Project Acronym:	ACAT
Project Type:	STREP
Project Title:	Learning and Execution of Action Categories
Contract Number:	600578
Starting Date:	01-03-2013
Ending Date:	30-04-2016

Deliverable Number:	D5.5
Deliverable Title:	Software and hardware architecture and integration – Update of D5.3
Type (Internal, Restricted, Public):	PU
Authors :	L. Bodenhagen, D. Chrysostomou, G. Lisca, H. Langer, M. Tamosionaite
Contributing Partners:	SDU, AAU, UoB, UGOE

Contractual Date of Delivery to the EC:	31-08-2015
Actual Date of Delivery to the EC:	07-09-2015

Contents

Executive Summary

This deliverable is an update of the D5.3 and focuses on the changes on the demonstrators since D5.3 has been finalized. All demonstrators are now functional and considered to be in their final state.

The IASSES setup has been subject to several smaller adjustments arising from project needs, while only minor changes were required in the ChemLab scenario. Furthermore, the overall software architecture has been adjusted by introducing an additional compilation step and thereby allowing for a separation of symbolic and subsymbolic processing. In the IASSES scenario this additional compilation step involves a translation from ADTs to skills, where one ore multiple action chunks are matched to a skill and parameters of this skill are extracted from the ADT.

1 Introduction

This deliverable is a brief update of D5.3, Software and hardware architecture and integration. There has been one change occurred to the software structure in ACAT summarized in section 2, while the sections 3 and 4 cover changes on the IASSES and ChemLab scenarios respectively. Updates to the simulation facilities are given in section 5.

2 ACAT Structure

The structure of the ACAT software system has, as indicated on Fig. 1, been only slightly adjusted compared to the first version of the deliverable. In particular the compilation of new instructions has been split into a symbolic part, leading to a blueprint of the new ADT, and a subsymbolic part, where information which is based on sensorial input, e.g. object poses, is added to the ADT prior to execution.

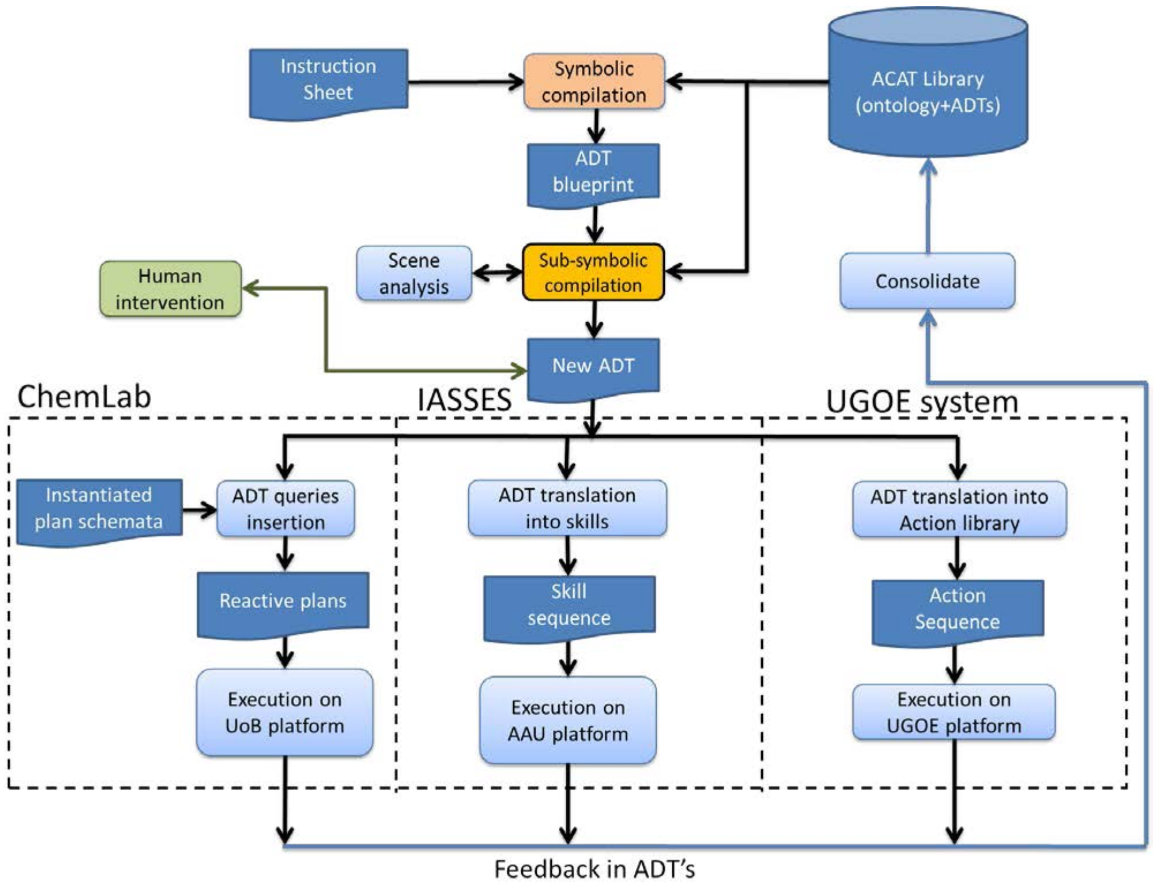


Figure 1: Block diagram providing an overview of the ACAT software system.

3 The IASSES Scenario

In the following the changes to the IASSES hardware setup (section 3.1) and the software components related to the IASSES setup (section 3.2) are summarized.

