TRACEBOT

Basic Software models: Software interface for first demonstrator defined and required simulation aspects

Deliverable 5.2

Horizon 2020

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

Deliverable Title	D5.2 Basic Software models: Software interfaces for first demonstrator defined and required simulation aspects
Deliverable Lead:	University of Bremen (UOB)
Related Work Package:	WP5: Traceable Semantic Twin: Planning, reasoning, Audit Trail
Related Task(s):	T5.1: Definition of the domain process structure and taxonomy T5.2: Replication of medical lab environments into Traceable Semantic Twin Knowledge Bases for Reasoning T5.3: Traceability-aware process (re-)planning, reasoning and regulatory digital audit trail
Author(s):	Prof. Michael Beetz
Dissemination Level:	Public
Due Submission Date:	11/30/2022
Actual Submission:	11/29/2022
Project Number	101017089
Instrument:	Research and innovation action
Start Date of Project:	01.01.2021
Duration:	51 months
Abstract	This deliverable describes the basic software models of the reasoning apparatus and the integrated simulation aspects for the TraceBot project. The key approach in the reasoning framework is the usage of a hybrid knowledge-based approach whose interfaces will be described by the key building blocks.

Horizon 2020

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

Versioning and Contribution History

Version	Date	Modified by	Modification reason
v.01	15.11.22	Prof. Michael Beetz(UOB)	Initial version
v.02	28.11.22	Prof. Michael Beetz(UOB)	Revision based on internal review

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 main objective of this deliverable is an interface description for the interaction between the components developed in work package 5, mainly the reasoning apparatus, and the other modules of the TraceBot ecosystem for the first real world demonstrator. The description also links the different interfaces to the simulation aspects that have been developed in the Semantic Digital Twin(semDT) as well as the underlying symbolic knowledge representation.

Key components are presented on a functional level to describe the expected inputs and the results. The description is divided into the main building blocks for the reasoning apparatus, namely the knowledge representation & reasoning, simulation-based reasoning, physical reasoning and imagistic reasoning in the perception executive. The specification is formulated in a query-answering scheme which connects the use case specific reasoning tasks to the capabilities provided by the components of work package 5.

2 Introduction

Robotic agents tasked with complex manipulation actions in sterility testing scenarios and human-friendly environments, are posed with many uncertainties and underspecified action descriptions. In order to solve these challenges, the robot needs an extensive reasoning apparatus to compute the necessary parameters for successful action executions and ultimately task completion. In TraceBot, the reasoning architecture is a hybrid-based system which combines the strengths of symbolic reasoning with a simulation-based component[1]. The latter is a game engine based simulator, which provides realistic physics simulation as well as photorealistic rendering to enable the robot to hypothesize a subsymbolic belief state about the world. This belief can then be used to reason about the expected task outcomes and compare it to the actual state of the robot. Additionally, it also allows the robot to express in detail what the current belief about the world was at every timestep during the task execution while being able to ground information into the symbolic part of the knowledge base.

In this deliverable we will describe the interfaces top-down. At first, we will give an overview of how the robot is interacting with the overall system architecture showing the high-level data flows. After that, we will provide more details about the key building blocks of the reasoning system and describe the functionality of these components. This includes the expected inputs and results of each of these functions and a description about the relation to the TraceBot project. The key components of the reasoning framework will be described in a query-answering scheme in three steps. In the first step, we will present the knowledge representation and reasoning subsystem which enables the robot to ground the manipulation actions that are to be executed, into a symbolic knowledge base.

[1] For a conceptual introduction to the overall approach of Traceable Semantic Twins(TST) and Semantic Digital Twins(semDT), we kindly refer the reader to Deliverable 5.1 which provides an overview of the key elements in the hybrid architecture and introduces core components.

Horizon 2020

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

This provides a machine-interpretable structure, which can then be refined with additional properties, descriptions of partaking entities and results of reasoning operations over the course of the executions and learned data. It also enables the robot to reason about handling task relevant entities, like for example the objects that are to be manipulated. This allows the robot for example to access context-dependent information for the task execution, like for example a suitable grasp pose given the relevant object and the state of the robot actuators. To cover the TraceBot related interface requirements, we have made significant efforts to design suitable interfaces with our partners in the consortium based on the different requirements that have been identified.

The two following steps in our interface specification, are focused around the simulator capabilities that the Semantic Digital Twin(semDT) offers. We will describe how the simulator is interacting with the other software components of the robot in order to gain the possibility to ground the belief state into the game engine based semDT. In this area, we can mainly divide the efforts into two categories: flexible robot simulation and environment/world model manipulation. For the TraceBot project, we develop a comprehensive robot simulation for the semDT, which is capable of simulating not only the movement of the robot hardware that is used in the project, but also physical interaction with objects in the sterility testing environment. The goal is, that the robot in the simulation can execute the targeted tasks and therefore reason about the intended process steps in a subsymbolic way. Besides the robot simulation, we also need to update the environment model of the simulator, based on the object belief state changes. In this case every time, the perception is called and returns new information about the objects in the surroundings of the robot. This technical foundation also enables us to investigate methods of imagistic reasoning, which we see as means that are not limited to the symbolic level, but also uses information from the visual domain by using the rendering of the expected camera image given the hypothesized belief state. In the third step of the description, we will outline the interface that will enable robots in the sterility testing use case to state imagistic reasoning queries to reason about the task outcomes in the individual substeps of the TraceBot process.

We have created a video that demonstrates the first integration effort and shows the general interaction between the robot and the semDT, which can be found here: https://youtu.be/fSFGd6tFg3Q. The video features an explanation of the basic capabilities and the interfaces of the components developed in work package 5. The current integration efforts between the first real world demonstrator and the semDT are ongoing in parallel with the creation of this document, but features mostly the same concept that is demonstrated in the linked video and is therefore envisaged to visually support the proposed concepts. We have also added links to public implementation repositories in the corresponding chapters.

3 Description of work & main achievements

The main content of this chapter is a systematic overview of the developed software models for the first demonstrator and the interfaced that have been developed over the course of the project.

Horizon 2020

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

Simulation aspects and interaction is described in subchapters that provide an overview from the conceptual high-level down to the individual key building blocks of the system.

