TRACEBOT

System Architecture

Deliverable 1.1

Deliverable Title	D1.1 System architecture
Deliverable Lead:	ASTECH PROJECTS LIMITED
Related Work Package:	WP1: Requirements management and verification preparation
Related Task(s):	T1.2: System Architecture T6.1: Demonstration hardware integration T6.2: Software integration
Author(s):	Ben Gordon (AST) Syafiq Rosidi (AST)
Dissemination Level:	Public
Due Submission Date:	28/02/2022
Actual Submission:	28/02/2022
Project Number	101017089
Instrument:	Research and innovation action
Start Date of Project:	01.01.2021
Duration:	51 months

Abstract	System architecture of both software and hardware. Also including scenarios with single, and dual robotic Discussion around risk evaluation in the project is included, with focus on appropriate tools (such as FMEA) and their applicability to this project.

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	18.01.2022	Ben Gordon (AST)	Initial template and draft
v.02	09.02.2022	Anthony Remazeilles (TECH)	Template update, and revision of the architecture section
v.03	11.02.2022	Syafiq Rosidi (AST) Patrick Mania (UOB) Miguel Sarasola (TECN) Julie Dumora (CEA) Mathieu Grossard (CEA)	Revision of Software Architecture sections – contributions from various partners and approval from Astech
v.04	15.02.2022	Syafiq Rosidi	Ready for internal revision
v.05	21.02.2022	Ben Gordon (AST) Miguel Sarasola (TECN) Anthony Remazeilles (TECN) Jean-Baptiste Weibel (TUW)	Minor revisions ready for final checks
v.06	24.02.2022	Mathieu Grossard (CEA)	Added software information for grasp control
V.1.0		Ben Gordon (AST)	Revised version read 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 setup a viable system architecture for use with the TraceBot system.

This involved creating a virtual environment for the robot to operate in, as well as modelling the robot and any useful objects for it to interact with (e.g. pump and canister) to represent the physical counterparts. This allowed for a pre-testing of the software framework, and allow testing of varying hardware components (such as varying arm lengths) before deciding on the final hardware framework.

The results presented demonstrate a functioning architecture, with single and dual arm control, and implementation (either fully or partially) of all aspects of the system architecture and the starting of implementation of the hardware framework.

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 purpose of the TraceBot project is to develop an automated solution for sterility testing where all actions are fully traceable. In order to achieve this, we need to create a system with multiple functional modules focused around manipulation of objects and in capturing and referencing this against a Digital Twin. The systems include a dual arm system interacting with a pump, an advanced vision-based system for monitoring the process, dexterous manipulators for articulating & moving lab apparatus and an advanced reasoning system that requires development of learning capabilities. Milestone 1 was focused on software integration and ensuring that software elements combined well together. Moving forward in preparation for Milestone 2, we will also be looking at integrating hardware. The system architecture describes the interface and connectivity between different components of the system.

There will be a large amount of challenges to overcome with the TraceBot project, therefore having a strong system architecture will be key to allow extensive testing of the system.

To achieve this, building up a virtual environment with ROS and Gazebo (as discussed in D6.1, see Figure 1) allows for early testing of the system, and allowed for us to begin building up the foundational software framework for the system.

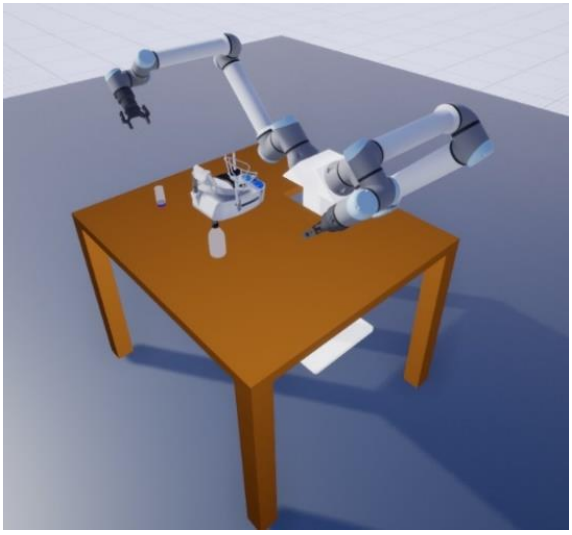

Figure 1 - Visualisation of the TraceBot system within the Digital Twin

Having a virtual environment allows us to work towards a functioning software framework, with consideration and virtual testing of the hardware components that will be using it. By simulating virtually, the risk of failure and damage of the physical hardware framework is significantly reduced.

The hardware specification and software specification of the system provides 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)

The system architecture will demonstrate both single and dual arm functionality within the simulated environment, with consideration to what hardware will be needed to achieve this functionality. Various tests have been carried out throughout the project to test the best hardware for the job, specifically around the arms and stand-in gripper. The stand-in gripper will be used in place of the CEA gripper until a functioning version is completed in a later milestone.

Risk mitigation strategies are considered within this deliverable in order to ensure all design elements are considered. We consider the use of tools such as Failure Modes & Effects Analysis and their appropriateness to this application.

Horizon 2020

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

3 System Architecture

The system architecture (Figure 2) has been built up over the past 14 months, planning the hardware framework, building this up in simulation to allow for the building and testing of the system architecture. This will form the foundation of the project and will allow for the further integration of the respective TraceBot systems from Work Packages 2-5.

It has to be mentioned that all the different components or nodes required to control the system are not necessarily described here. The integration policy is to provide a skill interface for any capability provided to the robot. The related skill can then either be implemented in a single action (purple bubbles in Figure 2), or rely on interaction with other nodes to complete its duty. The distributivity of the ROS framework enables to cope with both approaches seamlessly.