3.1 Changes to IASSES hardware setup

In the area of hardware integration, several changes occurred in AAU Little Helper in order to meet the demands of the demonstrations. An overview is presented below. As a prior step all the hardware modules have been designed, assembled and integrated in Solidworks to ensure the best physical reproduction of the setup and assist the modeling of a simulation setup. Figure 2 illustrates this modeling.

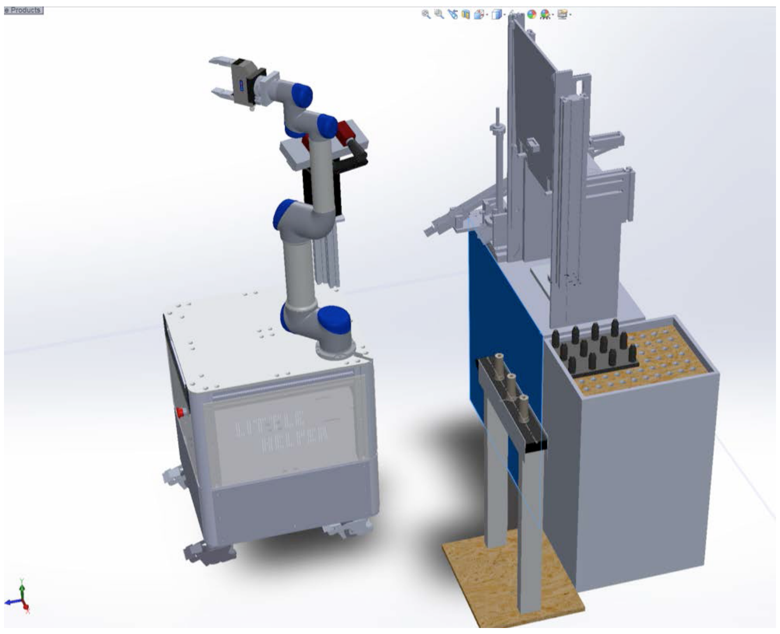


Figure 2: Modeling of the IASSES scenario prior to integration.

In order to comply with the real operational scene from Grundfos, a mockup of the actual press with all the necessary feeders and fixtures along with a mockup of a static conveyor were designed, manufactured, assembled and are now part of the IASSES operational environment. Below, a general overview picture of the robot setup for the IASSES scenario is shown in Figure 3, while the mockup of the conveyor is shown in Figure 4a.

One of the major changes in AAU Little Helper platform is the exchange of the UR5 manipulator with the newest model using the CB3 controller. The exchange was necessary in order to be able to use the improved true absolute encoders of the manipulator alongside with its new software. This technical improvement enables a faster startup of the robot arm. The absolute position is recognized right after the switching-on, eliminating the need for jogging during initialization of the robot arm. Moreover, due to requests from partners the three-finger gripper from Robotic has been exchanged with one made from SCHUNK (model WSG-50), as illustrated in Figure 5a, in order to take advantage of its interchangeable fingers. Finally, a force torque sensor acquired and integrated successfully in the end effector of the UR5 manipulator to assist the integration of force related skills developed in cooperation with partners from JSI (see Figure 5a). As Figure 5 depicts, several

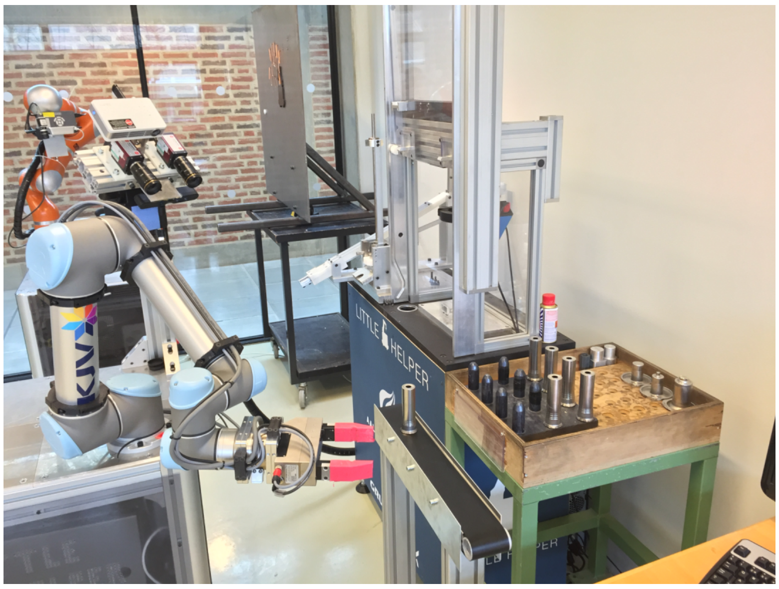


Figure 3: The updated physical robot setup for the IASSES scenario.

