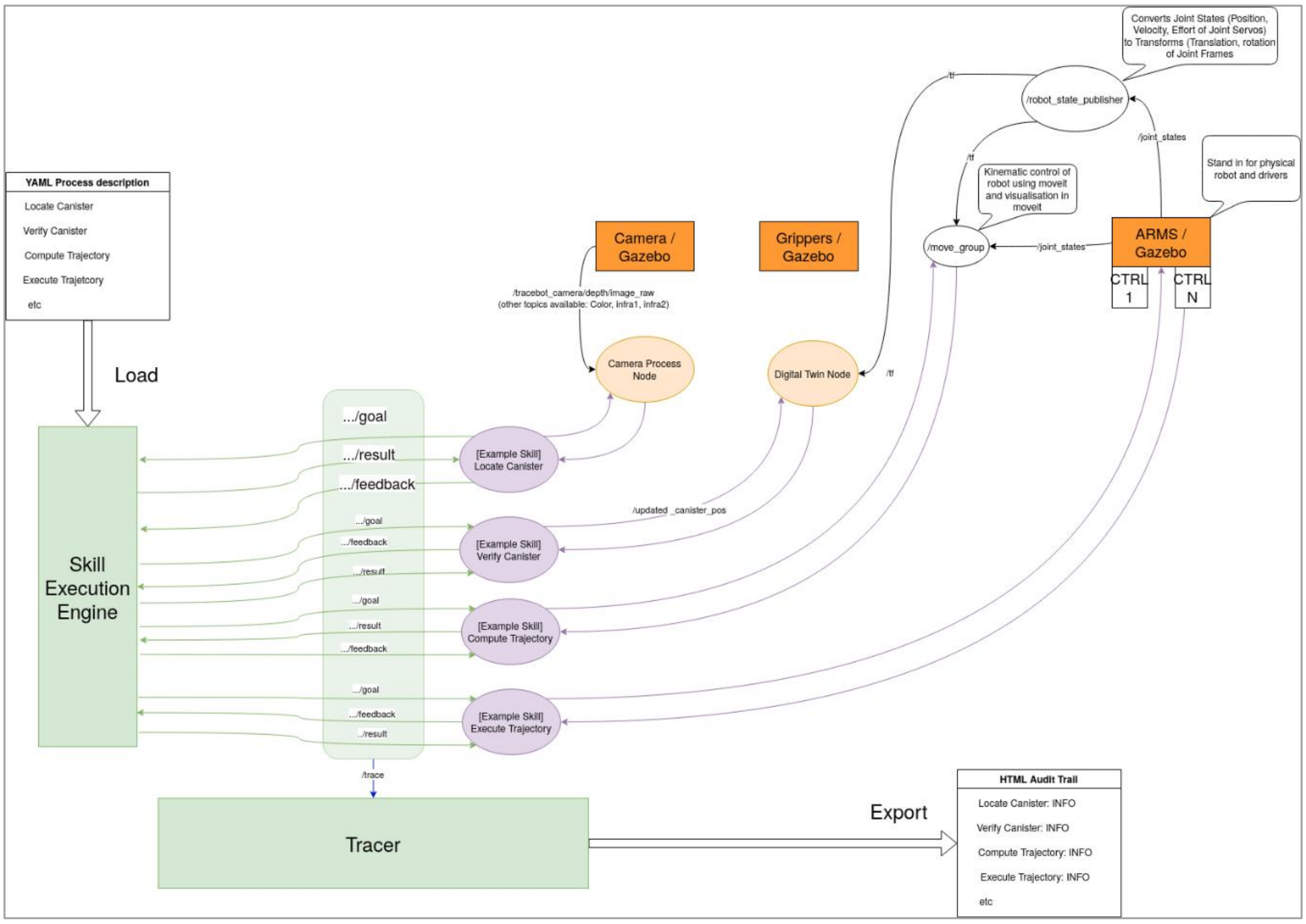


Figure 2: Software architecture developed using the online in-Browser Flowchart creator draw.io. This version of the architecture shows some example skills for a given TraceBOT process and how the process ultimately connects to the Camera Nodes, digital twin

Horizon 2020
TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

3.2 CAD Mock-Up: Mechanical Concept Design

In order to provide an initial visualisation of the TraceBOT setup, a CAD Mock-Up was developed to include a basic overall model, including the robot arms, example gripper units and some of the apparatus to be manipulated. Images of these can be seen in Figure 3. The setup is initially quite basic with both arms mounted on a block at 90° orientation to the table surface. The camera is setup looking directly on the table surface.