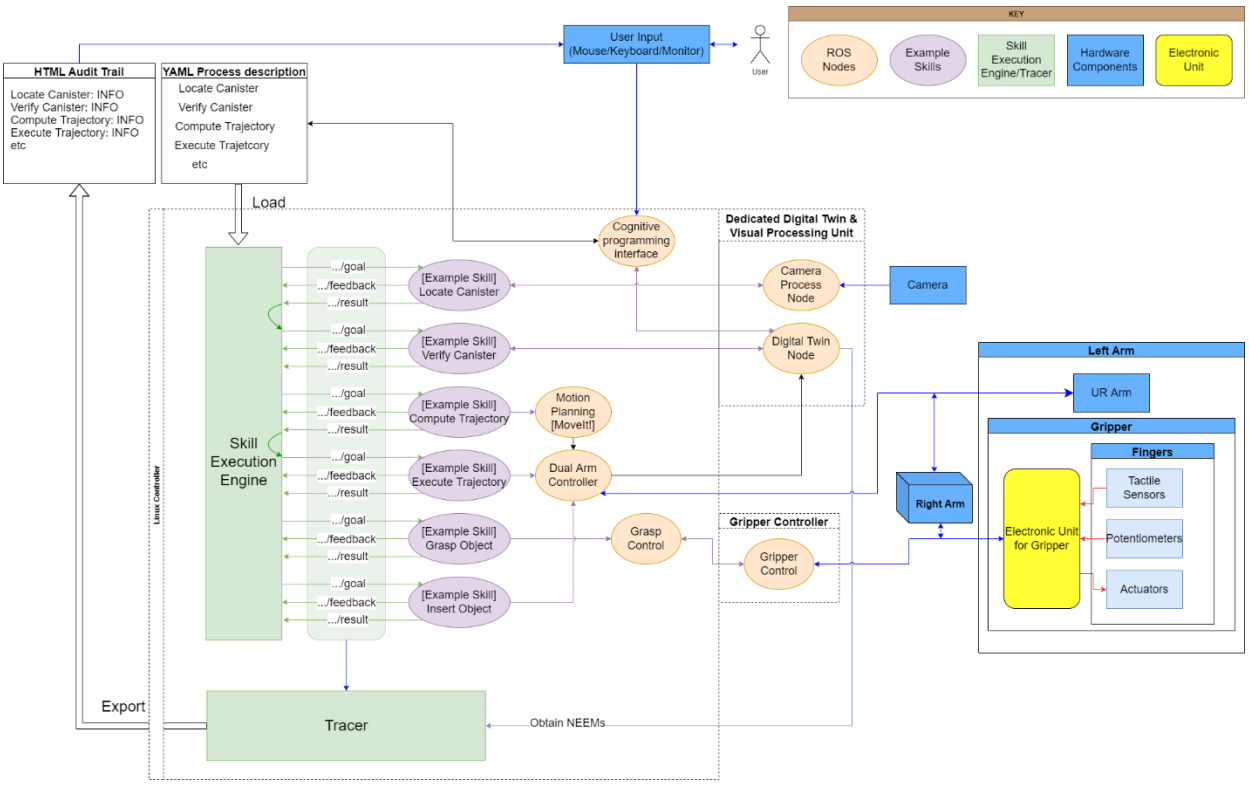

Figure 2 - System Architecture Flow Diagram.

3.1 Software framework

The development of the software framework provides a structural approach to understanding all software elements and their interactions. The proposed structure is based on initial integration performed in simulation (see D6.1 [1]). By creating the software framework for control of the simulated system first, it allows for thorough testing of the system prior to using physical hardware, reducing the risk of damage and potential costs of having to repurchase incorrect or incompatible hardware.

Horizon 2020

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

The TraceBot architecture is developed within the ROS framework, which provides crucial tools for connecting components (through standard topic, service or action mechanisms), encapsulating element complexity, reusing components developed by the community, easing the code sharing and deployment among contributors.

The definition of the overall architecture is quite complex in the sense that several key components are currently under development, and therefore not totally bounded. Nevertheless, the simplified use case analysis and implementation (mentioned in D6.1 [1] and D1.6 [2]) enabled to highlight the main components that have to be interconnected to perform the TraceBot operations. Most of them are placed in the Figure 2. They are briefly detailed now[1].

3.1.1 Skill Framework

A key component of the architecture is the skill framework that is used to define and encapsulate most of the capabilities of the robotic system. At the integration level, it presents the advantage to define a clear interface which must be fulfilled by any of the behaviours the system should be equipped with. This framework, developed by Tecnalia in T3.1, provides different implementation modes for the concept of skills, and the diagram presents one of the modalities which is based on the concept of ROS actions, for which each skill is defined through an action interface, with input, output and periodic feedback. In this framework, a robotic task is defined through a YAML process description, which defines the set of skills that have to be progressively executed to conduct the required operations. The skill framework is hierarchical, in the sense that a skill can be defined as a composition of skills.

The component in charge of loading the action plan and executing it is the Skill Execution Engine. Based on a YAML process description, it generates a hierarchical finite state machine with the required behaviour. The generated state machine represents the skill composition hierarchy, allowing the required levels of granularity for an effective responsibility distribution. The framework allows, therefore, each partner to implement the skills wrapped as a ROS action and to integrate them as a state of the state machine. Figure 3 illustrates the generated state machine for the simplified use case example. The Skill Execution Engine progressively triggers the execution of the skills, handling the input/output parameters according to the YAML plan, monitoring the good execution of the process.

[1] Most of these elements are developed in other Work Packages. When appropriate, we will refer to the related task or deliverable to get more detailed information.