3.1 Reasoning framework overview

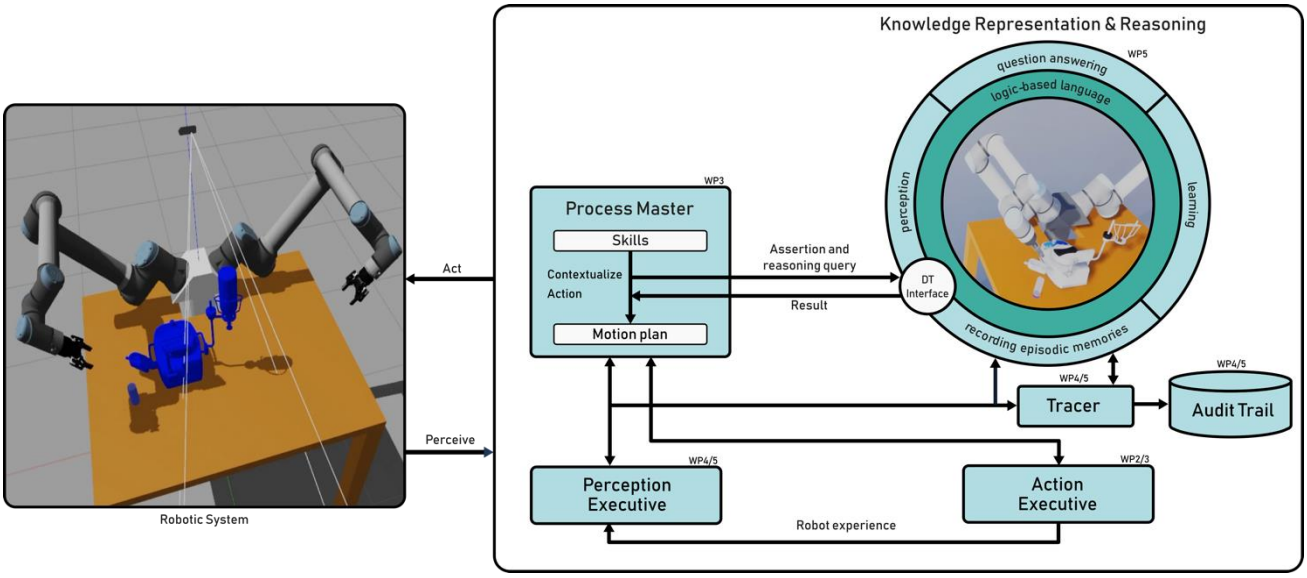

FIGURE 1: OVERVIEW OF THE INTERACTION SCHEME BETWEEN THE COMPONENTS IN THE TRACEBOT ECOSYSTEM, HIGHLIGHTING THE CONNECTION BETWEEN THE ROBOTIC SYSTEM AND THE KNOWLEDGE REPRESENTATION AND REASONING INCLUDING THE SEMANTIC DIGITAL TWIN (SHOWN IN THE CIRCLE ON THE RIGHT).

In this chapter, we would like to introduce the interfaces in the TraceBot ecosystem from a high-level perspective and explain the different key components of the work package 5 top-down. As illustrated by Figure 1, a conceptual framework for the knowledge representation and reasoning system in TraceBot was elaborated, validated and presented in the former deliverable D5.1. The overview shows the general interaction between the real robotic system on the left and the system architecture on the right, which is in a continuous perception-action loop. Observations are forwarded to the system, reasoned about and translated into action commands that will finally move the robot actuators to execute the required manipulation actions in the sterility testing use case. The internal reasoning processes in the system are important to make informed decisions, parametrize the manipulation actions correctly and maintain a belief state about the state of the environment and the task.

Essentially, the overarching goal of the TraceBot robotic system is to maintain deep knowledge and understanding about the environment and the ongoing processes in order to sufficiently inform the process executive about the process state so that the TraceBot scenarios can be successfully accomplished but also and very importantly to report and give accounts of the process outcomes (e.g., audit trail). In order to achieve this information exercise, the system applies various reasoning techniques such as pure logics-based reasoning, physical reasoning and imagistic (i.e., minds eye)

Horizon 2020

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

reasoning on a very rich representation of the world, which is implemented as a semantic (i.e., ontology-grounded) physico-realistic digital twin of the environment termed as semDT.

3.2 A formal query language as a knowledge interface in the TraceBot robotic system

In terms of system interfaces, the above description suggests that the knowledge representation and reasoning in Figure 1 can be regarded as the core information provider to the rest of the system and should therefore handle the following fundamental interactions of a query process with the rest of the system:

- TELL: the rest of the system enriches the hybrid knowledge base (symbolic knowledge and semDT) of the knowledge representation and reasoning module. In other words, the knowledge representation and reasoning receives information about changes to the environment. It can be an assertion (i.e., new fact), an update (i.e., changing existing facts), or a removal (i.e., deleting existing facts). Instances of such queries can be asserting that a canister has been detected, updating the pose of an asserted canister or deleting the fact that an object has a certain disposition: the canister is no more graspable after it has been grasped.
- ASK: the rest of the system queries the knowledge representation and reasoning for some specific information that should be retrieved, derived or constructed through reasoning steps. This is for instance about requesting the pose of an asserted canister, verifying the success of an action or collecting NEEMs (Narrative-Enabled Episodic Memories) for elaborating audit trails.

3.2.1 Vocabulary, grammar & semantics from logics-based world ontology

In order to efficiently achieve sufficiently detailed TELL/ASK interactions between the knowledge representation and reasoning and the rest of the system, we propose a sufficiently rich and logics-based formal language for the query process, which is presented in the following sections.

To summarize, the grammar, vocabulary and semantics of the language are intrinsically built around the structure and content of knowledge in the ontology in terms of concepts, attributes of concepts and relations among concepts but as well as around the formal description logics languages (e.g., OWL, RDF, SWRL) in which the ontology is defined. That is, a query is essentially a conjunction of predicates where the predicate is designated by a specific concept, attribute or relation in the ontology and the parameters are either known or unknown instances of the predicate. For instance *instance_of(X, Canister)* means that X is an instance of the concept or class or category Canister, while *subclass_of(Canister, Container)* means that Canister is a sub- class or sub-concept or sub-category of Container. This being said, the task of the knowledge representation and reasoning after receiving a query consists on the one hand in instantiating the unknown parameters of the predicates through retrieval, derivation or construction of facts and on the other hand in asserting, retracting or updating facts in a way that the whole predicate evaluates to true. Then, a boolean answer is returned on whether the processing of TELL-query or the ASK-query as decision problem

Horizon 2020

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

was successful or unsuccessful and the value of the unknown parameters are returned as answers for ASK-query as a search problem. Notice however that specifying a query merely as such a conjunction of predicates remains ambiguous with respect to the above TELL/ASK behavior. Imagine that the query *instance_of(canisterId1, Canister1)* is sent to the system, this could be to tell that the individual canisterId1 is an instance of the Canister concept, but it could also mean at the same time to ask whether the individual canisterId1 is instance of the Canister concept. In order to escape this ambiguity, the query as conjunction of predicates is further passed to one of the predicates below:

- *kb_call(P)*: passing a conjunction of predicates to kb call causes the system to regard P as an ASK- query which is either a decision problem in which case a boolean answer should be returned (e.g., *instance_of(canisterId1, Canister1) = True*) or as a search problem in which case all the unknown parameters should be instantiated in order to make P true. If no answer is found then the empty set is returned and the predicate evaluates to false (e.g., *instance_of(X, Canister) = True and X=canisterId1*).
- *kb_project(P)*: passing a conjunction of predicates to kb project causes the system to regard P as a TELL- query in which the facts in P are asserted in the knowledge base (semDT) and true is returned to certify the success of the assertion. However, if P is going to cause the inconsistency of the knowledge base, then the assertion will fail and false will be returned. (e.g., *kb_project(instance_of(canisterId1, Canister)) = True* and canisterId1 is now known as an instance of Canister).
- *kb_unproject(P)*: passing a conjunction of predicates to kb_unproject causes the system to regard P as a TELL-query in which the facts in P are retracted from the knowledge base. (e.g., *kb_unproject(instance_of(canisterId1, Canister)) = True* and canisterId1 is no more known as an instance of Canister)

3.2.2 From extended world ontology to extended language

Notice that the power of the language resides in the richness of the ontology. For this reason, we enrich this language by extending the SOMA[4] ontology (Socio-physical Models of Activities) presented in D5.1 with TraceBot-specific object-related, action-related and state-related concepts.