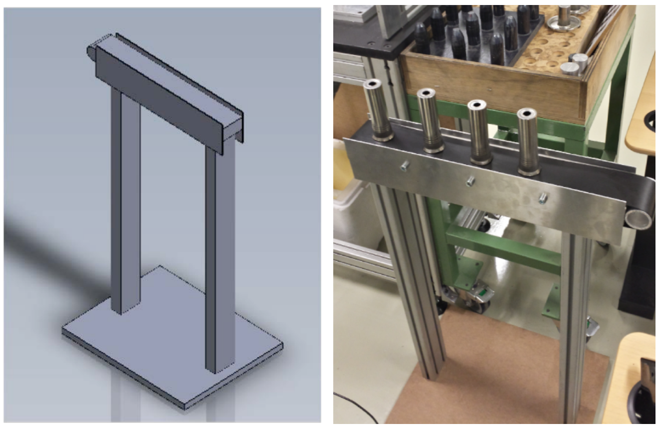
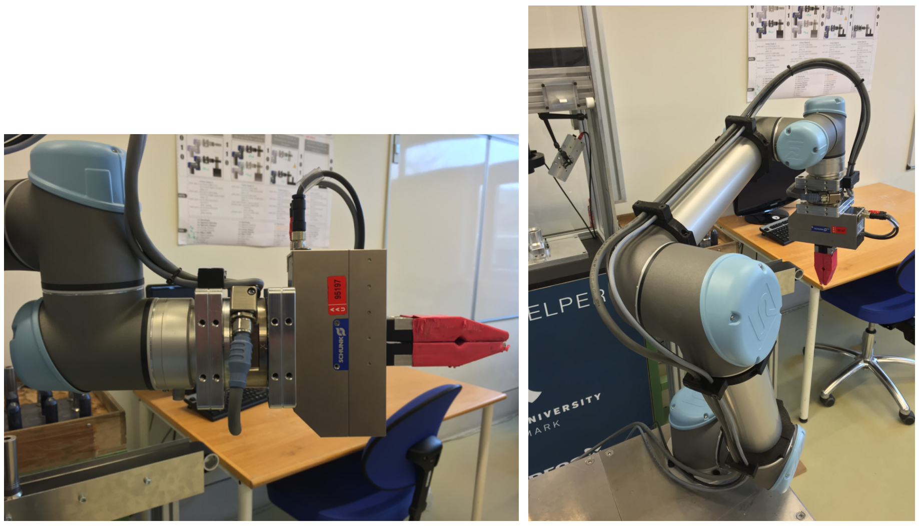

(a) (b)

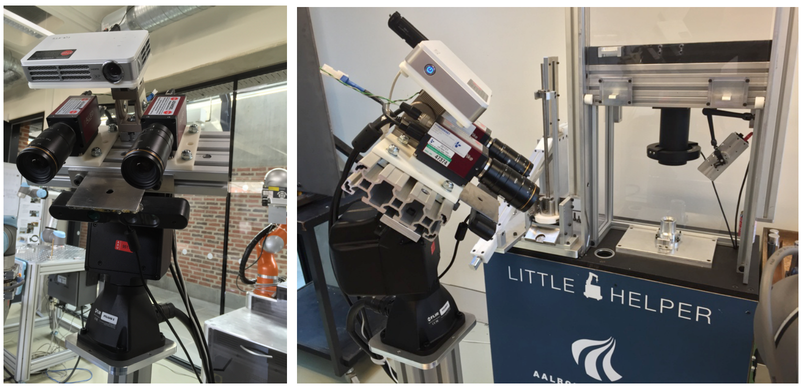
Figure 4: The virtual (a) and physical (b) mockup of the conveyor for the needs of the IASSES scenario

auxiliary metal plates had to be designed, manufactured and attached in order to facilitate the installation of the F/T sensor and the gripper. Besides, a sophisticated way for cable management has been adopted by mounting the necessary cables into circular cable holders alongside the body of the manipulator. Such cable setup allows the manipulator to approach the IASSES objects without interfering with them and enables movements of the end effector without the risk of damaging the cable connectors. The 3D printed black cable holders are illustrated in the Figure 5b. Finally, the vision system of the Little Helper platform is now finalized as shown in Figure 6.



(a) (b)

Figure 5: The force torque sensor attached between the end effector and the SCHUNK gripper; The 3D printed cable holders to ensure minimum interference of the cables with the execution.



(a) (b)

Figure 6: The IASSES Vision system consists of a pan tilt unit, a depth camera, a stereo camera system and a projector.

3.2 Changes to IASSES Software Architecture

The exchange of the UR5 manipulator to the newest model enabled us access to enhanced teaching functionalities and therefore to be able to develop and create an ADT translator into robot skills. Besides, the development of novel drivers for the SCHUNK gripper, the force torque sensor and the preliminary driver for the pan tilt unit the connections altered the connections between the different components (illustrated in Figure 7). As far as the robot-camera calibration is concerned, a software system developed by AAU in the TAPAS project was adapted in order to calibrate successfully the robot manipulator in relation with the position of the cameras on the vision pole and the IASSES scene.