Horizon 2020

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

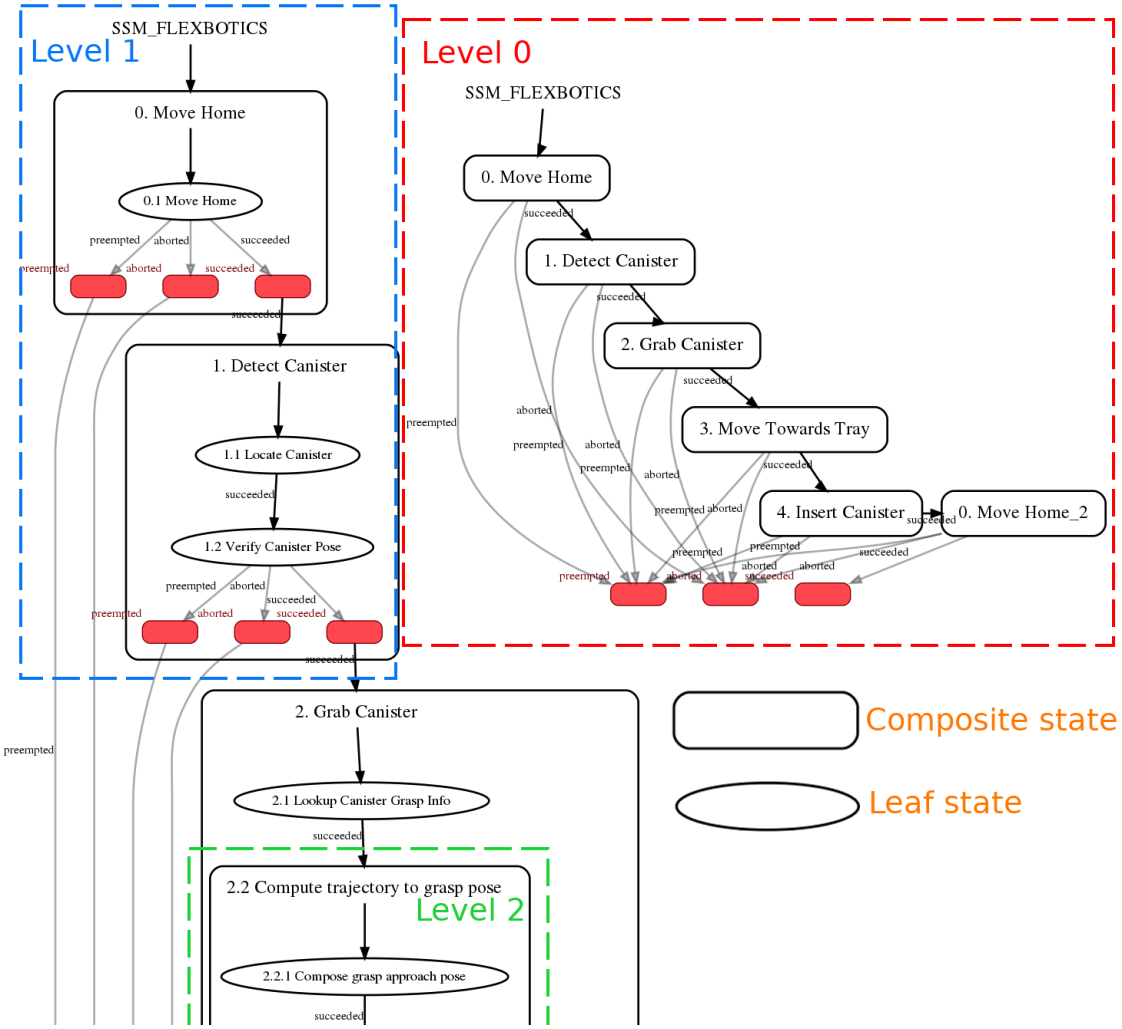

Figure 3 - Hierarchical finite state machine generated by the Skill Execution Engine. Based on the sequences of behaviours described in the simplified use case process file, *insert canister*.

3.1.2 Tracer & NEEMS

The Tracer collects information about the operations taking place, to form the basis of the audit trail. Any skill component being executed transmits all its input and output data, as well as the place of that skill within the process execution tree, to a tracer component through a single canal. The tracer component merges the information from every skill in the process into a single structure. This component, and the overall Traceability Framework, is developed within WP4, and more details about its implementation can be found in Deliverable 4.1 [3].

The tracer is also tracing the semantic of the operations executed, through its connection with the Narrative Enabled Episodic Memories (NEEMS). NEEMS are rich representations of the experiences made by the robot during a task's execution and backed by knowledge bases. They also allow introspection of the robot belief state during task executions within a given action, while also providing semantic information about the environment and the relevant entities that have been manipulated by the robot.

Horizon 2020

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

This enriches the tracer with logged methods which represent the knowledge acquired by the robot during the execution of a testing procedure, as well as related background knowledge.

The communication between the Tracer and the NEEMs is handled by our knowledge processing framework, which offers a query interface to generate, enrich or request NEEMs. Therefore, whenever a new sterility testing procedure is started in the system or new information like feedbacks, goals, or results from the skill engine are acquired, the Tracer can assert this information into the NEEM generation which is grounding the information into the ontology and establishes the semantic relations internally. After the episode is finished it is possible to extract the knowledge with queries. This allows us to enhance the audit trail with semantic information.

3.1.3 Digital Twin

The orange ellipses (Figure 2) detail several ROS nodes the architecture relies on. Among them, the Digital Twin (DT) maintains an internal representation of the world and robot state. As stated in D6.1 and further on detailed in D5.1 [4], the DT uses a physics-based game engine simulation to generate a digital replica of the estimated belief state of the world. This includes for example information provided by the perception components or the feedback of the changes of the robot configuration. For the DT, we developed a robotic simulation component for the TraceBot scenario that allows us to work with the description of the robotic hardware and import it into the game engine simulation environment. This allows the DT to be 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 at the moment) while doing manipulation actions. It can thus be used for verification purposes. Examples for this are the estimation of the expected robot or object configuration after the execution of a given manipulation action, action effects or, as illustrated on the figure, rendering the canister pose computed by the vision system to enable a correspondence estimation to check if the estimation is well aligned with the percept generated from the camera data. For testing and demonstration purposes, we propose to start with specific DT skills which can be directly added within the process files (as the `verify_canister` previously mentioned).

The Digital Twin component can be packaged as a standalone simulation program allowing it to be integrated into the system easily without being dependent on the full software stack that is necessary to be installed when developing in the game engine.

3.1.4 Cognitive Programming Interface

Prior to the execution of the plan by the Skill Execution Engine, the Cognitive Programming Interface (CPI) is the module used by the human operator to configure the process plan, selecting the appropriate skills within the database of skills of the system. The CPI provides a Graphical User Interface to allow the user to program the robot. It allows the user to define a complete task by building the desired sequence of skills the robot will perform.

Mainly, the CPI will receive the user inputs (from the mouse, maybe the keyboard) and once the sequence of skills is selected, will publish this sequence through a YAML Process description file to be processed by the robot.

This block will be made up of two nodes – the Graphical User Interface (GUI) and the planner helper. The GUI displays information to the user and allows them to select skills. The planner helper aims at supporting the user in selecting the appropriate skills depending on their requirements and effect

Horizon 2020

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

onto the environment. The helper analyses the skills chosen by the user in the GUI to deduce its potential impact on the world state (i.e if it was indeed executed by the robot). For that purpose, it should query the knowledge base about the effects of the selected skill. Then, it asks the knowledge base which skills are feasible according to the updated desired world state and sends the list of feasible skills to the GUI so that the user can select a suitable skill to build the sequence.

The knowledge base is embedded in the Digital Twin (DT) node. So each time the planner helper asks the knowledge base, its node publishes a "/query_skill_metadata" to the DT node and receives the corresponding answer.

In addition to the core operation, the CPI will have additional functionality for printing reports, user management and system maintenance.

Authorised users will be able to view and print reports relating to traceability, hardware configurations, and audit trails. This will maintain the traceability functions of the project, by allowing access to the system configurations, as well as who made/changed/authorised these configurations.

Authorised users may also make changes to, or add/remove users, and update hardware configurations. When updating users, the user can update permissions and login details. Within the hardware configuration will be editable values (with predefined ranges and precision) for the configuration of the TraceBot system. These may include Operating Speed, Locate accuracy and Grasp Attempts to name a few.

Other config details may also be updated such as GUI preferences.

3.1.5 Motion Planner

The motion planning module is the ROS tool used to perform Point to Point Planning of arm motions. We will rely on the MoveIt package [5], a well-known motion planning framework in the ROS ecosystem, which is able to cope with obstacle avoidance.

We opted to utilise an existing motion planning framework as motion planning is not one of the key research focus areas of the project. We decided on MoveIt for it's easy integration within ROS and existing off-the shelf support for the chosen UR arms. This allowed for a quick setup of the motion planning and to move towards the testing stages of the project.

3.1.6 Dual Arm Controller

Interface to the robotic arms is handled by the Dual Arm controller component. This component leverages ros_control to define interfaces to hardware resources, i.e. the robot joints and sensors. Such ros_control-enabled driver components (Figure 4) provide a means to load/unload and to start/stop controllers at runtime, ensuring there is no conflicting access to hardware resources from the controllers which use them. Each controller can request access to hardware resources managed by ros_control and may also use ROS communication primitives to interact with other systems, e.g. to receive commands or to provide feedback from the robot's sensors.