- Object-related Concepts: As illustrated by Figure 2, SOMA has been extended with abstract and concrete commonsense knowledge about objects in the TraceBot environments (e.g., Canister, pump), their property (e.g., size, material, shape, mesh) and the relations among them (e.g., cap is part of bottle).

- http://www.ease-crc.org/ont/SOMA.owl#DesignedContainer
  - http://www.semanticweb.org/smile/ontologies/2022/3/TraceBot#Canister

Horizon 2020

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

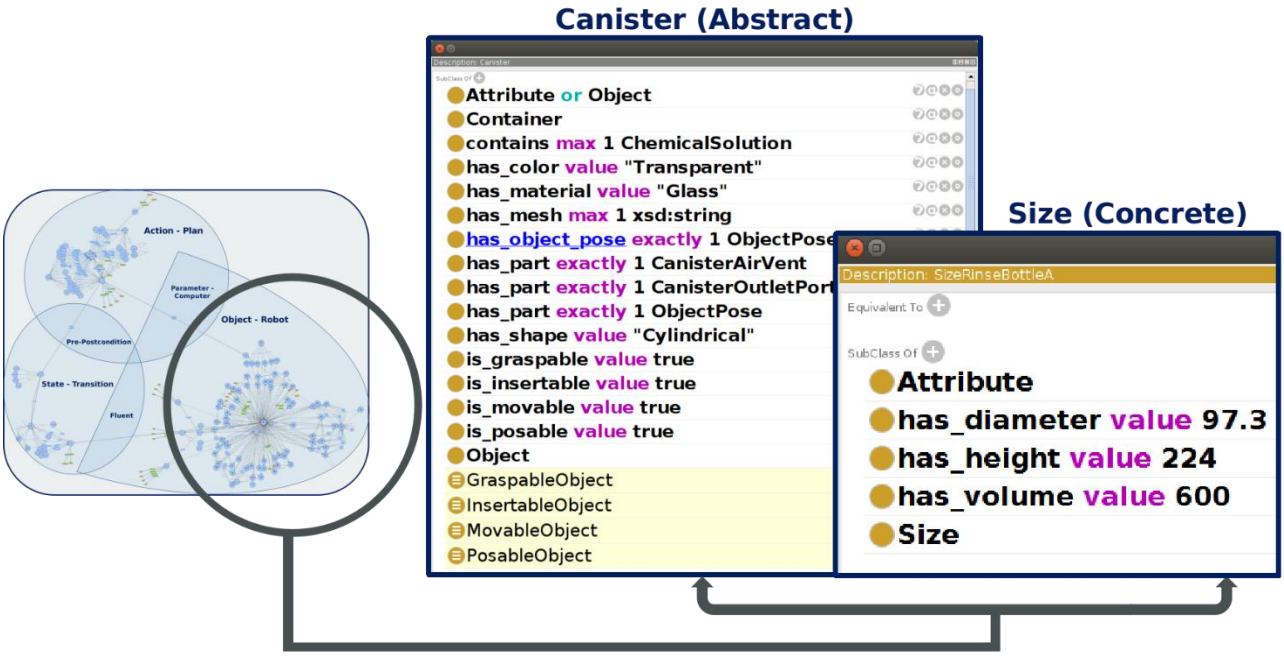

Figure 2: Extending SOMA with abstract and concrete commonsense knowledge about TraceBot-specific objects.

A query for accessing the diameter of the canister will then be:

```
kb_call(
        [
           subclass_of ( Canister , A ) ,
           has_description ( A , exactly ( has_size , 1 , B ) ) ,
           subclass_of ( B , C ) ,
           has_description ( C , value ( has_diameter , D ) )
        ]).
```

The value of the canister's diameter will be stored in the variable D. The query states that A is a superclass of Canister and has exactly one size of type B. And C is a superclass of B and has a diameter of value D.

- Action-related Concepts: Likewise, as illustrated by Figure 3, SOMA was extended with commonsense knowledge about TraceBot-specific actions in terms of identity, parameters, plans and the skills from work package 3 which realize these actions were grounded in the ontology as well so that while reporting about the process course and outcomes, one can make use of the ontology for understanding.

- http://www.ease-crc.org/ont/SOMA.owl#PhysicalTask

  - http://www.ease-crc.org/ont/SOMA.owl#Manipulating

Horizon 2020

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

- http://www.ease-crc.org/ont/SOMA.owl#PickingUp

- http://www.semanticweb.org/smile/ontologies/2022/3/TraceBot#TraceablePickingUp

An example of query for saving information about executed action and even for recording NEEMs, which are

foundations for audit trails, will then be:

```
kb_project([
        new_iri(Episode, soma:'Episode'), is_episode(Episode),
        new_iri(Action, tracebot:'Grasping'), is_action(Action),
        new_iri(Object, tracebot:'Canister'), is_object(Object),
        new_iri(Agent, tracebot:'LeftUR10Arm'), is_agent(Agent),
        new_iri(TimeInterval, dul:'TimeInterval'),
        holds(Action, dul:'hasTimeInterval', TimeInterval),
        holds(TimeInterval, soma:'hasIntervalBegin', StartTime),
        is_setting_for(Episode,Action),
        is_performed_by(Action,Agent),
        has_parameter(Action,Object)
        new_iri(Role, soma:'AgentRole'),
        has_type(Role, soma:'AgentRole'), has_role(Agent,Role)
]).
```

In the above query, the first line asks the system to create an instance of soma:'Episode' and store the id of the instance into the variable Episode. The second line does create likewise an instance of the action tracebot:'Grasping' and store the id into the variable Action. The third line does create an instance of tracebot:'Canister' and the id is saved into the variable Object. Then, an agent of type tracebot:'LeftUR10Arm' is created and stored into Agent. The time interval in which the action takes place is created as well and the action is associated to the Episode. Finally, the agent of the action as well as the parameter of the action are set as well.