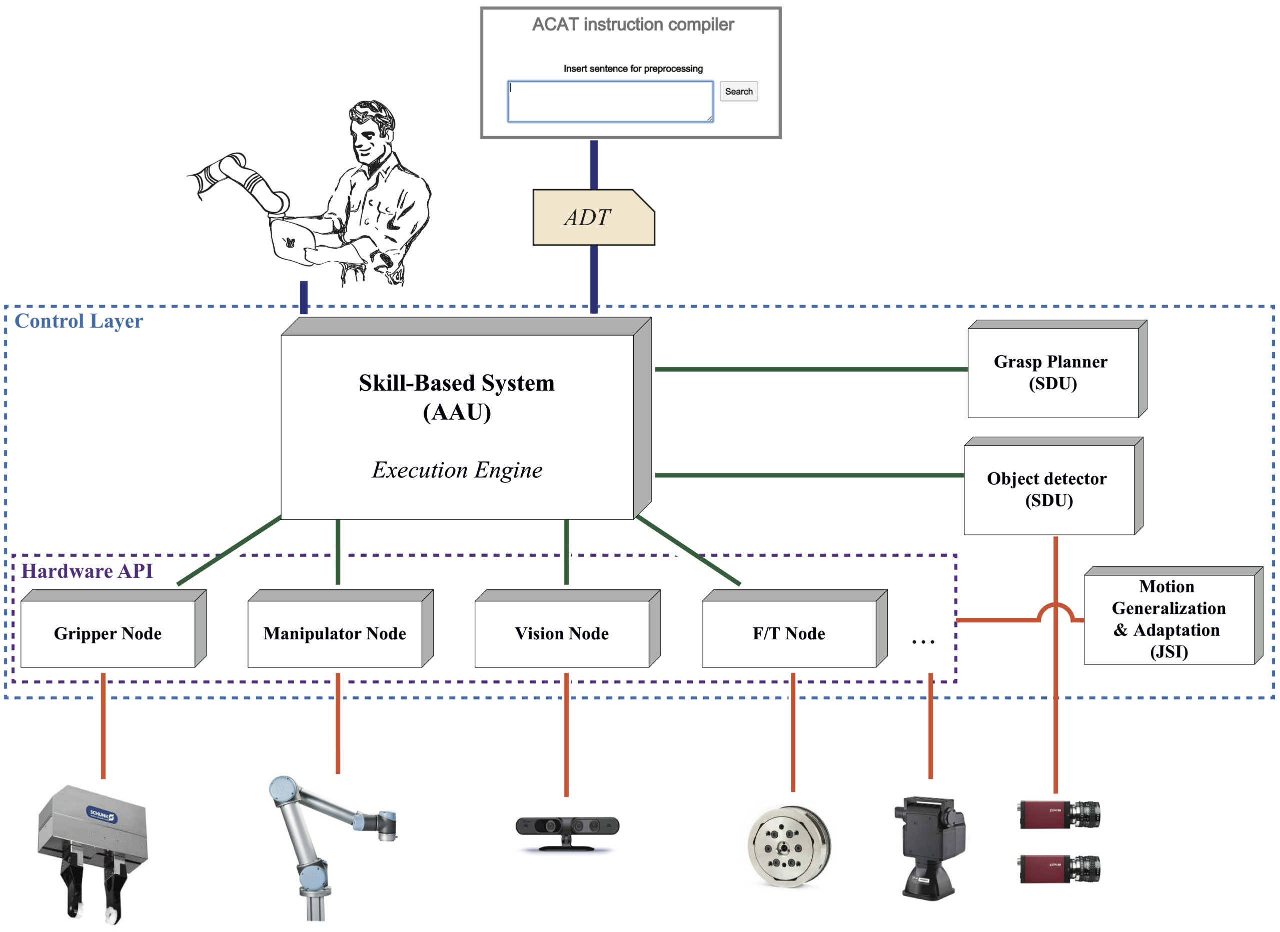


Figure 7: The updated connections between the components of the IASSES scenario

3.2.1 ADT translator in the IASSES scenario

In order to have a holistic approach towards the demonstration of the IASSES scenario, we developed an execution engine tailored for Little Helper and its Skill Based System. The initial instruction under consideration in the IASSES scenario is 'Pick a rotor cap from conveyor and put it on the fixture'. This instruction is divided into two action primitives namely 'pick up' and 'put down'. As Figure 8 below illustrates four action chunks are produced from further analysis of the aforementioned action

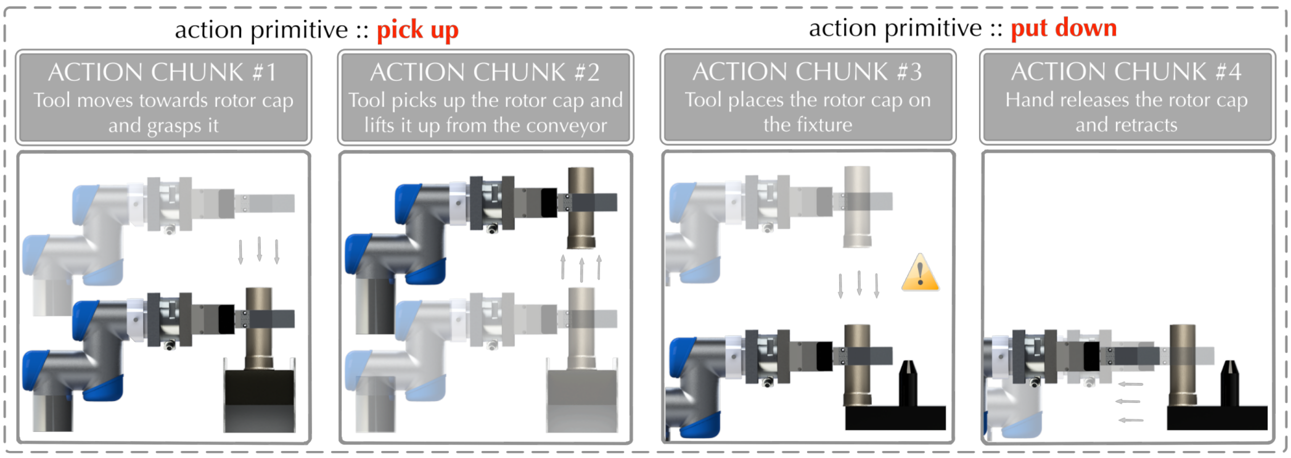

action primitive :: pick up
action primitive :: put down
ACTION CHUNK #1
Tool moves towards rotor cap and grasps it
ACTION CHUNK #2
Tool picks up the rotor cap and lifts it up from the conveyor
ACTION CHUNK #3
Tool places the rotor cap on the fixture
ACTION CHUNK #4
Hand releases the rotor cap and retracts

Figure 8: Action primitives and action chunks involved in the instruction 'pick up rotor cap from conveyor and put it on the fixture'.

primitives.