Horizon 2020

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

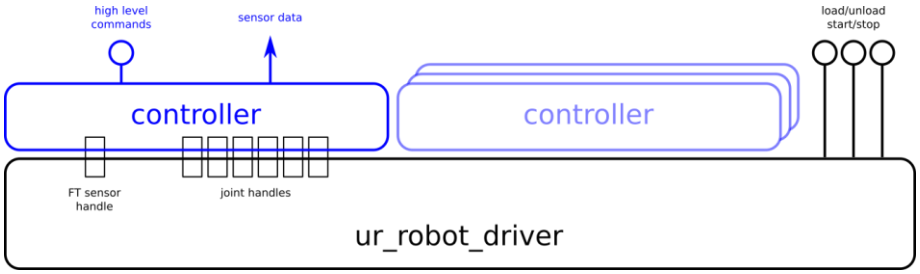

Figure 4 - A ros_control based driver with a single active controller (dark blue) and multiple stopped controllers (light blue). Controllers can be switched in real time using ROS services.

Some off-the-shelf controllers are provided by the ros-control organization, such as the *joint_trajectory_controller*, which implements the ability to execute a pre-computed joint-space trajectory, and which is already used in the TraceBot mock-up to execute trajectories generated by the *motion planning* module as illustrated by the *Execute Trajectory* skill in Figure 2. Additional controllers implementing custom control laws for specific actions will be provided as a result of task T3.2. It will contain a set of controllers to be loaded and launched for the specific manipulation to be performed (such as grasp object or insert object as illustrated in Figure 2 and Figure 3).

Making use of the composability of individual hardware abstraction components into a single executable driver component, it is proposed to combine the control of both UR10e robots from a unique *Dual Arm controller component*. This way, individual controllers will be able to claim control of the joints of both robots, allowing for a much more efficient way to coordinate control of both arms compared to using separate drivers communicating via ROS primitives.

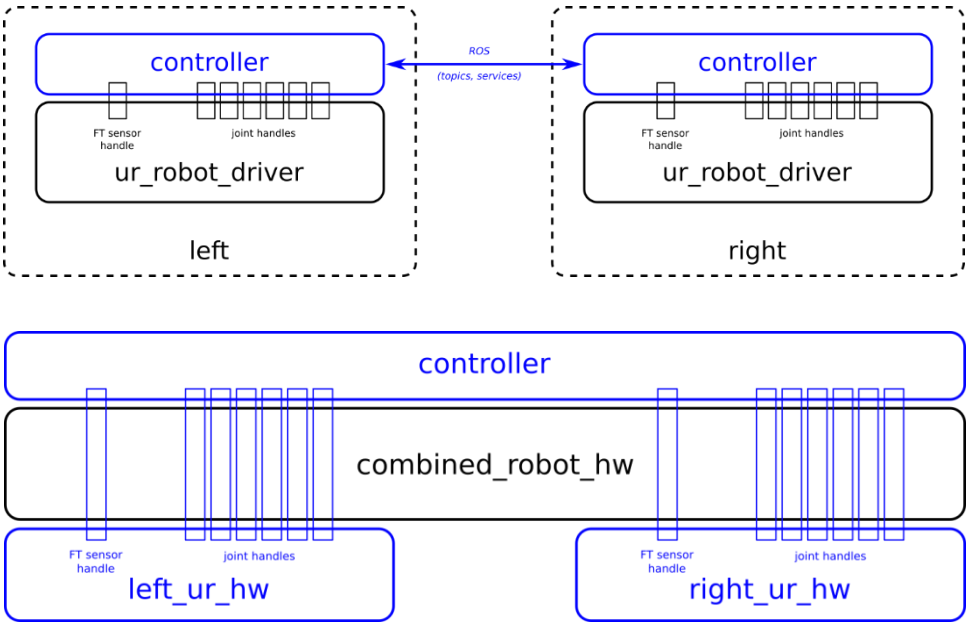

Figure 5 - A system with standalone drivers coordinating using ROS primitives (top) is much less efficient than a composite driver where a single component can directly access resources from both robots (bottom).

Horizon 2020

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

3.1.7 Grasp Control

The Grasp control is the ROS interface to the dexterous hands developed by CEA in WP2.

To facilitate the software integration, CEA will develop the low-level control command of the to-be-designed gripper. A software bridge will also be developed between the CEA gripper controller and the ROS environment to exchange gripper related data with other WPs (Figure 2/Figure 11).

In WP2, CEA plans to contribute to the gripper's low-level controller for activating the selected grasp patterns coming from the other WPs as inputs. It aims to perform some identified unitary operations that are required by the sterility testing use-case.

To activate the requested grasp pattern with the gripper, it will firstly be controlled at joint level. Control approaches will be investigated for controlling the newly-designed hand device (developed in WP2) either in position (in free space) or in force (when in contact with the object).

In that perspective, the algorithms developed in the gripper controller will be essential to determine accurately the object contact with the manipulation system and to ensure the non-sliding effect of the objects during grasping phase, taking advantage of both piezoresistive and piezoelectric tactile sensors and self-sensing actuation units. In addition, precisely controlling the multiple degrees of freedoms of the redundant gripper is challenging, especially during complex tasks such as dexterous manipulation of objects in space. To reduce this complexity, robust and performant approaches of control will be used, likely considering model-based strategies.

To facilitate the software integration, CEA will develop this low-level controller in a real-time Linux based controller. A software bridge will then be developed between this gripper controller and the ROS environment to exchange gripper related data with other WPs, especially for receiving the orders of grasp planning strategies coming from the other WPs.

More information on this can be read in Deliverable 2.1 [6]

3.1.8 Camera Processor

The camera process node encapsulates, for simplicity, the regular camera driver and the advanced image processing layers. Deep learning approaches will be deployed to detect and estimate the pose of all relevant objects in the project, while the depth sensing will be used for collision avoidance in the motion planning of the robot arms, and pose verification and refinement for the non-transparent objects. The development of this component is performed with T3.3 and T3.4, as well as in WP4 for the visual verification items.

Horizon 2020

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

3.2 Software deployment on computer units

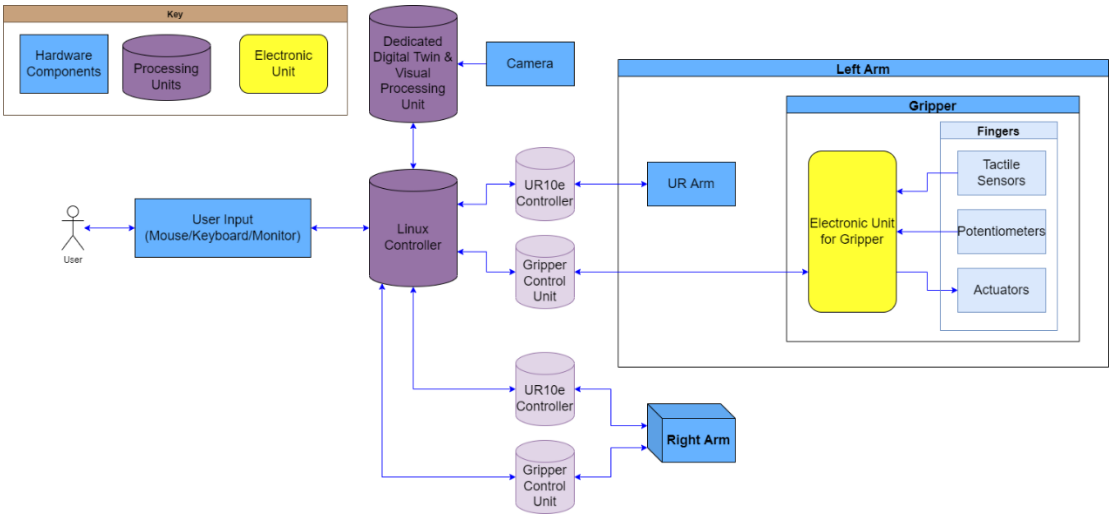

Figure 6 - Hardware Framework Flow Diagram

The flow diagram (Figure 6) is a simplified view of the system detailed in the System Architecture flow diagram (Figure 2), detailing only the hardware components of the system.