There are a few limitations that were identified from this initial mock-up. The first of these relates to the specification of the UR5e robot arms. CEA identified that the likely weight of their bespoke gripper hand would approach 4 to 5kg[1] - the maximum payload that a UR5e is designed to manipulate. In addition to this, the gripper would be expected to manipulate an object weighing approximately 1 kg. Thus, it was decided that a robot with a larger payload capability would be required.

Both UR10e and UR16e models were considered, which both have a bigger payload (10kg and 16kg respectively) as well as a larger reach. Figure 4 provides an illustrative comparison between the UR5e and UR10e arms. It is clearly visible that the reach, size and construction of the UR10e arm are significantly larger, to provide required capability of handling a larger payload. Figure 5 provides a comparison between the UR10e and UR16e robotic arms with the UR16e demonstrating an even sturdier construction than the UR10e. The larger size of both the UR10e and UR16e may provide some loss of flexibility in manipulation actions, however, these do not appear to be significant.

In addition to the greater payload capability, the greater robotic arm size provides a greater reach. This comparison is illustrated in Figure 6, with the UR5e arm showing limitations in potential reach, whilst the UR10e appears to demonstrate accessibility to all areas of the workstation. Thus, the consortium decided that the UR10e arm was much more suitable to the TraceBOT application in comparison to the UR5e arm and that the project should progress under the assumption of UR10e arms being used. The UR16e arms were not considered any further since the UR10e was deemed sufficient for the applications and would cost less than the UR16e.

A final consideration on the design of the workstation was based on the orientation of the robotic arms. It was identified that the initial design concept with the robot arms at a 90° could present a potential risk of singularities in programming the system. Thus, a layout of the system with the arms set at 45° was drafted, as seen in Figure 7. This is to reduce the risk of singularities in programming robotic motions. The robot arm mount is also made independent from the workstation, providing better structural support.

[1] Note that the weight of the envisioned gripper is of similar weight to comparable dexterous hand. For instance, the Shadow robot hand weights 4.2 kg.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