The aforementioned relationships between the parameters of the skills and the values of the components of the ADTs are located in the core of the software module built for the Little Helper platform. In order to be easier for the operator to interact with them a user interface has been created. As illustrated in Figure 9, the user is able to load the ADT produced from the symbolic compilation. Later one can obtain all the raw data from the ADT at first and verify the correct interpretation into the parameters required from the skills (Figure 10).

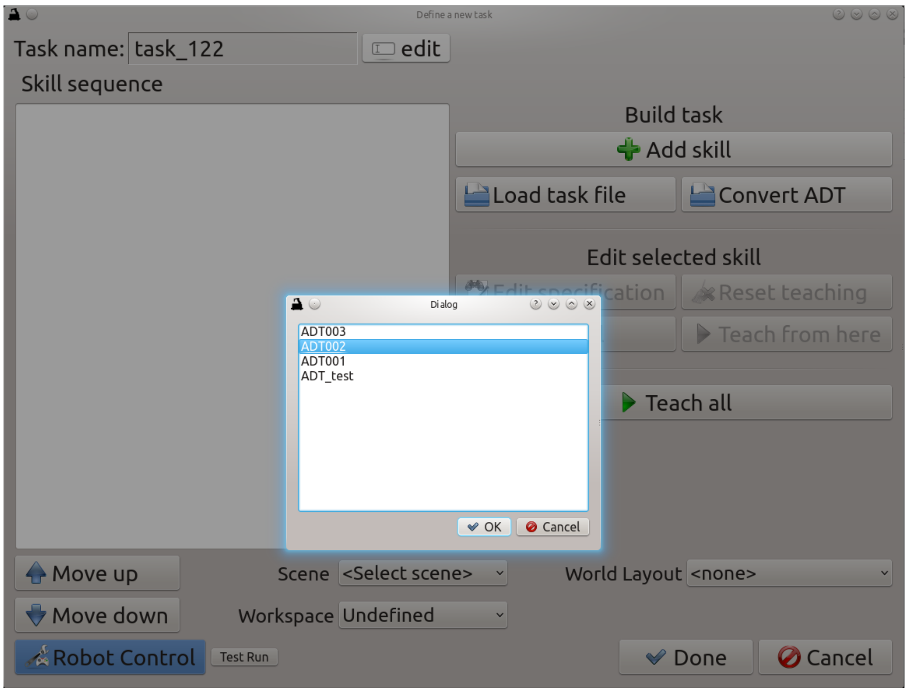
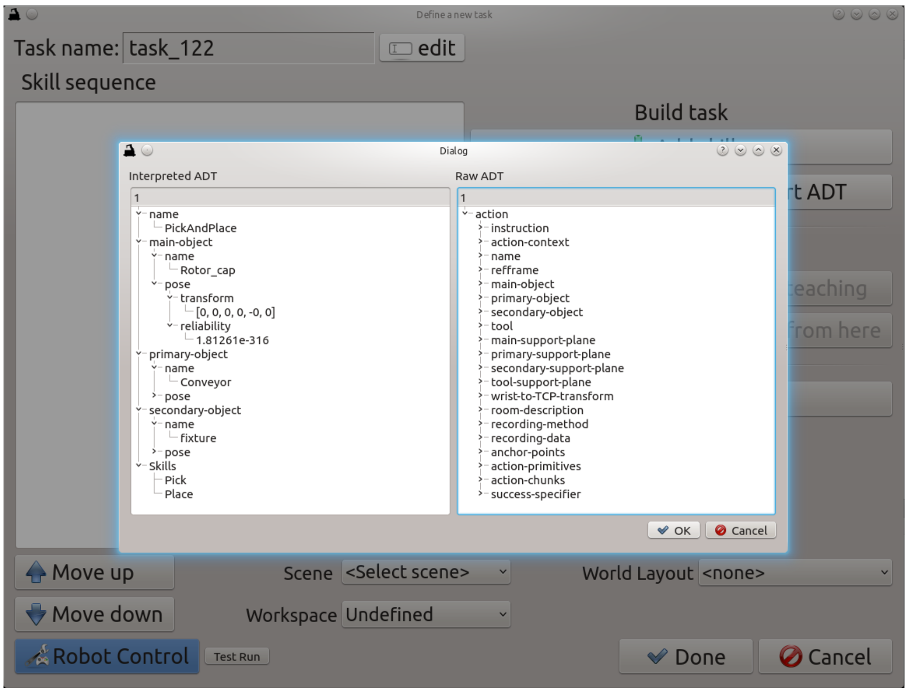


Figure 9: User interface for the ADT translator of the little helper system. The operator, in this first step, can load and convert the ADT produced by the symbolic compilation.



Task name: task_122

Skill sequence

Scene <Select scene>

World Layout <none>

Workspace Undefined

Interpreted ADT

```
1
name
  PickAndPlace
main-object
  name
    Rotor_cap
  pose
    transform
      [0, 0, 0, 0, -0, 0]
    reliability
      1.81261e-316
primary-object
  name
    Conveyor
  pose
secondary-object
  name
    fixture
  pose
Skills
  Pick
  Place
```

Raw ADT

```
1
action
  instruction
  action-context
  name
  refframe
  main-object
  primary-object
  secondary-object
  tool
  main-support-plane
  primary-support-plane
  secondary-support-plane
  tool-support-plane
  wrist-to-TCP-transform
  room-description
  recording-method
  recording-data
  anchor-points
  action-primitives
  action-chunks
  success-specifier
```

Figure 10: The operator can obtain all the raw data from an ADT and then verify how the translator interpreted them into the skills..