This system will consist of 5 or 6 computer units.

The two primary units (Figure 6 in darker purple) consist of the *Linux Controller* and *Dedicated Twin and Visual Processing Unit*.

The primary unit is the Linux controller which will contain the core of the system architecture, and will handle the skill execution engine, tracer and other planning controllers (as detailed in 3.1).

To assist this controller will be a Dedicated Digital Twin & Visual Processing unit. These will be running on a dedicated computer as both of these processes have strong computational requirements, in particular a powerful Graphics Processing Unit (GPU). A NVidia model is needed for both components and will be used for rendering purposes by the Digital Twin and computational purposes by the Vision layer. Recorded issues with Nvidia GPU drivers and real-time kernel needed for controllers drove the decision to put those components on different computers.

The remaining units (Figure 6 in lighter purple) are used as controller for the individual hardware components.

The Gripper Control Unit will be developed by CEA and used as a controller for both the grippers. The Linux controller will communicate with this via high-level grasp actions, and the Gripper controller will handle the conversion from grasp to joint states, as well as outputting sensor input in a usable format. Depending on the final functionality of this, there may be either one or two Gripper controllers, therefore dedicating one controller per gripper.

In addition to this there are two controllers dedicated to the UR10e arms, these are included with the purchase of the arms and manage the drivers.

More details of the other physical components are detailed in the next section.

Horizon 2020

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

4 Testbed Mechanical Design

They key hardware components are described in more detail throughout this section. With the current purchase of the UR Arms, and finalisation on the camera and gripper being used, the next stages of development will be constructing the physical system and begin implementation of drivers and testing of the integration with the existing system in place.

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).

4.1 Overall design

To date the TraceBot design has been through two iterations. The initial design (Figure 7) consisted of the robot arms mounted on a stand which is mounted directly to the table. The camera is directly about the workspace facing down at 90°. This design was initially created as a rough draft layout of the robot.

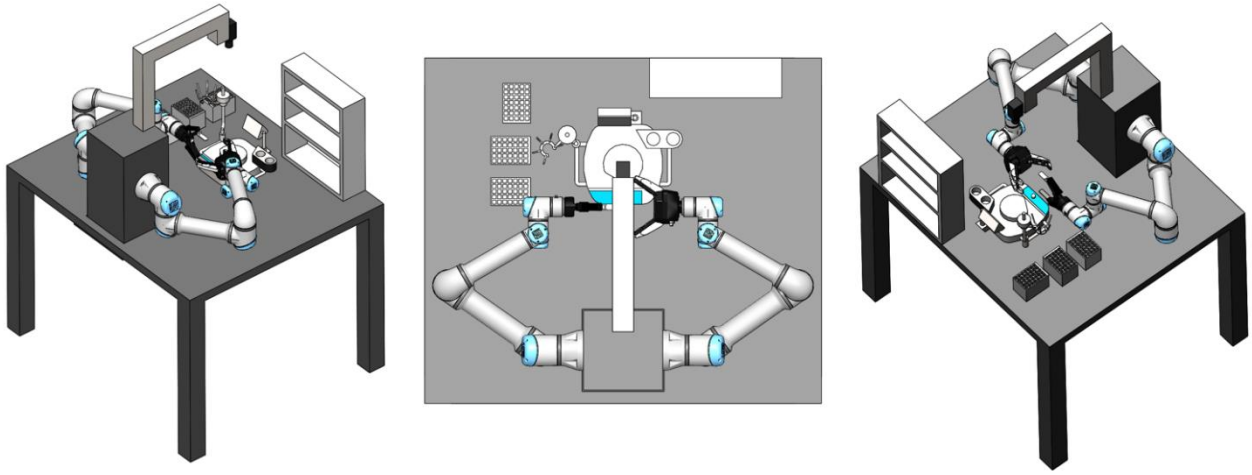

Figure 7 - Initial design for TraceBot

The current design for TraceBot (Figure 8) moves the arms onto a 45° angle to allow for greater manoeuvrability within the workspace. These are then mounted on a stand which is fixed directly to the floor. By having this fixed to the floor rather than the table, it prevents vibrations of the robot disturbing any of the loose objects on the table, as well as providing a sturdier foundation.

The camera in this design is mounted closer to the arms, and pointing down at an angle of 45° allowing for a greater FOV of the work area, and the ability to see the fronts of objects (for things such as labels or liquid levels) with reduced need to pick up the objects.

Horizon 2020

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

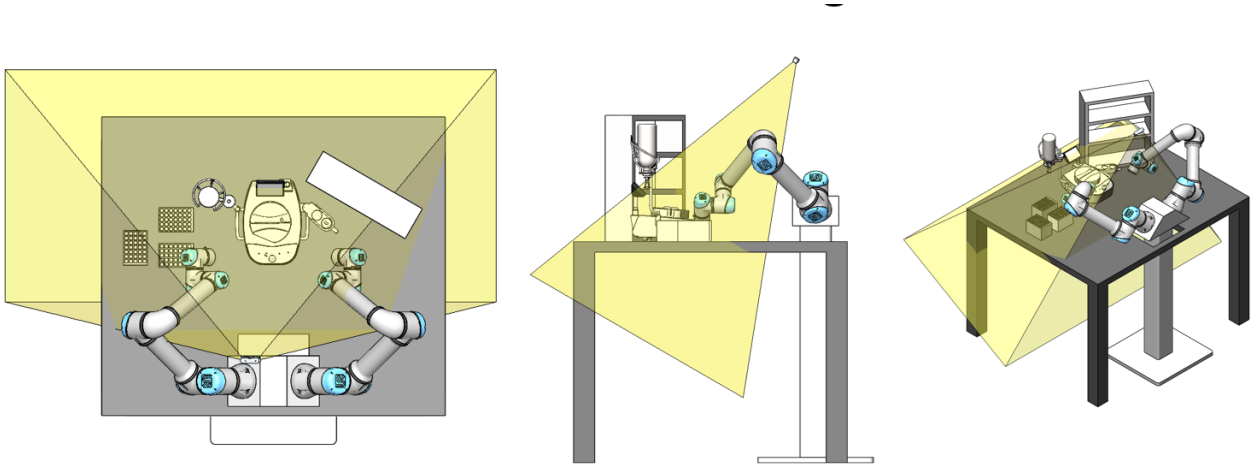

Figure 8 - Current design for TraceBot

4.2 Robot arms

The UR10e [7] arms will be used for this system build. Although the UR5e [8] was considered, with it's limited payload of 5kg it was not strong enough to lift both the CEA gripper *and* any additional payload (e.g. picking up sample bottle). This is due to the estimated weight of the gripper itself being 5kg.
Although there was discussion to have one UR5e and one UR10e, due to the potential to have one CEA gripper (on the UR10e) and one standard gripper, the final decision for two UR10e arms was made to allow for testing of one or two of the CEA grippers.