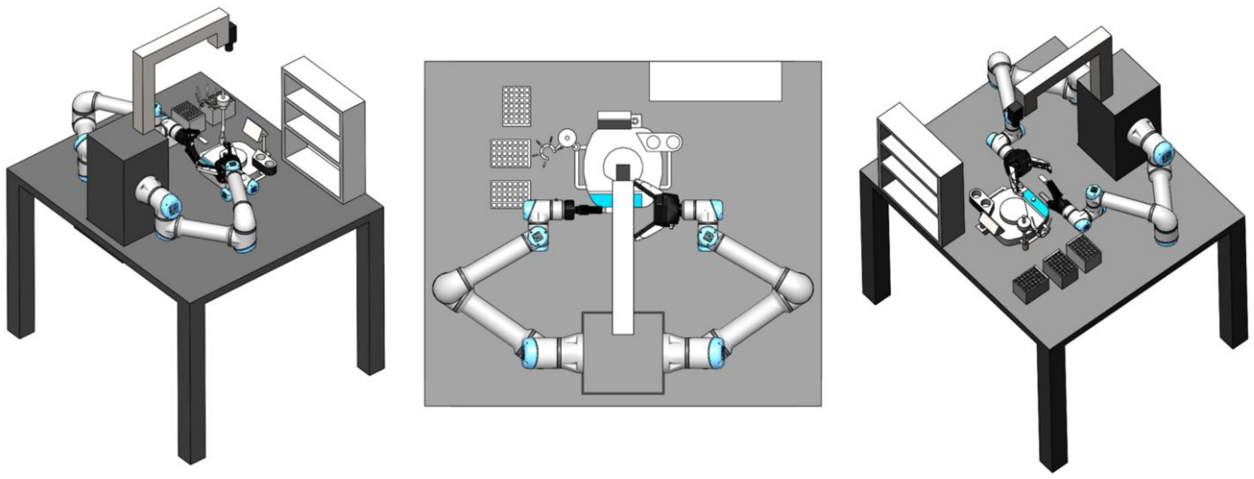


Figure 3: Initial CAD Visualisations of the TraceBOT workstation

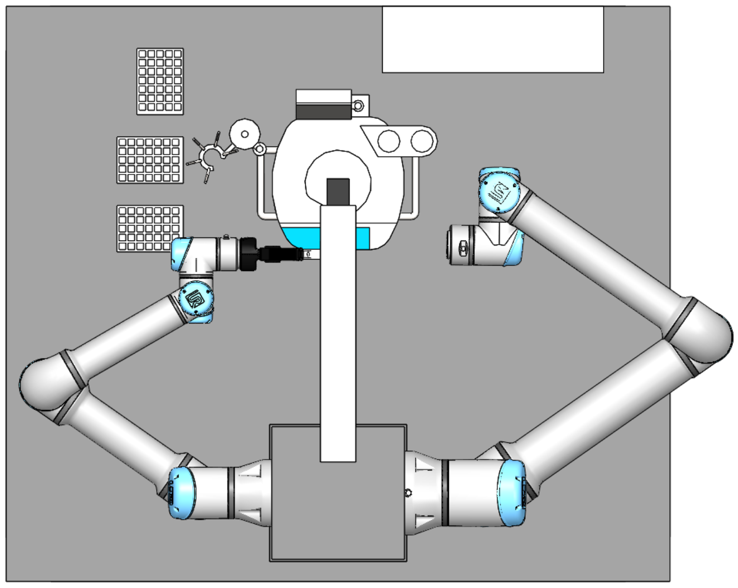


Figure 4: Visual Comparison between UR5e (left) and UR10e (right) Robot Arms

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

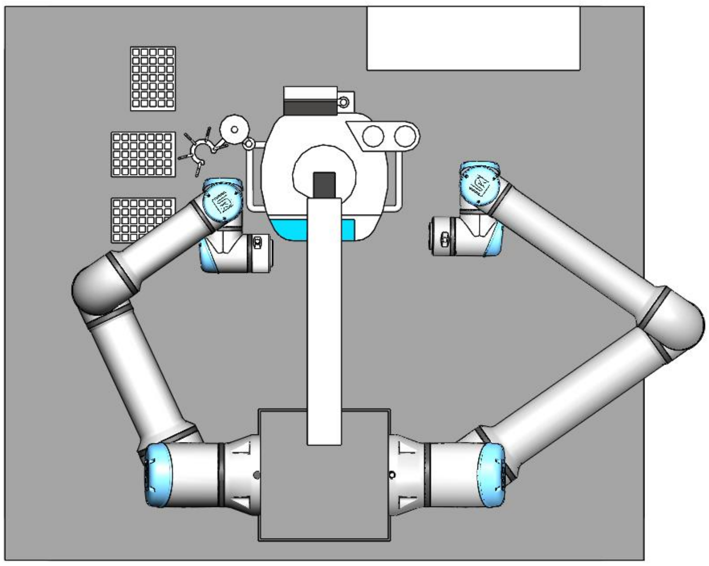


Figure 5: Visual Comparison between UR16e (left) and UR10e (right) Robot Arms

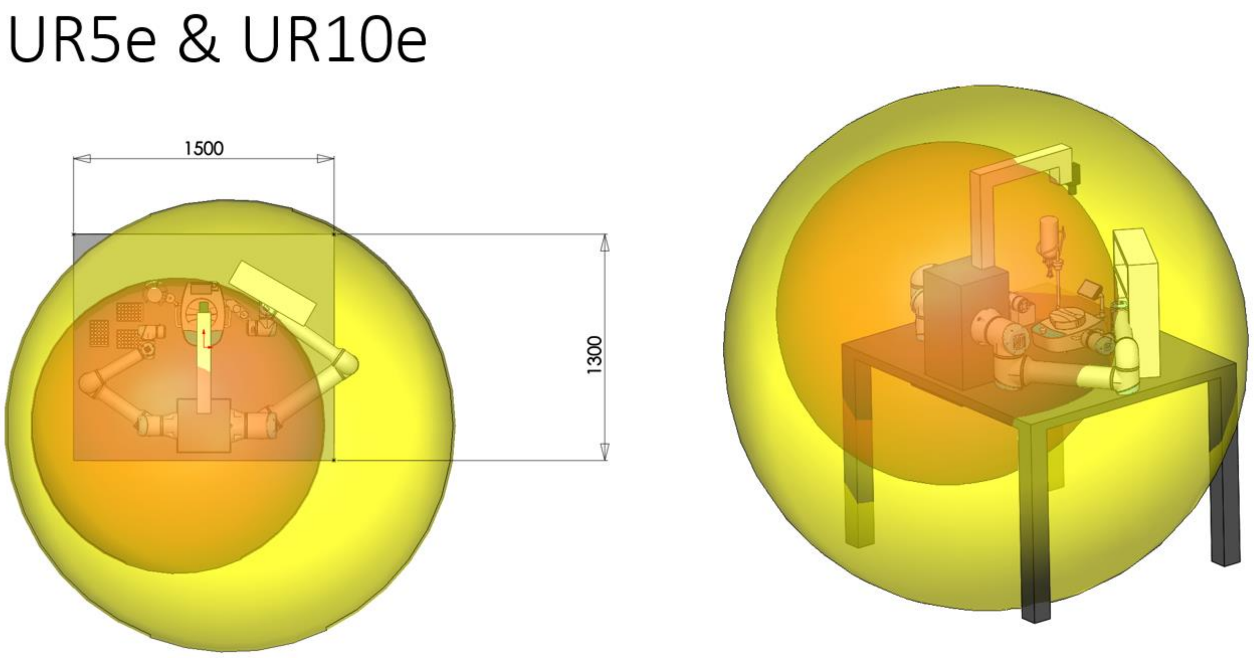


Figure 6: Reach Comparison between UR5e Robot Arm (red sphere) and UR10e Robot Arm (yellow sphere)