Horizon 2020

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

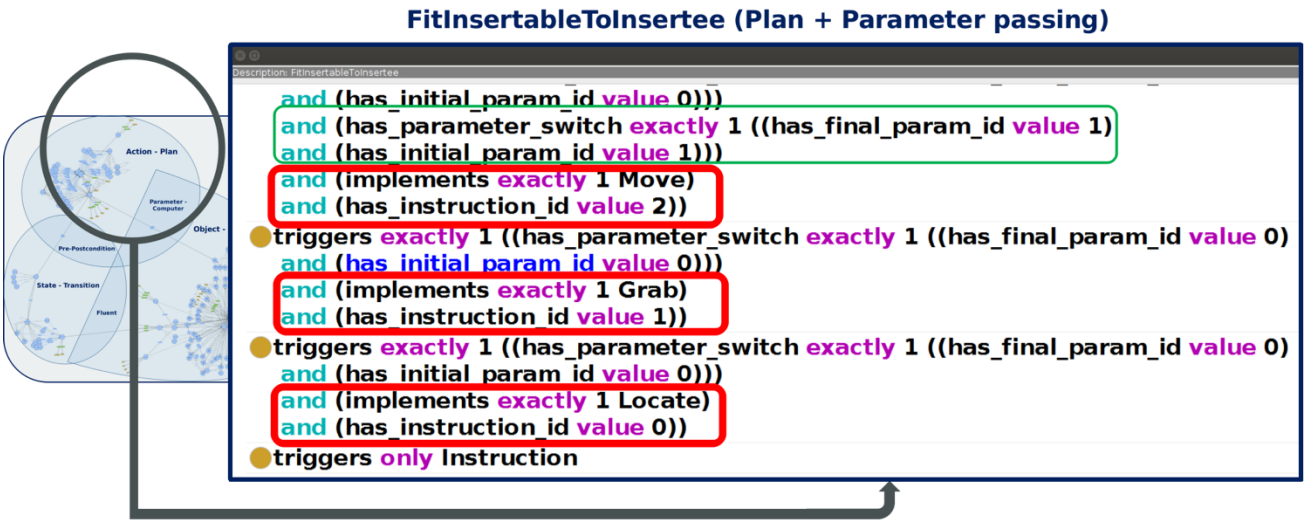

Figure 3: Extending SOMA with abstract and concrete commonsense knowledge about TraceBot-specific actions: Illustration with the typical FitInsertableIntoInsertee action from TraceBot which is about inserting an insertable object into another object called insertee.

- State-related Concepts: In TraceBot, verification and validation of processes are primordial and very crucial operations in this regard are the feasibility verification of actions prior to execution and the success verification of these actions after execution. In order to overcome these challenges, the concepts of state- transition were introduced in order to model the effects of actions as well as the predispositions of the environment w.r.t. these actions. And once these predispositions and effects have been modeled such as illustrated by Figure 4, one can basically make use of the predicates is successful(A) and is feasible(A) to check whether the Action A was successful whether it is feasible.

After creating and saving information about an action in the knowledge base such as shown in the previous point, one can query its feasibility and success as follows:

```
kb_call(
   [
     is_feasible ( 'http://.../ontologies/2022/3/TraceBot#Grasping_Instance1')
   ]
).


kb_call(
   [
     is_successful ( 'http://.../ontologies/2022/3/TraceBot#Grasping_Instance1')
   ]
).
```


Horizon 2020

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

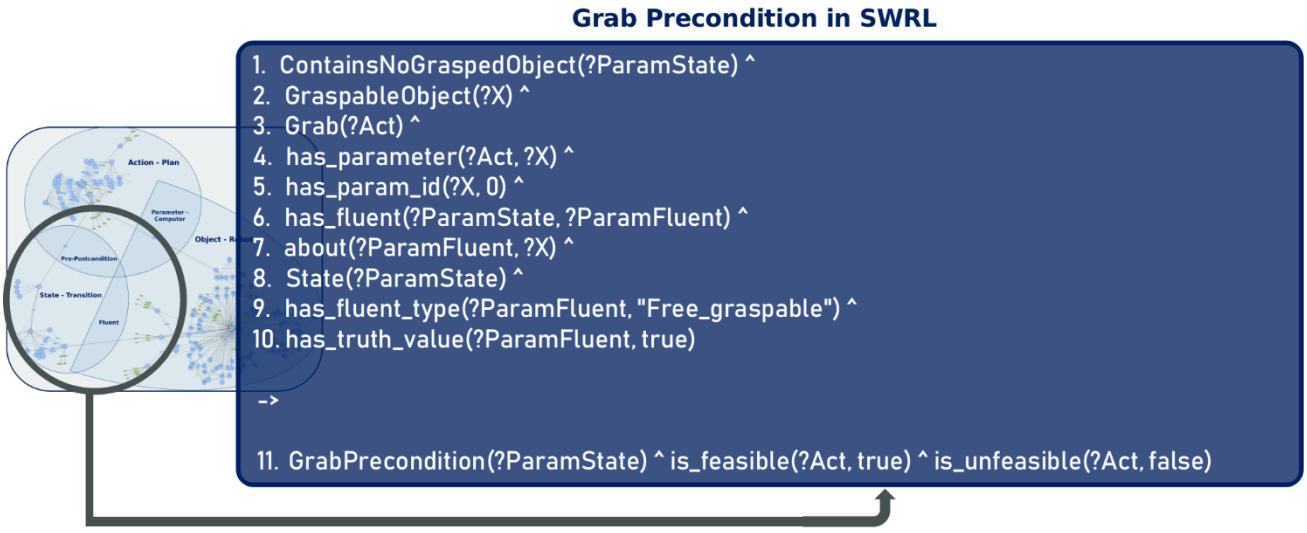

Figure 4: Extending SOMA with abstract and concrete commonsense knowledge about TraceBot-specific actions' pre- and postconditions: Illustration with the typical Grab-Grasping action from TraceBot.

This grasp/grab precondition definition states that (1) (8) if ?ParamState is a state containing no grasped objects (expressible with OWL), (2) ?X is a graspable object, (3) ?Act is a grab action, (4) ?X is a parameter of ?Act and (5) ?X is the first parameter, (6) ?ParamFluent is a fluent (i.e., time-variable property) in ?ParamState, (7) ?ParamFluent is about ?X, (9) ?ParamFluent defines the property Free_graspable (i.e., true if the concerned object is not actually grasped), and (10) ?ParamFluent has the value true, then (11) ?ParamState is a precondition of ?Act and ?Act is therefore feasible.

The previously described ontology provides the knowledge-based structure for storing and accessing information about action executions and reasoning about the quality of the task outcomes. Audit Trail generation in the TraceBot system is the combination of the Tracer component, which is assessing the high-level information from the process master, and NEEMs. Process information will be asserted via the TELL (Section 3.2) interface which grounds the action and object information into the knowledge representation and reasoning system and therefore links the task tree with semantics about the execution. For more details, we kindly refer to Deliverable 4.1.

So far, the role of the knowledge representation and reasoning in the TraceBot robotic system has been recalled and a sufficiently rich and formal query language has been presented in detail as an interface between the knowledge representation and reasoning and the rest of the system. However, the language has been only illustrated at a higher-level of semantics (i.e., with queries that performed complex tasks). In the rest of the document, we further present how these higher-level queries are further broken down into specific queries that are solved by various reasoning methods namely the physical and imagistic reasoning fundamentally based on physico-realistic simulations of the ongoing processes.

Horizon 2020

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

3.3 Simulation-enabled reasoning

The semDT provides a simulation environment[2] for robotic agents[5]. The task of this environment is to emulate actions of the real robot, build a continuous belief state of the real world and support physical and imagistic reasoning. Based on the real hardware setup, different requirements for the simulation exist. In order to match the state of the real robot, we need to be able to spawn the robot with different joint configurations, since calibrating can change this. Furthermore, perception makes it necessary that we are able to add, move and get information of objects.