The UR10e has a 150% higher payload giving it the lifting capacity of 12.5kg. With the weight of the gripper taken into account this gives the arm a lifting capacity of 7.5kg which is more than sufficient as most objects it will be lifting will be less than 1kg.

By going with the larger arms it also gives an increase reach of up to 1.3m (51.2in), vs the UR5e's 0.85m (Figure 9).

Horizon 2020

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

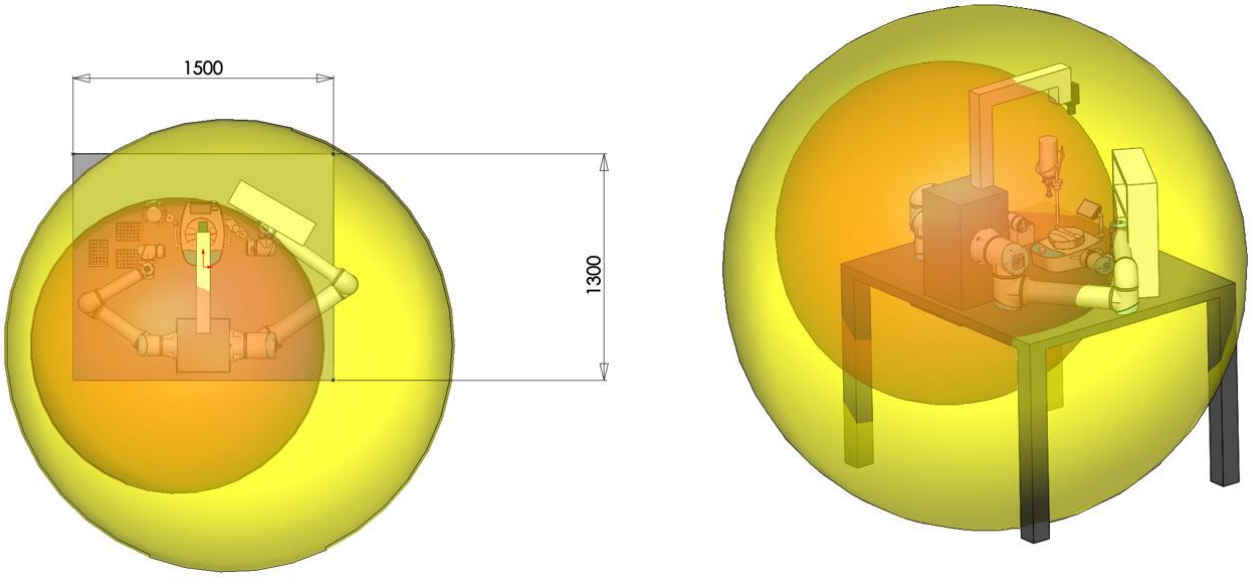

Figure 9 - Reach comparison between UR5e Robot Arm (Red Sphere) and UR10e Robot Arm (Yellow Sphere)

Interfacing of the UR Arms is greatly helped by the use of open-source 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.

4.3 Camera

Figure 10 - The Intel RealSense D435 depth camera. From left to right: NIR sensor, NIR projector, the other NIR sensor and the RGB sensor

For the camera we are using the Intel RealSense Depth Camera D435 [9]. With a range of 0.3m – 3m and a Field of View (FOV) of 87° x 58°, it is the most suitable sensor of the RealSense product range for our application.

Horizon 2020

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

This camera also has ROS drivers readily available, both for simulation and real hardware, allowing for easy implementation with the system.

When mounted at 45° (Figure 8) it provides sufficient coverage of the work area, with the robot able to lift items into view if needed for a clearer picture. Other options may also be considered, such as using multiple cameras, or eye-in-hand mounting, for better visibility when in operation.

4.4 Gripper

4.4.1 CEA Gripper

The design rationale for designing the multi-fingered device will follow the technical specifications provided by results from Task 2.1 and Deliverable D2.1 [6]. The objectives are to guarantee a high degree of versatility (i.e. capability to grasp and manipulate the various objects from TraceBot use-case), and at the same time improving the quality of force control to improve skill and overall manipulation efficiency and safety. To this respect, CEA will take advantage of innovative instrumentation technological building blocks (mainly, the actuators and the sensors embedded in the gripper) to extend its capacities in terms of force-control and force-sensing. To this end, the to-be-designed CEA gripper will exploit low-inertia, low-friction and self-sensing actuation technology, together with the previously developed tactile sensors in Task 2.2, to provide such device with high performance in terms of sensing capabilities, while also being suitable for fine finger motions.

From a control perspective, on one hand, using such force-controlled actuation technology requires a real-time based controller (Figure 6/Figure 11) (running deterministically at a few milliseconds), to ensure satisfactory performances in terms of torque sensing and control at the actuation level. On the other hand, exploiting piezoresistive and piezoelectric sensing units requires dedicated electronic units (Figure 6/Figure 11) for low-level signal treatment and digitalization. Such electronic units are foreseen to be embedded in the gripper itself, and to send data through communication bus to the main CEA real-time Linux based controller. The instrumentation and electronic units dedicated to control for the CEA grippers will be developed in the coming months in the context of the Task 2.3.