Horizon 2020
TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

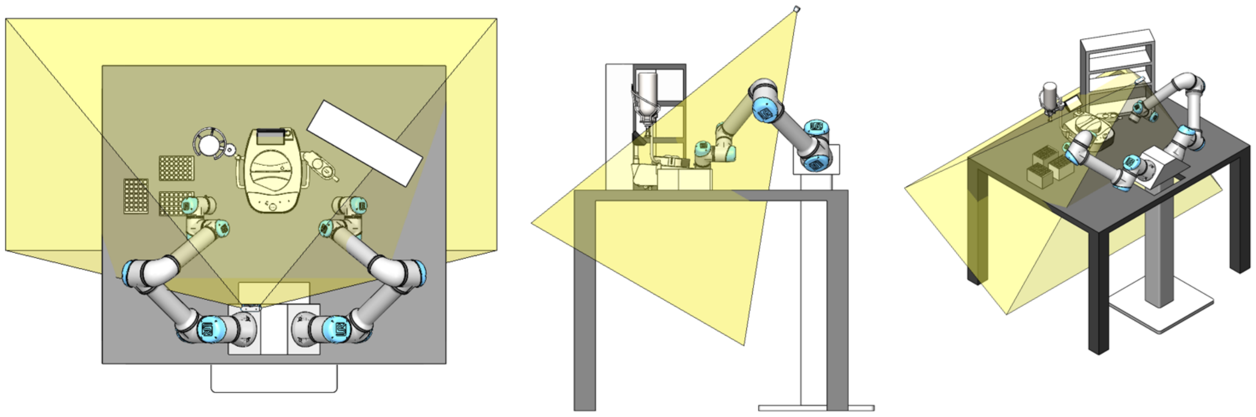


Figure 7: Illustration of UR arms mounted on an independent mount oriented at an angle of 45° to the workstation surface. The camera location has been adjusted to provide a better field of view for using a single camera. Later discussions will include detailed investigation optimising visual perception strategies.

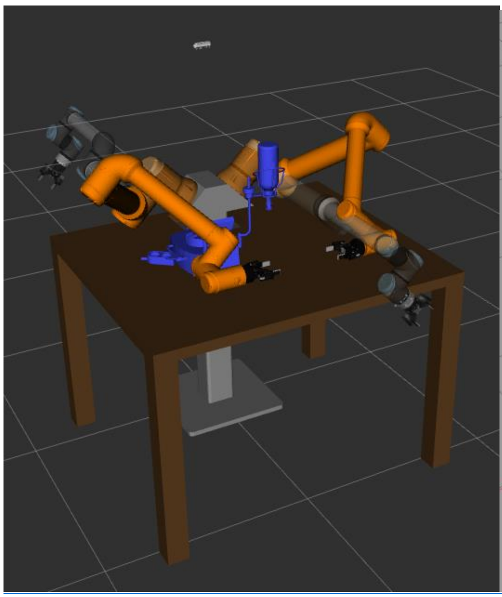
3.3 Software Integration

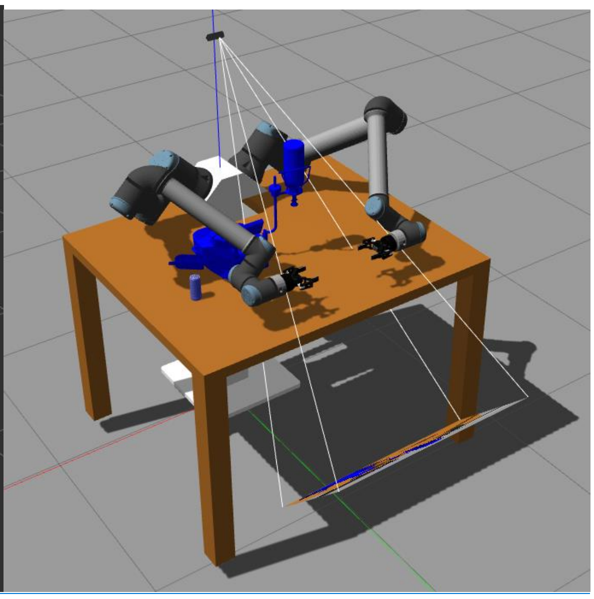
3.3.1 ROS Framework

The initial TraceBOT Description package was created by Tecnalia, then developed upon by Astech. A gazebo simulation and MoveIt configuration (for point-to-point robotic arm motion planning) were added, as well as replacing the abstract shapes with accurate models of the arms, camera, gripper and environment (such as pump, table, stand and canister).