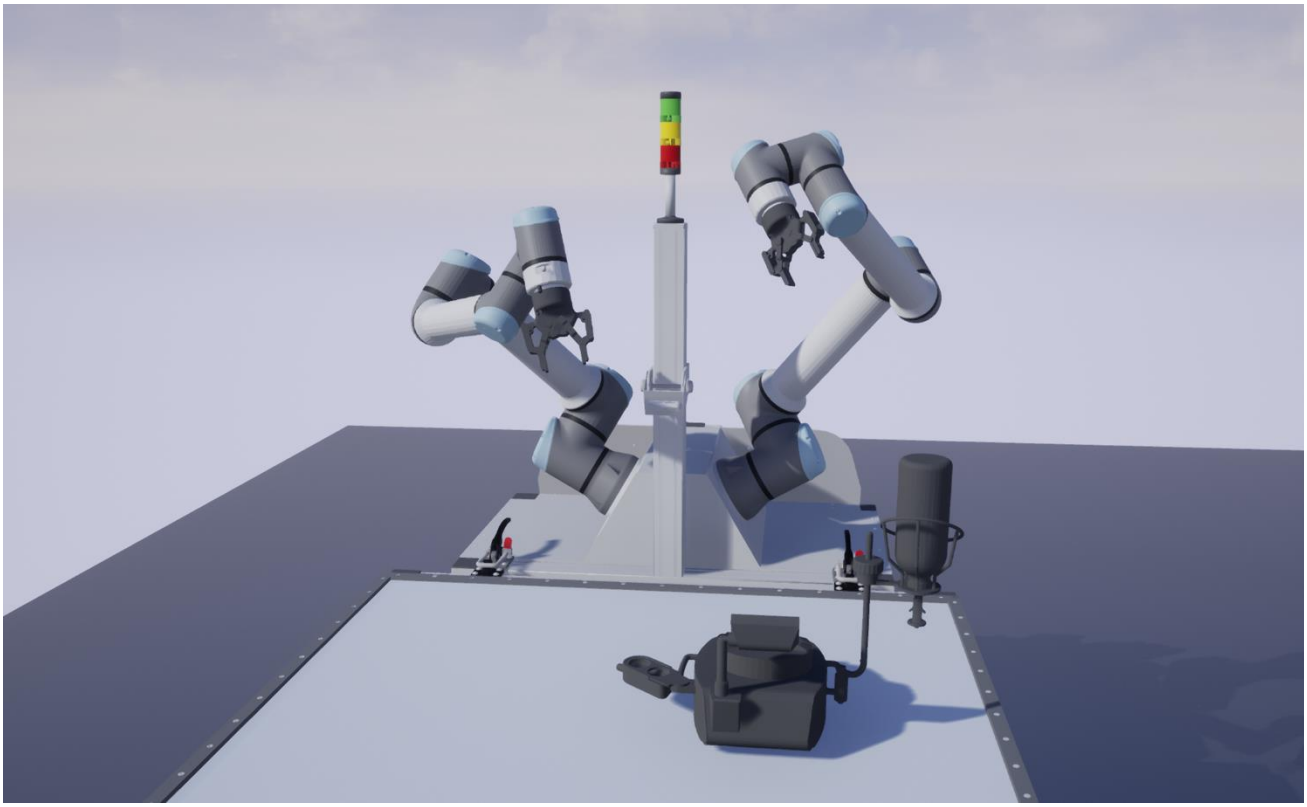

Figure 5: The semantic digital twin of the current revision of the TraceBot demonstrator constructed by Astech in our simulation component.

As part of the semDT, the simulation environment supports physical reasoning, which will also be motivated in more detail in the following chapter. In addition, the simulation environment enables imagistic reasoning. We have integrated a real-time, virtual RGBD sensor into our simulator, which enables us to render RGB and depth images from given camera parameters. This is realized by a game engine plugin that is using the rendering functionalities of the underlying Unreal Engine to

[2] https://github.com/urobosim

Horizon 2020

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

generate percepts from an arbitrarily posed camera in our virtual world model. The plugin is also realizing the interface between the game engine simulation and the TraceBot ecosystem, by providing individual ROS topics to fetch the required percepts. Important for the imagistic reasoning is also a method to update the game engine based belief state after perception actions. Whenever object poses are updated by the vision components of the system, these changes should be immediately reflected in the game engine based belief state. For this, the simulation- enabled reasoning offers an interface to update the virtual world in the simulator from the other components of the overall system. This encompasses the addition, updating and removal of objects. Whenever a new object shall be handled by the skill plan execution, vision components are detecting the new or changed objects and assert them into our knowledge base and the simulator.

The communication between the simulation is realized by services and topics. A service is a synchronous communication between a client and server. This means the client, which sends a request to the server waits for the response before continuing with the task execution. We defined them as service(Type, [Request, Response], Parameter1, Parameter2, ….).

The second communication type are topics. This type of communication is asynchronous where one or more publishers can broadcast data to a topic, from which a subscriber can retrieve the data. The communication is asynchronous because the publisher and subscriber are decoupled of each other. The publisher can proceed with his tasks after publishing the data. Subscriber call a callback method when new data is published to the topic to process the data. If the subscriber is slower than publisher it uses the most recent data on the topic. Subscriber can also stop listening to the topic without affecting the publisher. Topics support m to n relationships between publishers and subscribers. We define them as topic([topic type, action]).

The use cases in TraceBot required us to enhance the simulation capabilities of Unreal Engine with modeled logic functionalities for situations that are not natively supported. Such use cases include the relationship between needle and needle cap as well as the insertion of the needle into the septum. In order to simulate this, deformation would have to be simulated accurately. As this is not the possible with the current technology under the given time constraints, we introduced object-interaction models defined as logic(Type, Parameter1, Parameter2, ….) that model the expected behavior.

In order to fulfill the requirements of TraceBot, the following functionalities were implemented and made available via interfaces:

- The spawning of the robot during runtime is handled by service(SpawnRobot,[robot_description, result], Model)

  The request robot_description contains the kinematic chain of links and joints of the robot, while the answer result contains information about successful spawning the robot. The second parameter is the effect of the service, which is the spawned Model inside the semDT. The robot automatically initializes and connects to the ROS system after spawning. This ensures that the robot can automatically receive the state updates from the real world robot and receive commands.

- Emulating the motion of the real robot: topic([joint_state, Motion])

Horizon 2020

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

The simulated robot subscribes to the joint state topic published by the real robot, in order to replicate the motion of said robot. For our application, the relationship is 1 to n, n >= 1. There is only one publisher for the joint state, but there will be more then one subscriber to this topic, among others the simulator as well as the tracer.

- Spawning objects in the simulation is handled by service(SpawnObject, [object_description, spawn_information], Object)

  When an object is perceived for the first time it has to be added to the simulation. In order to spawn the correct object, we have to send the request object description to the simulation. This request contains parameter such as pose, mesh, material and physical properties of the object to spawn. The spawn_information contains the id of the object as well as the name of the spawned object. This could differ from the name in object_description to make it unique in the semDT.

- If an object was already perceived, the pose is updated by the perception result using service(UpdatePose, [object_info, result], Object)

  The object_info is a pair of object_id and the new pose of the object. The object_id has to match the response from SpawnObject. The response of UpdatePose is the success state of the service. If the pose of the object is successful the result will be true or false otherwise.