Horizon 2020

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

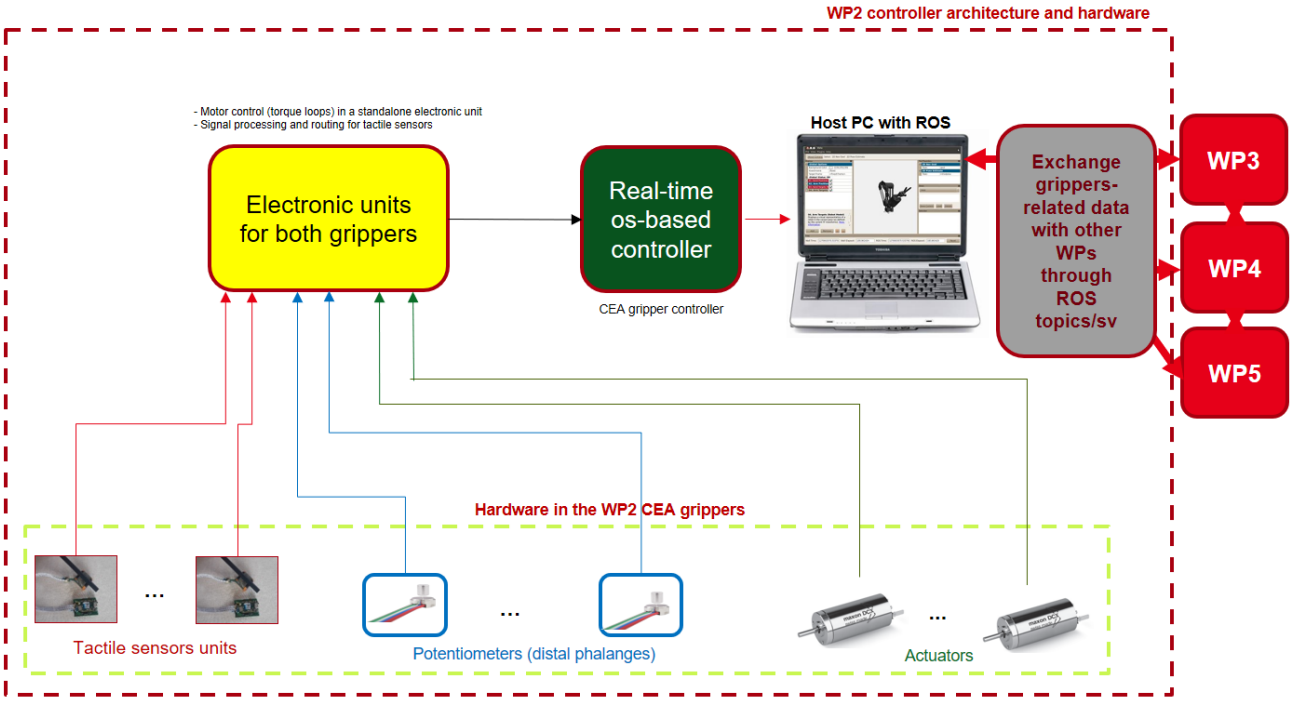

Figure 11 - Hardware and software interfaces between CEA gripper developed in WP2 with respect to other WPs.

4.4.2 Stand-In Gripper

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 Milestone 3and so an interim gripper is required for initial testing.

For an interim gripper we are looking to use the Robotiq 2F85 gripper [10] which should provide sufficient functionality for early testing of the system, until getting access to the CEA gripper.

Figure 12 - Robotiq 2F85 Two Fingered Gripper [10]

Horizon 2020

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

Using a simpler gripper such as the Robotiq one enforces us to consider use cases that can be handled with a significantly simpler tool. More advanced and dexterous manipulation will be addressed once the CEA gripper is provided. Further details on the specification of the CEA gripper is provided as part of deliverable D2.1 [6].

Horizon 2020

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

5 Risk Evaluation

As part of the standard design process, risk evaluation tools can be used in order to predict potential points of failure for a system and to consider how to mitigate them in the design. A typical tool for this is the use of a Failure Mode and Effects Analysis (FMEA). The objective of this tool is for a designer to identify potential points of failure in a system and provide features in the designed product to reduce or remove the failure points. This is typically used on complex systems composed of multiple sub-assemblies, to provide a thorough and systematic approach to ensure that all potential points of failure are considered as part of the design.

In development of an FMEA, one must consider the conceptual designs against the use case of the system (i.e. what is the purpose of the system and what obstacles are present that prevent the system from achieving its purpose).

There are 2 scenarios present, which include use of single robot arm and dual robot arms. With single robot arm, the main considerations are focused on collision avoidance and correct manipulation of objects. In the dual robot arm situation, the motion controller must make decisions on, which arm is responsible for each task – this adds an element of complexity to the system.

The complete TraceBot system has been described above completely, as a combination of the architectures and the testbed. For each specific element of the system architecture, there are deliverables in place for each respective work package that provide thorough and detailed content describing the required functionality & design of the respective architecture components and their rationale ([3] [4] [6]). Thus, a separate FMEA is not required for the architecture elements, as this would be redundant with respect to work being carried out in these deliverables.

Considering the testbed mechanical design alone, there were a few noteworthy elements that required careful consideration in order to provide an optimal design. As mentioned above, the specification of the UR5e was not sufficient for our application, as the mass of the CEA gripper alone was expected to 5kg alone – thus the decision was made to upgrade the robot arms to the UR10e specification. The other consideration was based on maximising the field of view, the camera position was amended to have a 45° field of view from the direct plane normal to the testbed surface. As the number of hardware components for the TraceBot is quite low, the need for an FMEA was deemed unnecessary at this stage of the project as the issues mentioned above have been resolved by iterative design, which was possible due to simplicity of the system needs (purely accounting for the basic functionality of the components and not for the full use case of the system). With the project containing 2 system development phases after the initial hardware integration (MS3 & MS4), it is expected that any further design amendments required to optimise that system will be captured within these iterations and that this system architecture will evolve with the project.

Horizon 2020

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

6 Deviations from the workplan

The following deviations from the workplan are listed below, with reasoning provided for their necessity.

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. The plan for these arms in 2022 will be an initial setup, integrating the work done as part of Deliverable 6.1 [1] with these arms, to provide a first demonstration of hardware control. Building off the back of this work will be further development of the testbed and associate subsystems to facilitate a successful hardware integration as part of Milestone 2.

2. **Risk Evaluation Tools** – In the original agreement for this task, discussion as made around the use of tools for risk evaluation of the system to be built, particularly around the notion of developing an FMEA. As discussed in section 5, it was decided that the use of an FMEA was not required at this point, as a comprehensive FMEA would require a completely defined use case, which is not available from the outset of the project due to its research nature (refer to Deliverable 1.6 [2] for further details on use case description).

3. **Future Developments of the System Architecture** – Due to the research nature of the TraceBot project, it is expected that this system architecture will develop and evolve with the project. Thus, it should be revisited and amended to capture any deviations from this initial draft to represent milestone 2.

Horizon 2020

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

7 Conclusion

A complete and thorough overview of the complete system architecture has been provided with a detailed description of each of the components and functions. References to relevant work package deliverables are additionally provided to provide even more detail on the development of system components. All major hardware components have been specified and listed with an initial arrange designed as a concept in order to maximise the utility of each component.

This architecture is the foundation of the work required to develop an initial hardware integration to be delivered as part of the work for Milestone 2 and is linked to tasks across WP2-WP6.

Horizon 2020

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

8 Annexes

8.1 YAML process illustration

The following figures are provided to illustrate how is structured a definition of a process, as a combination of skills, which is then loaded for execution by the Execution engine (please refer to section 3.1.1).

Figure 13 presents the YAML file from which the simplified use case is loaded. It is defined as a sequence of six successive operations or skills. In this example, all items are defined into other files. This functionality is convenient for reusing complex behaviours while hiding their complexities. The current specification only handles the data transmission in between skills when appropriate.