Figure 8 TraceBOT in RViz/MoveIt

Figure 9 TraceBOT in Gazebo





Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

The Description contains all of the models required to bring the robot together, such as the stand for the arms, and other environment models such as the pump, table and canisters. A URDF file brings these components together to form a working model of the TraceBOT system in RViz (Figure 8). The MoveIt Config package is used for the path planning of the arms and grippers. These can be interacted with within RViz (Figure 8), or by other ROS packages. Until we begin testing with physical hardware, we also have the Gazebo package (within TraceBOT mock-up), in order to simulate the robot (Figure 9). In the simulated environment the robot can pick up and move objects within the scene (Video 2). This allows the object to be fixed to the gripper when the fingers make contact. Currently to avoid issues the collision gets disabled when the object is picked up, then re-enabled on release to prevent physics issues in Gazebo.

The description and gazebo packages make use of other 3rd party repositories for the UR Arms, RealSense Camera and Robotiq Gripper. These provide the models and URDF files for RViz and gazebo, as well as the drivers for the physical hardware.

3.3.2 Digital Twin (UoB)

The TraceBOT Digital Twin contains a visualisation of the robot and environment within unreal engine. This is used as a comparison for every action that the robot takes, allowing for errors or deviations between expectation and reality to be found.

A TraceBOT Digital Twin interface is used to allow the Digital Twin to communicate with other TraceBOT Packages, in particular for enabling all verification processes, or to fill the DT with sensor input. The DT is also connected to the robotic system joint information, so that the DT representation of the world is automatically updated when the robotic arms move (even if this is done through the Gazebo emulation).

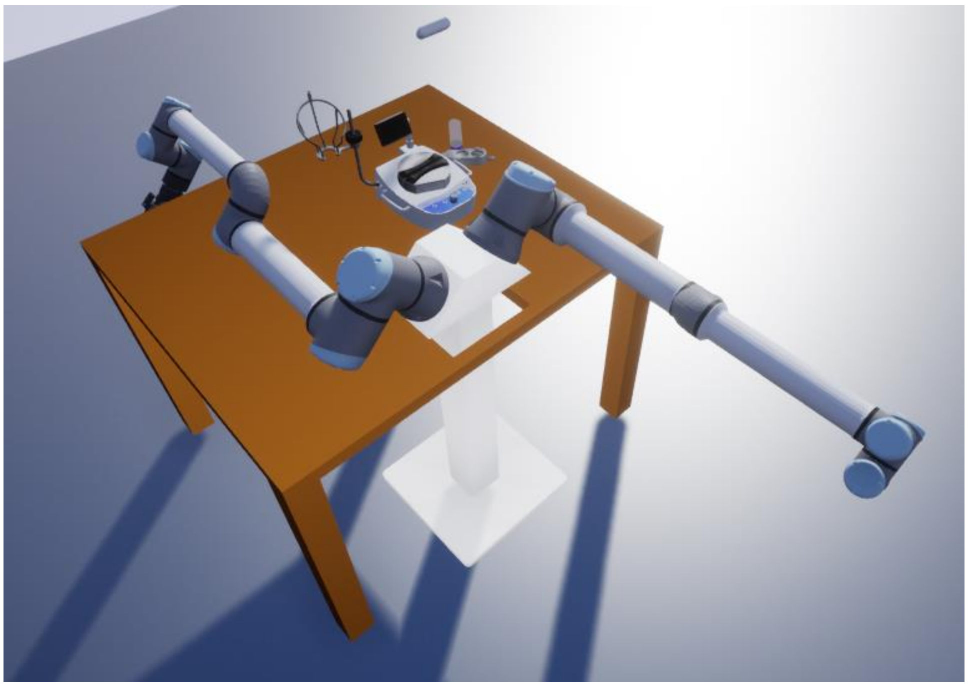


Figure 10: TraceBOT Digital Twin

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

3.3.3 Further Development

All of the TraceBOT packages are contained within a structured git project (Annex A), allowing everyone in the consortium to view, add to, or change any of the available packages.

3.3.3.1 TraceBOT Tracer (TUW)

The tracer component developed as part of the TraceBOT project is responsible for creating the final audit trail based on the process execution. It therefore needs to keep track of every action the robot is taking and the data justifying this action, but also the data proving the accurate execution of any given action.