3.2.2 IASSES Vision Software

The assembly task in the IASSES scenario imposes a difficult problem for the object recognition since most of the objects are metal made, hence shiny and with little to none texture. By utilizing both a carmine sensor, providing low-resolution 3D point clouds and a high resolution stereo rig combined with a pattern projector it was still possible to achieve sufficiently high recognition rates and performance. In particular low-resolution point clouds have been used to locate larger relevant structures, e.g. the conveyor belt, while the high resolution point clouds are then used to detect objects in the relevant part of the scene. Furthermore, different features have been merged to handle to handle also object with very limited shape information [3]. To cope with the limited field of view the both camera systems are mounted on a pan-tilt unit. Further details on the recognition system will be be provided in D4.3, *Scene analysis and movement description – Update of D4.1*.

3.2.3 IASSES Grasp planner software

The grasp planning as such was reported upon in D5.3 and has been adjusted to changes on the setup during the project continuously. Furthermore, the interface has been extended to cover not only gripper poses but also collision free paths by utilizing sensor readings rather than solely relying on a static scene model. Thereby, changes in the scene such as objects being transported by a conveyor are also reflected during the motion planning.

4 The ChemLab scenario

Changes in the ChemLab scenario are outlined in the following, focusing on the setup in section 4.1 and on the related software in section 4.2.

4.1 Changes to the ChemLab setup

The hardware platform used in the ACAT ChemLab scenario is basically the same Willow Garage PR2 robot that has already been described in the original version of this deliverable. Since then, there have been several hardware and software updates:

- the robot is now equipped with a kinect2 sensor system
- an additional backpack computer has been installed for running the kinect2
- the PR2 has now a wrist-mounted force/torque sensor
- currently, the robot is running with Ubuntu 12.04 and ROS Hydro, another update (Ubuntu 14.04, ROS Indigo) is planned for the near future

Data logged from the DNA extraction experiment are available from our open web-based knowledge service openEASE[1], while a detailed description of the ChemLab scenario has been presented by Lisca et al. [4].

4.2 Update on the ChemLab Software Architecture

Probabilistic Robot Action Core (PRAC) models [5], representing real world event probabilistically, as well as structured reactive controllers (SRC) which can be executed by the CRAM execution engine [1] have been reported on in D5.3.

At execution time the SRC's control routines query the KnowRob [6] knowledge processing system in order to retrieve the specific knowledge required by the SRC's control routines. ChemLab scenario uses the knowledge stored in ADTs trough KnowRob by importing them in KnowRob. The subsection dedicated to ADT execution will provide more details about this process.

4.2.1 PRAC in openEASE

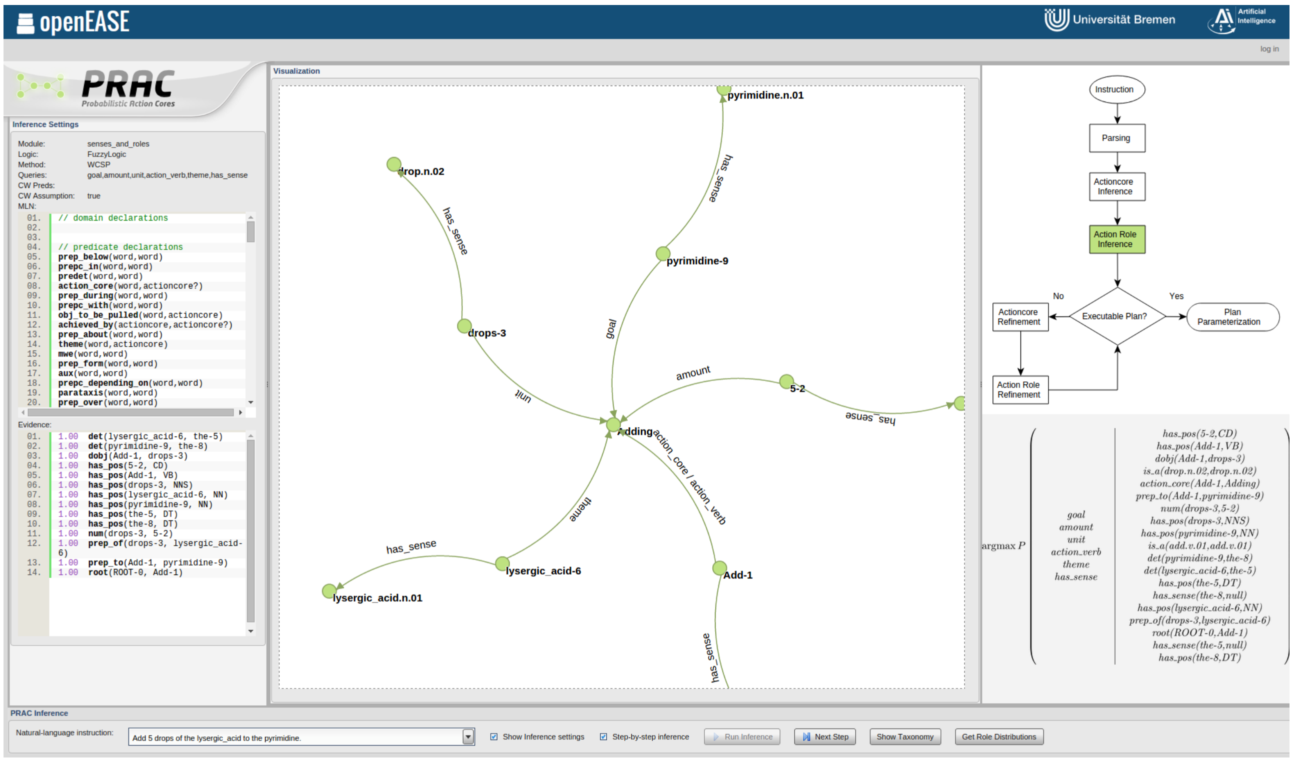
UoB has developed a web service which allows users to generate a robot-executable CRAM plan from natural-language instructions. The service is available on openEASE[2] and provides a visualization of the executed inference pipeline.