Data transmission is illustrated in between the 2nd and 3rd skill defined. Skill `detect_canister` generates a result, named `detect_canister_result`. This information is used by the following skill, `Grab_canister`, which requires this information as input to be well configured. The connection of skills input-output requires the knowledge of these parameter format, we are currently looking at how such connection could be facilitated for the user, in particular for such high-level operations.

```
#!/usr/bin/env -S python -m flexbotics_execution_manager

# The simplified use case is split into six meta-skills:
#   move_home
#   detect_canister
#   grab_canister
#   move_towards
#   insert_canister
#   move_home
# The sequence of skills for each meta-skill is described in separate .yml file.
# The userdata can be overwritten to share information between meta-skills.
---
!Sequence
states:
  - !Include:tracebot_process:process/move_home.yml
    name: 0. Move Home

  - !Include:tracebot_process:process/detect_canister.yml
    name: 1. Detect Canister
    result: detect_canister_result

  - !Include:tracebot_process:process/grab_canister.yml
    name: 2. Grab Canister
    params:
      canister_pose: !Link detect_canister_result.canister_pose.object_poses[0]
      canister_pose_frame_id: !Link detect_canister_result.canister_pose.header.frame_id

  - !Include:tracebot_process:process/move_towards.yml
    name: 3. Move Towards Tray

  - !Include:tracebot_process:process/insert_canister.yml
    name: 4. Insert Canister

  - !Include:tracebot_process:process/move_home.yml
    name: 0. Move Home
```

Figure 13 - Simplified use case illustration – insert Canister into tray. Root file, referencing the six skills it relies on.

Following Figure 14 present a snapshot of one of the included skills, `grab canister`. The required parameters are firstly defined in the `params` section (they are indeed configured in the upper file presented above). This skill is again a combination of skills. The second one, named `compute trajectory to grasp pose` is itself a skill defined in another file which path is indicated. A process definition is thus hierarchical and has potentially no limit of level complexity.

Horizon 2020

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

```
#!/usr/bin/env -S python -m flexbotics_execution_manager
---
!Sequence
params:
  canister_pose: !Python:geometry_msgs.msg:Pose {}
  canister_pose_frame_id: ''
states:
  - !Python:tracebot_tracer:TraceableAction
    name: 2.1 Lookup Canister Grasp Info
    action_name: /dummy_server
    action_spec: tracebot_msgs.msg:DummyAction
    params:
      placeholder_goal: "CANISTER_GRAPS_INFO"
    result: canister_grasp_info

  - !Include:tracebot_process:process/compute_trajectories/compute_grasp_trajectory.yml
    name: 2.2 Compute trajectory to grasp pose
    params:
      canister_pose: !Link canister_pose
      canister_pose_frame_id: !Link canister_pose_frame_id
    result: grasp_trajectory_result

  - !Python:tracebot_tracer:TraceableAction
    name: 2.3 Configure Grasp
    action_name: /preshape_grasp_right
    action_spec: tracebot_msgs.msg:PreshapeGraspAction
    params:
      grasp_type: !Python:tracebot_msgs.msg:GraspType
        grasp_type_id: 1
    result: grasp_configured

  - !Python:tracebot_tracer:TraceableAction
    name: 2.4 Execute trajectory to grasp pose
    action_name: /pos_joint_traj_controller/follow_joint_trajectory
    action_spec: control_msgs.msg:FollowJointTrajectoryAction
    params:
      trajectory: !Link grasp_trajectory_result.approach_and_grasp_trajectory
    result: execute_grasp_trajectory_result

  - !Python:tracebot_tracer:TraceableAction
    name: 2.5 Grasp Object
    action_name: /execute_grasp_right
    action_spec: tracebot_msgs.msg:ExecuteGraspAction
    params:
      close_grasp_type: !Python:tracebot_msgs.msg:CloseType
        close_type_id: 1
    result: grasp_object_result
```

Figure 14 - Simplified use case illustration – grab canister (snapshot)

The other skills presented on the screenshot are defined as `TraceableAction`, which is the base structure we are using to gather all partner developments. The term “action” indicates that the component is implemented as a ROS action, which name and data format are then introduced. The advantage of relying on standard ROS format is that the learning curve for defining a skill-like component is significantly reduced (“any” ROS action could be inserted in a skill process). Note that other skill models are available (such as ROS service-like, or “ad-hoc” python script for instance).

The `Traceable` part of the skill format name is related to the specific wrapper implemented to add the tracing capability. Without any change in the underlying ROS action server, the input, result and feedback of the action is automatically gathered and sent to a unique canal. This functionality added in TraceBot could be generalized to centralize traces of any action-based ROS programme.

To conclude, by parsing such file, the Execution engine is able to check the sanity of the process (under development), know which component has to be loaded (in such case mainly ROS action clients to be defined), and how to connect the outcome of one component to the following one.

Horizon 2020

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

8.2 Definitions

Term	Definition
FMEA	Failure Mode and Effects Analysis
CEA	Commissariat à l'énergie atomique et aux énergies alternatives *Alternative Energies and Atomic Energy Commission*
UoB	Universität Bremen *University of Bremen*
UR	Universal Robot
FOV	Field of View
DT	Digital Twin
VPU	Visual Processing Unit
GPU	Graphics Processing Unit
NEEMS	Narrative Enabled Episodic Memories
ROS	Robot Operating System
CPI	Cognitive Programming Interface
GUI	Graphical User Interface
WP	Work Package

Horizon 2020

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

8.3 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)

9 References

[1] S. Rosidi and B. Gordon, “D6.1 ROS Middleware Mock-Up,” 2021.

[2] T. Cichon and C.-H. Coulon, “D1.6 Use Case Specification,” 2022.

[3] M. Vincze, J.-B. Weibel, P. Mania and F. Vial, D4.1 Traceability Framework for Laboratory Automation, 2022.

[4] M. Beets, “D5.1 Definition of the Conceptual and Reasoning Framework and Semantic Models,” 2021.

[5] I. Sucan A. and S. Chitta, “"MoveIt", [online] Available at https://moveit.ros.org/”.

[6] M. Grossard, F. Gosselin and J. Escorcia, “D 2.1 Technical specifications as recommendations for the design of the multi-fingered gripper,” 2022.

[7] “UR10 Robot,” Universal Robots, [Online]. Available: https://www.universal-robots.com/products/ur10-robot/. [Accessed 31 01 2022].

[8] “UR5 Robot,” Universal Robots, [Online]. Available: https://www.universal-robots.com/products/ur5-robot/. [Accessed 31 01 2022].

[9] “Depth Camera D435,” Intel RealSense, [Online]. Available: https://www.intelrealsense.com/depth-camera-d435/. [Accessed 2022 01 31].

[10] “2F85-140 adaptive robot gripper,” Robotiq, [Online]. Available: https://robotiq.com/products/2f85-140-adaptive-robot-gripper. [Accessed 31 01 2022].

Horizon 2020

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