To achieve this, the tracer relies on Traceable Skills (every input, output and feedback of every skill is kept track of automatically), and the idea that verification can be implemented as skills. Thanks to the flexibility of the skill definition, a given action and its corresponding verification can be combined as a Verified Action. The verification provides some level of guarantee that the action was successful and the data gathered both from the action and the verification step allow a better understanding of the process execution.

To keep track of all the relevant information, the tracer is directly connected to both the skill execution engine and the Digital Twin, which can represent the robot knowledge at a given time through the concept of Narrative-Enabled Episodic Memories (NEEMs).

3.3.3.2 TraceBOT Msgs (Tecnalia)

The Msgs repository contains all of the custom ROS interface definitions, such as messages, services and actions for the TraceBOT project.

3.3.3.3 TraceBOT Processes (Tecnalia)

Process contains a sequence of skills grouped into a smach state machine. This allows for control over all of the sub-sub-tasks (skills/actions), and groups them together into sub-tasks and tasks in a similar structure as shown in the simplified use case (Figure 1).

Each skill contains ROS Action servers with a defined interface in order to share information with other ROS nodes through Goal, Feedback and Result messages. The TraceBOT process execution manager guarantees that the skills are executed in the right order and connect output and input of actions that depend on each other.

3.4 Link to Milestone MS1

The timing of this deliverable is aligned with the achievement of the first Milestone, MS1, named *Traceable Semantic Twin & Initial integration.*

Video 3 attempts to illustrate most of the architecture and integration items. But as all may not be straightforward to capture from the video, we comment here on the Milestone related to the software and integration effort. There are the following:

[WP1] Use Cases, Verification Plan & Initial Architecture Defined

The integration effort has started with the extensive description of the complete use case. To reduce complexity of the first integration, we narrow it down to the simplified use case presented in this document. This document also provides details on the architecture choices we made.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

[WP4] Traceability Framework Established in Simulation

A first version of the traceability structure or framework is proposed and implemented, as mentioned in this document. The notion of "traceable skill" is introduced, implemented as a wrapper of skills, enabling to automatically handle the transmission of information towards the tracer component. On the tracer end, everything is kept track of using IDs attached to every message. This helps distinguishing between messages even if they arrive out of order because of network bottleneck, and enable the execution of parallel skills while guaranteeing a correct structure in the final trail. The tracer collects all information and is then able to create a unique trail of information. The tracer also saves the information received regularly during execution guaranteeing that no data will be lost in case of a software issue, a situation in which a valid audit trail is of particular importance. A barebone "audit trail" maybe formatted in HTML, is currently prepared. We would like to then present such materials to the Advisory Board, to converge towards useful content and format. In further iterations, the tracer will be more tightly connected to the concept of Narrative-Enabled Episodic Memories (NEEMs) from the Digital Twin to enable a more complete representation of the robot knowledge during every skill execution.

[WP4] First Verification Task Implemented

In parallel with the traceability framework, work has been underway to implement a first verification task. As mentioned previously, verification tasks will also be implemented as skills and, therefore, benefits from all the work done in the process execution manager and the Traceable Skills. In line with the objectives of the simplified use-case, the first verification task is kept simple and lets us test the validity of our architecture. We chose to use the comparison between the expected canister pose (given by the Digital Twin) after the canister insertion in the tray and the detected canister pose as given by the 'LocateCanister' skill.

[WP5] Definition of the Conceptual, Reasoning Framework and Semantic Models

The main framework supporting the digital twin has been identified and prepared. We conceptualized a hybrid knowledge-based approach for the sterility testing use case which will combine symbolic representation techniques with reasoning methods supported by a game engine based Digital Twin. The underlying concept will be outlined in more detail in the related deliverable D5.1. We are also studying the possibility to use the reasoning models also within the intuitive programming interface.

[WP5] Fundamental Simulation Aspect are Available in the TST

We have derived the fundamental, models required for the Digital Twin to handle the relevant manipulation tasks in the use case. Based on that, we have derived the required architecture and modalities in the game engine system which have to be implemented to support DT-based verification actions in sterility testing scenarios.

Fundamental aspects to support that functionality have been developed and provided in the DT simulation environment. This includes, for example, controllable dual robot arms as well as objects and environmental features involved in the process (Video 3).

More details on the system are provided in the upcoming Deliverable D5.1

[WP6] ROS Middleware Mock-up (D6.1)

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

This deliverable effectively describes the ROS mock-up.

[WP6] Virtual PC Distribution for Framework Evaluation

We decided to consider as much as possible standard installation and deployment tools as provided by ROS (roslaunch, rosinstall, wstool, …). On the main code repositories, we incorporated some CI tools to automatically verify the compilation and installation on fresh environment using containers and industrial_ci. The DT is for now deployed as a standalone executable.