- For the purpose of physical reasoning we have to retrieve the pose of an object from the semDT. This is done by service(GetObjectPose, [object_id, pose])

  The pose of the object with object_id is returned in world coordinates.

- The ability to insert the needle into the septum or the cap is provided by logic(inserting, [ObjectA, ObjectB], [Overlap1, LinearConstraint], [Overlap2, FixedConstraint], Force)

  If Overlap1 is true, meaning ObjectA overlaps with ObjectB, a linear constraint will be activated. This constricts the possible motion directions of ObjectA into the insertion direction. In the case of the needle insertion, this models the expected motion through the septum which is generally constrained by the needle interacting with the solid rubber allowing a certain insertion trajectory. If Overlap2 is true and the volume of ObjectA overlapping with ObjectB surpasses a defined threshold, a fixed constraint will be activated to keep the objects attached. This means of no relative motion is possible until the applied force is larger than Force.

3.4 Physical reasoning based embodied probabilistic simulations

Scene understanding a.k.a. perception in complex environments especially dynamic and human-centered such as in TraceBot ones goes beyond classical tasks such as classification usually known as the what- and where object-questions from sensor data, and poses at least three challenges that are missed by most and not properly addressed by some actual robot perception systems. Note that

Horizon 2020

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

sensors are extrinsically (e.g., clutter, embodiedness-due noise, delayed processing) and intrinsically (e.g., depth of transparent objects) very limited, resulting in a lack of or high-entropy data, that can only be difficultly compressed during learning, difficultly explained or intensively processed during interpretation. (a) Therefore, the perception system should rather reason about the causes that produce such effects (how/why-happen-questions). (b) It should reason about the consequences (effects) of agent object and object-object interactions in order to anticipate (what-happen-questions) the (e.g., undesired) world state and then enable successful action on time. (c) Finally, it should explain its outputs for safety (meta why/how-happen-questions). We introduce therefore a novel white-box and causal generative model of scene understanding (NaivPhys4RP [1]) that emulates human perception by capturing the Big Five aspects (Functionality, Physics, Causality, Intention, Utility) of human commonsense, which recently established, seem to invisibly (dark) drive our observational data and allow us to overcome the above problems. However, NaivPhys4RP particularly focuses on the aspect of physics, which ultimately and constructively determines the world state: hence physical reasoning. Briefly, physical reasoning a.k.a. intuitive physics consists in understanding, even without prior in explicit physics education, the physics governing the behavior of objects in the physical world and then using this knowledge to anticipate and explain physical changes (i.e., object fall, object pose), which then enables successful, safe and smooth action in real-time.

3.4.1 Methodology

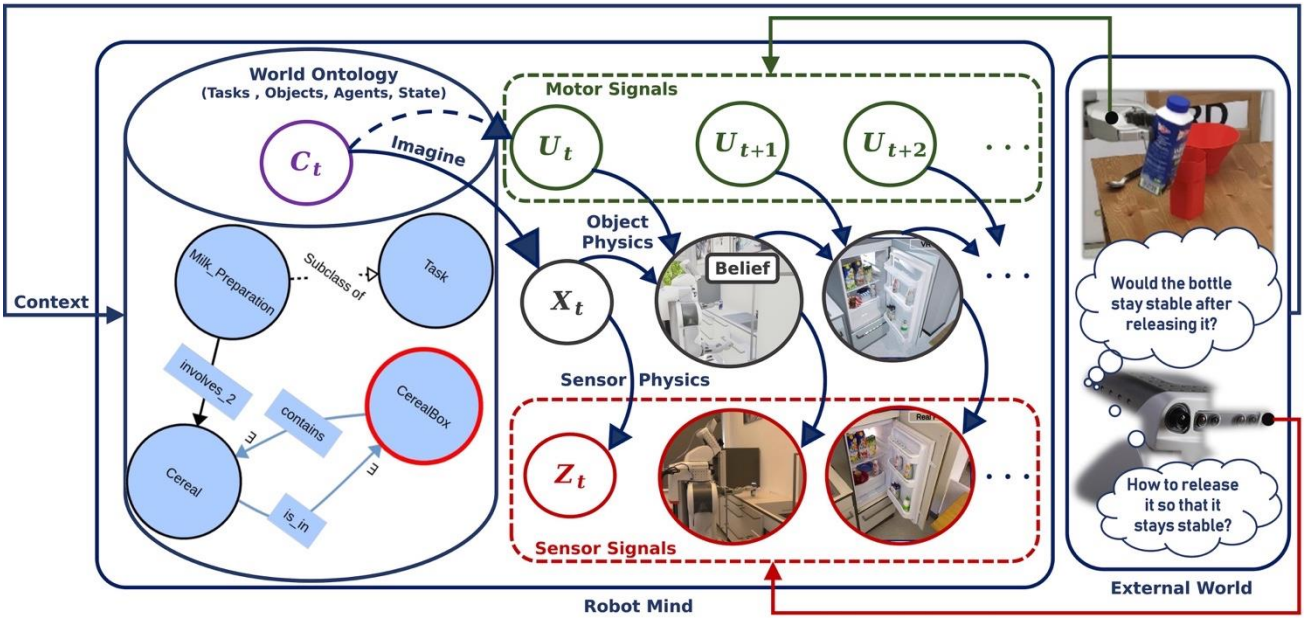

Figure 6: Physical Reasoning based on Embodied Probabilistic Simulations for Understanding of Complex Scenes (e.g., dynamic, human-centered, mission-critical)

We formalize the problem addressed by NaivPhys4RP in four steps. (i) We model the world state, as shown by Figure 6, as a Situated (i.e., take place in a context) Partially-Observable (i.e., only partial

Horizon 2020

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

sensor data) Hidden (i.e., not directly accessible information) Markov Process (i.e., state dependency) (SPOHMP) that evolves through the physics that scene entities (e.g., objects, robots, sensors) undergo. The context is supposed to catch other domains of the commonsense that drive the physics such as intentions, utility and functionality. Actions are already explicitly modeled. (ii) We model the hidden state a.k.a belief of the SPOHMP with the semDT, a photo-realistic and physics-faithful replication of the world grounded in the world ontology for semantics. This makes the internal world representation suitable for emulating the SPOHMP. (iii) Then, we regard perception as taskable through queries and these perceptual queries are clustered into anticipatory (i.e., consequences given causes) and explanatory queries (i.e., causes given consequences), that are abstracted as the bayesian/ markovian inference tasks. However, note that an actual accurate and rich belief of the world state is the informational source for answering these questions. Such a belief is continuously filtered over time through emulation of the SPOHMP. (iv) Finally, we efficiently implement the four main operators of the rao-blackwellized particle filter, however modified to five operators, which is a generic, practical and constructive (i.e., explainability) approach to simultaneously emulate the SPOHMP and address the bayesian inference tasks just mentioned, through embodied, physics- faithful, photo-realistic, probabilistic, partial and ontology-grounded simulations. Notice the genericity of the model as it also considers fundamental physical parameters of the world necessary for useful simulation and commits to estimating them.