The user can choose a natural-language instruction from a set of predefined examples or formulate their own. The inference process can be investigated step by step and the results of each step can be followed in a graph visualization. The visualization provides information about the relations between the terms in the natural-language instruction and their respective inferred meaning and role as well as the action cores (cf. Figure 11 middle) .

In each inference step the conditional probability $P(Q|E)$ is determined, solving for Q, (which is e.g. an action core or the action roles of the instruction terms) and takes as evidence E the result of the predecessor step. An example for such a conditional probability is shown in Figure 12.

[1]http://www.open-ease.org/robotic-agent-performing-chemical-experiments-overview/

[2]OpenEASE is a web-based knowledge service providing robot and human activity data (www.open-ease.org).



Module: senses_and_roles
Logic: FuzzyLogic
Method: WCSP
Queries: goal,amount,unit,action_verb,theme,has_sense
CW Preds:
CW Assumption: true
MLN:

```
// domain declarations


// predicate declarations
prep_below(word,word)
prepc_in(word,word)
predet(word,word)
action_core(word,actioncore?)
prep_during(word,word)
prepc_with(word,word)
obj_to_be_pulled(word,actioncore)
achieved_by(actioncore,actioncore?)
prep_about(word,word)
theme(word,actioncore)
mwe(word,word)
prep_form(word,word)
aux(word,word)
prepc_depending_on(word,word)
parataxis(word,word)
prep_over(word,word)
```

Evidence:

```
1.00  det(lysergic_acid-6, the-5)
1.00  det(pyrimidine-9, the-8)
1.00  dobj(Add-1, drops-3)
1.00  has_pos(5-2, CD)
1.00  has_pos(Add-1, VB)
1.00  has_pos(drops-3, NNS)
1.00  has_pos(lysergic_acid-6, NN)
1.00  has_pos(pyrimidine-9, NN)
1.00  has_pos(the-5, DT)
1.00  has_pos(the-8, DT)
1.00  num(drops-3, 5-2)
1.00  prep_of(drops-3, lysergic_acid-
6)
1.00  prep_to(Add-1, pyrimidine-9)
1.00  root(ROOT-0, Add-1)
```

$$\operatorname{argmax} P\left(\begin{array}{c|c} & has\_pos(5\text{-}2,CD) \\ & has\_pos(Add\text{-}1,VB) \\ & dobj(Add\text{-}1,drops\text{-}3) \\ & is\_a(drop.n.02,drop.n.02) \\ & action\_core(Add\text{-}1,Adding) \\ & prep\_to(Add\text{-}1,pyrimidine\text{-}9) \\ goal & num(drops\text{-}3,5\text{-}2) \\ amount & has\_pos(drops\text{-}3,NNS) \\ unit & has\_pos(pyrimidine\text{-}9,NN) \\ action\_verb & is\_a(add.v.01,add.v.01) \\ theme & det(pyrimidine\text{-}9,the\text{-}8) \\ has\_sense & det(lysergic\_acid\text{-}6,the\text{-}5) \\ & has\_pos(the\text{-}5,DT) \\ & has\_sense(the\text{-}8,null) \\ & has\_pos(lysergic\_acid\text{-}6,NN) \\ & prep\_of(drops\text{-}3,lysergic\_acid\text{-}6) \\ & root(ROOT\text{-}0,Add\text{-}1) \\ & has\_sense(the\text{-}5,null) \\ & has\_pos(the\text{-}8,DT) \end{array}\right)$$

Natural-language instruction: Add 5 drops of the lysergic_acid to the pyrimidine.

Figure 11: PRAC in openEASE

The progress of the inference pipeline (cf. Figure 13) is represented through a flowchart in which the current step is highlighted.

Additional information about the inference method, evidence and knowledge base (Markov Logic Network) used for the current step can be found in the inference settings panel.

The action parameters will be refined until they are no longer underspecified and robot-executable action plans can be instantiated by parameterizing generic plan schemata with the inferred information (cf. Listing 1).

```
(use-pipette (from (an object (type container.n.01)
                              (contains (some stuff (type lysergic\_acid.n.01)))
             (to (an object (type container.n.01)
                            (contains (some stuff (type pyrimidine.n.01)))
             (amount (a quantity (type drop.n.02)
                                 (number five.n.01))))
```

Listing 1: Generated plan from the example instruction Add 5 drops of the lysergic_acid to the pyrimidine.

4.2.2 Chemical Experiments in openEASE

A plan schema is instantiated after PRAC finishes refining plan's parameters such that they are fully specified. As a result of plan schema instantiation a specific plan is identified as the most appropriate plan which is able to make the robot perform the actions meant by the natural language

$$\operatorname{argmax} P\left(\begin{array}{c|c} \text{action\_core} & \begin{array}{c} \text{has\_pos(5-2,CD)} \\ \text{has\_pos(Add-1,VB)} \\ \text{det(pyrimidine-9,the-8)} \\ \text{has\_sense(the-5,null)} \\ \text{has\_pos(lysergic\_acid-6,NN)} \\ \text{prep\_to(Add-1,pyrimidine-9)} \\ \text{num(drops-3,5-2)} \\ \text{dobj(Add-1,drops-3)} \\ \text{has\_pos(drops-3,NNS)} \\ \text{has\_pos(pyrimidine-9,NN)} \\ \text{is\_a(add.v.01,add.v.01)} \\ \text{det(lysergic\_acid-6,the-5)} \\ \text{has\_pos(the-5,DT)} \\ \text{has\_sense(the-8,null)} \\ \text{prep\_of(drops-3,lysergic\_acid-6)} \\ \text{root(ROOT-0,Add-1)} \\ \text{has\_pos(the-8,DT)} \end{array} \end{array}\right)$$

Figure 12: Conditional probability solving for the action core of the instruction given the syntactic structure of the sentence and the respective word senses.