[WP6] Initial Specification of Hardware & Software Architecture

This item is covered in the previous sections of this document.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

4 Deviations from the workplan

There are 2 main deviations from the workplan mentioned in this deliverable that are worth discussion in this section:

1. **Payload Limitations of UR5e Arms –** Since the combined weight of the anticipated bespoke gripper hand and the expected payload to be manipulated would be greater than the designed capabilities of the UR5e arm, it was decided that the TraceBOT system would require an upgrade to UR10e arms. The expected payload is not anticipated to approach anywhere near the limits of the UR10e arms. These UR10e arms have since been acquired by Astech at a cost within the original anticipated budget for the UR5e arms.
2. **Virtual PC Distribution for Framework Evaluation –** We no longer required a virtual PC distribution as using git provides easy installation of packages as well as any dependencies they require. Allowing for fresh installs rather than a virtual workspace with everything pre-installed, also allows us to catch any dependencies that haven't been included, and therefore add them.
   We do still have tools in place such as CI to evaluate the framework systematically, and Tecnalia is generating docker images with the TraceBOT components installed which servers a similar function to the virtual PC.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

5 Conclusion

The original purpose of this deliverable was to demonstrate control of UR arms using a ROS middleware framework. We believe that this objective has been met and exceeded, with the benefit of providing a solid platform for future development. An initial system architecture has been defined that provides an illustration of the typical software elements and interfaces included in the system as well as a concise description of the major functional hardware that will provide sensing and control of the system. A CAD mock-up of the system has been designed that has provided the consortium with a visual aid to help make decisions on potential issues that the system may occur in the future tasks of the project. The specification of using a UR5e robotic arm has been upgraded to a UR10e robotic arm as a result of this as well as identification and solutions to potential issues such as accounting for singularities in programming the system. Finally, a simulated environment of the system has been developed in-situ of physical hardware system and drivers in order to develop programming of the system on the earlier defined 'simplified use-case'. A private repository of code has been developed on the open-source platform GitLab containing all elements developed to demonstrate this simplified use case. This will be used extensively as the project develops to include the physical hardware and the implementation of a ROS middleware which will allow for a simple transition from the simulated environment.

The result of this initial integration effort has provided the TraceBOT project with an element of confidence that we are progressing sufficiently for a hardware integration phase for Milestone 2. We will use these results to begin the full mechanical design process including detailing, procurement of major items and planning of assembly of an initial physical system. The development of the skills in the Simplified Use Case will be built upon further and this will eventually be extended to development of skills for a full use case specification as required for TraceBOT.

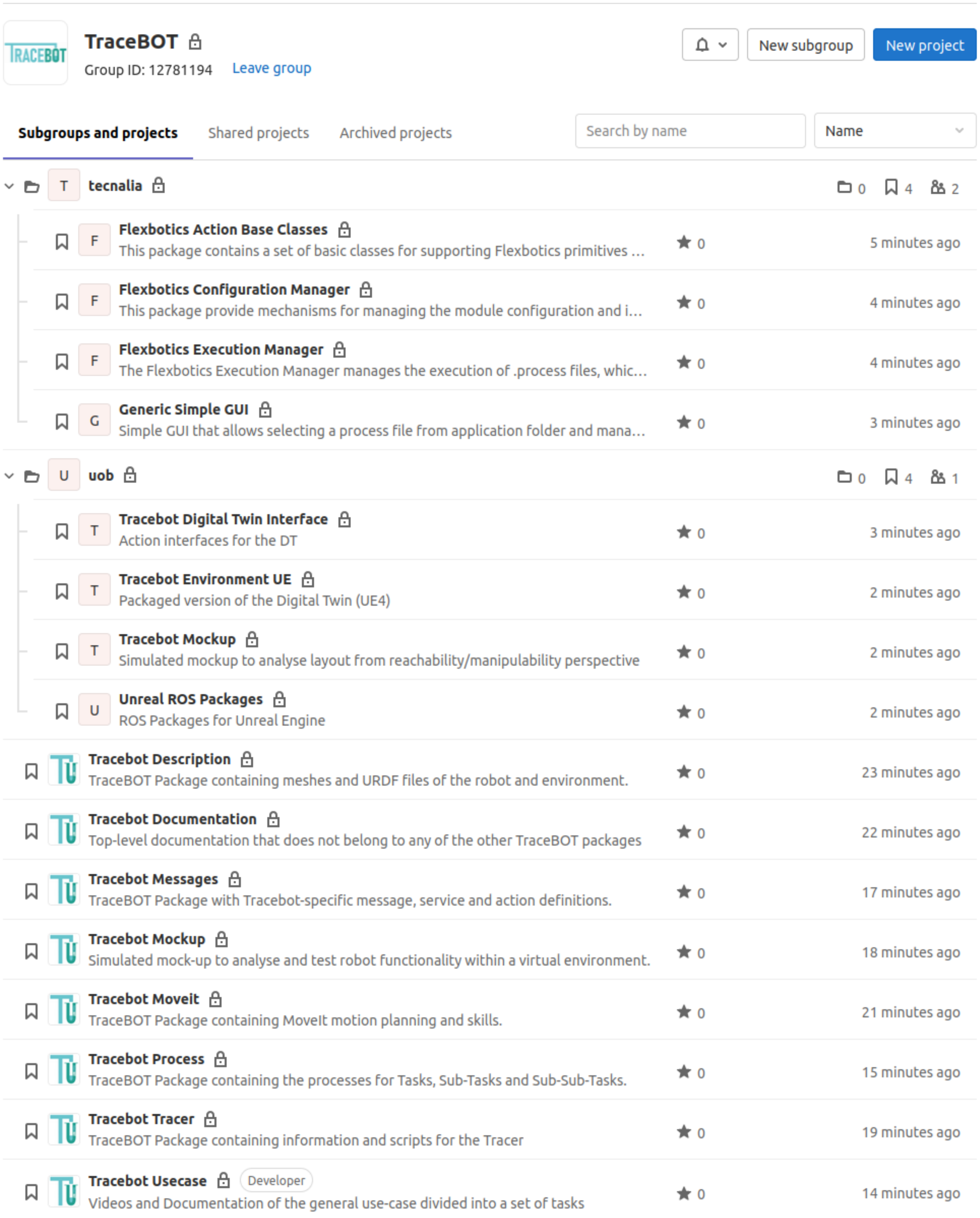
Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