This formalization is summarized by the following system of equations (S1):

$$\begin{cases} X_t^* \sim P(X_t | U_{0:t-1}, Z_{0:t}, C_{0:t}) & \text{, actual belief} \\ X_{t+1}^* \sim P(X_{t+1} | U_t, X_t, [C_{t+1}]) & \text{, state anticipation} \\ X_{t+1}^*, U_t^* \sim P(X_{t+1}, U_t | U_{t+1}, C_{t:t+1}, X_t, X_{t+2}) & \text{, state explanation} \\ Z_{t+1}^* \sim P(Z_{t+1} | X_{t+1}) & \text{, observation anticipation} \\ X_{t+1}^* \sim P(X_{t+1} | U_t, X_t, Z_{t+1}, C_{t+1}) & \text{, observation explanation)} \end{cases}$$

- *X*, is the world's hidden state (e.g., a semDT)
- *Z*, is the object/world observation (e.g., RGBD images)
- *U*, is the motion control (e.g., joint values, forces)
- *C*, is the process context (e.g., object, state and task knowledge)

Note that i, t, [.] and ~ respectively denote the particle index, the time index, optional priors and the argmax probabilistic sampling.

3.4.2 Interfaces to physical reasoning

In order to compute the five inference tasks in (S1), we propose the following five generic queries:

- $X_t^* \sim P(X_t | U_{0:t-1}, Z_{0:t}, C_{0:t})$, *actual belief*: In this task, we filter the best possible world state that explains all the past observations given all the past actions and context specifications. Since, we fundamentally rely on particle filter, this is task is computed incrementally and can be reduced to actualizing the current filtered state given new evidence. Hence, the query:

Horizon 2020

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

```
kb_call(
          [
               Instance_of ( A , ActualState  ) ,
               sensor_stream( S1 , sensor_type ) ,
               sensor_stream( S2 , sensor_type ) ,
               motor_stream( M1 , motor_type ) ,
               motor_stream( M2 , motor_type ) ,
               context_stream( C1 , context_type ),
               actualize-state(A,[S1,S2], [M1,M2],[C1],B)
          ]
). B is the target answer.

kb_unproject(
          [
               Instance_of ( A , ActualState  )
          ]
).

kb_project(
          [
               Instance_of ( B , ActualState )
          ]
).
```

- $X^*_{t+1} \sim P(X_{t+1}|U_t, X_t, [C_{t+1}])$, *state anticipation*: In this task, the state of the world given the current agent's action, world state and optionally the future context of the scene is predicted a.k.a. anticipated.

```
kb_call(
          [
               Instance_of ( A , ActualState  ) ,
               motor_stream( M1 , motor_type ) ,
               motor_stream( M2 , motor_type ) ,
               context_stream( C1 , context_type ),
               anticipate-state(A,[M1,M2],[C1],B)
          ]
). B is the target answer.


kb_project(
          [
              Instance_of ( AnticipatedState , B )
```


Horizon 2020

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

```
        ]
).
```

- $X^*_{t+1}, U^*_t \sim P(X_{t+1}, U_t | U_{t+1}, C_{t:t+1}, X_t, X_{t+2})$, *state explanation*: In this task, we are interested in the state and action that lead to a specific state however in a more practical manner (e.g., how should the robot hold the canister or bottle so that by releasing it, it does not fall on the working surface? How should the robot hold the canister and the tube so that by pushing the tube it gets into the pump?, …). Formally, given the actual belief X_t, we are looking for action U_t in the context C_t that would transform X_t into a state X_{t+1} within the context C_{t+1} so that by applying the given action U_{t+1} one could reach the given target state X_{t+2}. Hence the query:

```
kb_call(
        [
            Instance_of ( Xtp2 , TargetState  ) ,
            Instance_of ( Xt , ActualState  ) ,
            Instance_of ( Utp1 , TargetAction  ) ,
            Instance_of ( Ct , ActualContext  ) ,
            Instance_of ( Ctp1 , TargetContext  ) ,
            explain-state(Xt,Ct,Ut,Xtp1,Ctp1,Utp1,Xtp2)
        ]
). Xtp1 and Ut are the target answers.
```

- $Z^*_{t+1} \sim P(Z_{t+1} | X_{t+1})$, *observation anticipation*: In this task, we are interested in how likely Z_{t+1} can be the observation of the state X_{t+1}. Note that this task is fundamentally referred to as imagistic reasoning and is further illustrated in the next section of this document. Fundamentally, this task completes in two steps namely a sensory imagination of the given state and the comparison of this imagination with the given state observation. Hence the queries:

```
For constructing an image of the scene given its description:

kb_call(
        [
            Instance_of ( X , TargetState  ) ,
            Instance_of ( S1 , SensorType1  ) ,
            Instance_of ( S2 , SensorType2 ) ,
            imagine-state(X,[S1,S2],Z)
        ]
). Z is the target answer.

For deciding on whether an observation is the image of a state:

kb_call(
        [
            Instance_of ( X , TargetState  ) ,
            Instance_of ( Z , TargetObservation  ) ,
```


Horizon 2020

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

```
                Instance_of ( S1 , SensorType1  ) ,
                Instance_of ( S2 , SensorType2 ) ,
                imagine-state(X,[S1,S2],Zi)
                similarity ( Z , Zi , D)
        ]
).  D as similarity is returned as answers to the query.
```

- $X^*_{t+1} \sim P(X_{t+1}|U_t, X_t, Z_{t+1}, C_{t+1})$, *observation explanation*: This task addresses the traditional perception problem (i.e. extracting information from sensory data) however in a causal manner which is not only beneficial w.r.t. explainability and therefore safety but also robust against sensor limitations mentioned earlier. Notice that this task can be completed through state, observation anticipation and similarity defined above. Hence the following queries:

```
kb_call(
        [
                Instance_of ( Xt , ActualState  ) ,
                Instance_of ( Ut , ActualAction  ) ,
                Instance_of ( Ctp1 , TargetContext  ) ,
                Instance_of ( Ztp1 , TargetObservation ) ,
                explain-observation(Xt,Ut,Xtp1,Ctp1,Ztp1)
        ]
). Xtp1 is the target answer.
```

3.5 Perception executive

The Perception Executive is a component to enable the robot to observe the environment, extract task-relevant objects and return data, which is informative enough to ultimately be able to specify the necessary motions that the robot has to fulfill in the targeted scenarios.

In TraceBot, we combine state of the art computer vision routines with our semDT, to enable the system to additionally reason on an imagistic level for belief state verification. This poses several challenges, because perception in general is highly variable and task-specific. And additionally, the combination of these computer vision methods with a verification modality requires an additional form of flexible interfacing, because an efficient imagistic verification requires a semantic description to guide the attention to the most relevant aspects to validate based on the preconditions of a task or the desired outcome.

To interface the Perception Executive in the system and cover the described interfacing requirements, we have developed a Perception Task Language(PTL) that is used to describe the type of the different computer vision tasks and their related properties in the TraceBot use cases. In this chapter, we will describe the key aspects of the perception task language in relation to selected subtasks of the sterility testing process.

The perception task language can be roughly divided into detection-based perception tasks (DBPT) or imagination-enabled perception tasks (IEPT). DBPTs encompass perception queries that result in a direct analysis of the observations from the real world cameras of the system, yield the necessary

Horizon 2020

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

information to interact with the non-static objects in the scene and extend the belief state of the robot. Examples for this category are object detection tasks or pose estimation.

In the PTL, such tasks are described in the following form:

```
(detect
 (an object
     (type (canister|bottle|clamp|pump|...)))
```

Every task query begins with an action-based keyword, that describes the type of the perception tasks that has to be executed. A *detect* instructs the system to look for a specific feature or entity in the current scene, based on a semantic description. Take for example the isolated use case in TraceBot, where the robot is about to fit a canister into the drain tray of the pump. The skill plan of the system starts by finding the canister in the camera data of the system, which is implemented by the *LocateAction* ROS Action provided by TUW. In the semantic description of the example task, the system encodes that a specific object of type 'Canister' is looked for.

This allows the vision component to focus on the detection of that specific class and reject detection results that are not relevant for the current query. Semantic descriptions of such tasks will also allow to scale to new perception tasks easily, for example when we need to detect if there is liquid in a bottle or a spatial relation that has to be respected.

IEPT will enable the system to conduct mental imagery[2] by using the semDT as a photorealistic and physics-enabled belief state. This belief state can be rendered to virtual, embodied percepts that serve as a visual representation of the hypothesized world state.

This step is described as:

```
(imagine
 (a scene
     (from sensorX)))
```

where a rendered percept can be computed and returned based on one of the existing sensors in the TraceBot use case environment. RGBD sensors can be modelled in the semDT based on the camera parameters of the real world hardware, like for example the Field of View, to provide comparable sensory inputs. We assume, that the belief state in the semDT is always kept up to date by grounding all perception, reasoning and simulation results immediately into the underlying scene graph representation. As mentioned in chapter 3.3, our simulation-based reasoning is extending the system by subsymbolic functionality, which also includes visual data. By representing the belief state in the semDT, we can use the underlying game engine functionality to render image data from

Horizon 2020

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

given camera models and therefore get a visual representation of the scene, given the hypothesized belief.

In the next step, the rendered percept can be exploited to reason about similarities or discrepancies in the visual domain. The form of such perception tasks are the following:

```
(imagine-match
 (an image (from sensorX) (type real-world))
 (an image (from sensorX) (type imagination))
 (description
    (an object
      (type red-plug)
      (location (on canister)))))
```

The first two parameters are the real world sensor data and the corresponding virtual, rendered percept showing the hypothesized belief state after analyzing the real world from a previous DBPT. Both form the visual basis for the imagination match. To make the comparison effective and task-driven, we can guide the reasoning process by specifying a semantic description of the properties that we want to verify. This will allow us to control the attention of the comparison process to the most important features that need to be checked before or after an action took place.

The IEPT shown above, demonstrates a typical check for a manipulation action in the 'Wetting' step of the TraceBot process. Delicate red plugs are attached to the canister top part to create an overpressure and remove the washing medium from the canisters. In general, the successful outcome of this action would be that a red plug is visible on a canister and has not fallen down from the canister top or has not been properly grasped in the first place.

In the semantic description of the perception task, we define that the comparison shall be focused on an object of type red-plug which should be put onto a canister. Both of these semantic descriptions are grounded in the system's belief state and therefore allow to infer properties like the current pose, appearance features and also the corresponding region in the imagination-based percept. With this background information, the system can infer which of those features can be matched by the available imagination-enabled comparison methods.

With this general formulation, the PTL can also cover IEPT like:

- `Needle in bottle-septum` from the Insert needle subtask in the Needle preparation, Sample transferring or Media filling steps.
- `Needle-cap on needle` from the Remove needle cap subtask in the Needle preparation step.
- `bottle in bottle-holder` from the Move bottle into holder subtask in the Washing and Media filling steps.

Horizon 2020

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

Besides the development of the presented PTL, we have also investigated the necessary extensions to the perception framework to support mental imagery for the TraceBot use cases. This led to an improvement of the process model to support visual multi-hypothesis reasoning and the variability of the imagine-match pipelines. The new process model will allow us to reason about different possible outcomes of actions, like for example a needle cap that has been fully, partially or not removed.

With the new process model, we can also dynamically adapt the processing pipeline for IEPT based on the current task context and the knowledge that exists in the knowledge base for the object the system is focusing on right now.

4 Deviations from the workplan

No major deviation has been detected, and the document has been delivered on time.

5 Conclusion

The goal of this deliverable was the interface specification of the hybrid knowledge representation and reasoning system developed in work package five for the TraceBot project. The description followed a top-down explanation schema. Beginning from a high-level view on how the robot is interacting with the system, the interface descriptions have been embedded into the key building blocks of our system. After highlighting the interaction and the representation based on symbolic descriptions, we have described the reasoning methods that are extending the system subsymbolically with simulation-enabled methods w.r.t the TraceBot use cases.

A key enabler in this respect is the development of the Semantic Digital Twin (semDT), which features a photorealistic and physics-enabled simulation of the TraceBot use case. This component provides an interface and ultimately enables the robot to reason about physical and visual aspects when running traceable processes. The simulation features a rich robot model as well as ways of interacting with the environment representation during the task execution and grounds interactions into the underlying symbolic representation.

This deliverable mainly focused on the realization of the interfaces for the first real world demonstrator, while the overall conceptualization is also based on the requirements of the other subprocesses in the sterility testing process. We will therefore, based on this specification, extend the presented components and interaction means to the following targeted use cases in the TraceBot project.

Horizon 2020

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

6 Annexes

7 References

[1] Franklin Kenghagho Kenfack, Michael Neumann, Patrick Mania, Toni Tan, Feroz Ahmed Siddiky, Rene Weller, Gabriel Zachmann, and Michael Beetz. Naivphys4rp - towards human-like robot perception - physical reasoning based on embodied probabilistic simulation. In IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids), 2022. Accepted for publication.

[2] Patrick Mania, Franklin Kenghagho Kenfack, Michael Neumann, and Michael Beetz. Imagination- enabled robot perception. In International Conference on Intelligent Robots and Systems (IROS), 2021. Best Paper Award on Cognitive Robotics.

[3] Michael Beetz, Daniel Beßler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoglu, and Georg Bartels. Knowrob 2.0 – a 2nd generation knowledge processing framework for cognition-enabled robotic agents. In International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 2018.

[4] Daniel Beßler, Robert Porzel, Mihai Pomarlan, Abhijit Vyas, Sebastian Höffner, Michael Beetz, Rainer Malaka, and John Bateman. Foundations of the socio-physical model of activities (soma) for autonomous robotic agents. In Boyan Brodaric and Fabian Neuhaus, editors, Formal Ontology in Information Systems - Proceedings of the 12th International Conference, FOIS 2021, Bozen-Bolzano, Italy, September 13-16, 2021, Frontiers in Artificial Intelligence and Applications. IOS Press, 2021. Accepted for publication.

[5] Michael Neumann, Andrei Haidu, and Michael Beetz. Urobosim—a simulation-based predictive modelling engine for cognition-enabled robot manipulation.

Horizon 2020

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