6 Annexes

6.1 Annex A

Snapshot of the TraceBOT Gitlab area, showing all code developed in the project centralised in a single location.



TraceBOT
Group ID: 12781194 Leave group

New subgroup | New project

Subgroups and projects | Shared projects | Archived projects

Search by name | Name

Project	Description	Stars	Updated
tecnalia		0 subgroups, 4 projects, 2 members	
Flexbotics Action Base Classes	This package contains a set of basic classes for supporting Flexbotics primitives …	0	5 minutes ago
Flexbotics Configuration Manager	This package provide mechanisms for managing the module configuration and i…	0	4 minutes ago
Flexbotics Execution Manager	The Flexbotics Execution Manager manages the execution of .process files, whic…	0	4 minutes ago
Generic Simple GUI	Simple GUI that allows selecting a process file from application folder and mana…	0	3 minutes ago
uob		0 subgroups, 4 projects, 1 member	
Tracebot Digital Twin Interface	Action interfaces for the DT	0	3 minutes ago
Tracebot Environment UE	Packaged version of the Digital Twin (UE4)	0	2 minutes ago
Tracebot Mockup	Simulated mockup to analyse layout from reachability/manipulability perspective	0	2 minutes ago
Unreal ROS Packages	ROS Packages for Unreal Engine	0	2 minutes ago
Tracebot Description	TraceBOT Package containing meshes and URDF files of the robot and environment.	0	23 minutes ago
Tracebot Documentation	Top-level documentation that does not belong to any of the other TraceBOT packages	0	22 minutes ago
Tracebot Messages	TraceBOT Package with Tracebot-specific message, service and action definitions.	0	17 minutes ago
Tracebot Mockup	Simulated mock-up to analyse and test robot functionality within a virtual environment.	0	18 minutes ago
Tracebot Moveit	TraceBOT Package containing MoveIt motion planning and skills.	0	21 minutes ago
Tracebot Process	TraceBOT Package containing the processes for Tasks, Sub-Tasks and Sub-Sub-Tasks.	0	15 minutes ago
Tracebot Tracer	TraceBOT Package containing information and scripts for the Tracer	0	19 minutes ago
Tracebot Usecase (Developer)	Videos and Documentation of the general use-case divided into a set of tasks	0	14 minutes ago

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

6.2 Annex B

Table of videos

Link to video playlist: TraceBOT D6.1 - YouTube

Video 1 - Simplified Use-Case

Video 2 - Pickup of object in simulated environment

Video 3 - Digital Twin Functionality

N.B. The videos above are to provide an initial demonstration of some of the intended separate functionalities that have been developed and tested. Some irregularities may be presented but this is acceptable here as we are not looking to demonstrate a complete functionality at this stage.

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

6.3 Annex C

Table of figures

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)

7 References

No references required

Horizon 2020

TraceBOT receives funding from the European Union's H2020-EU.2.1.1. INDUSTRIAL LEADERSHIP programme (grant agreement No 101017089)