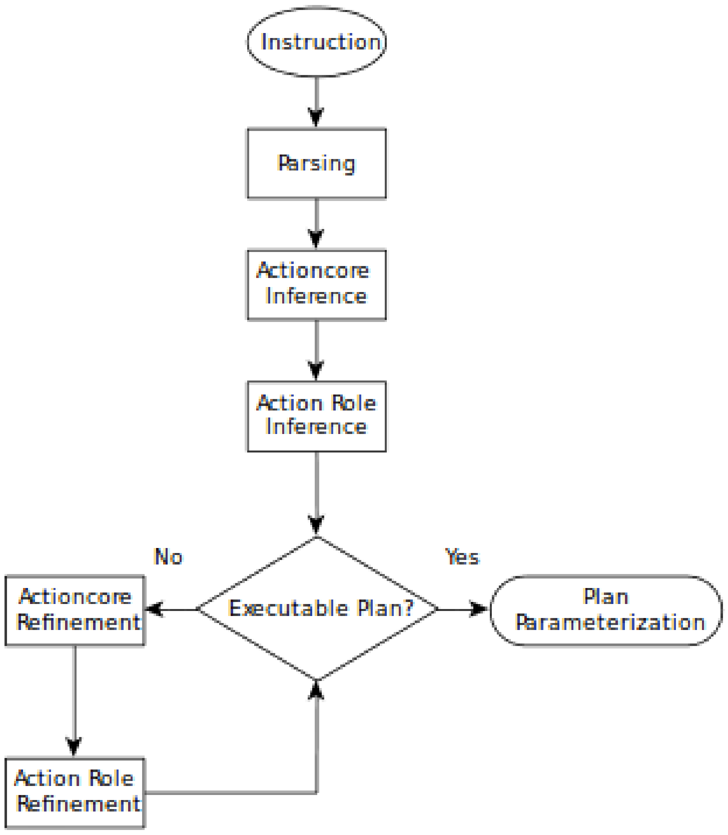


Figure 13: Flowchart indicating the progress of the inference pipeline.

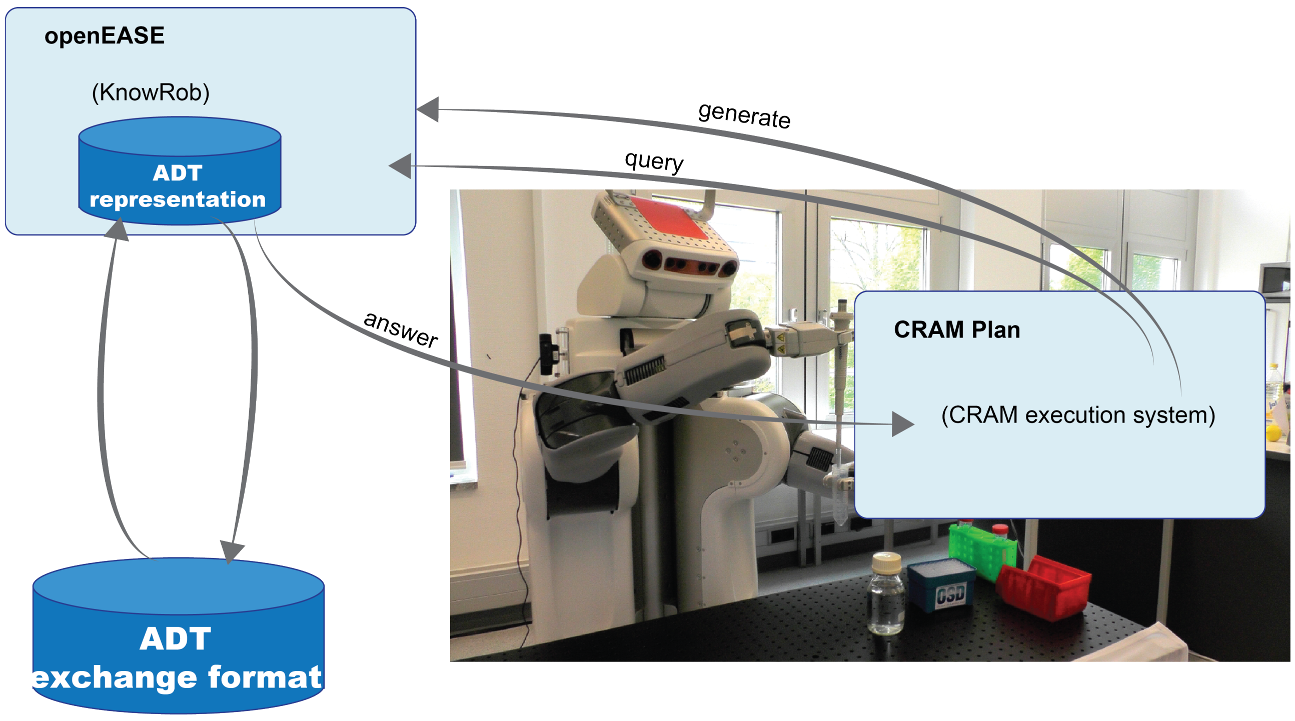


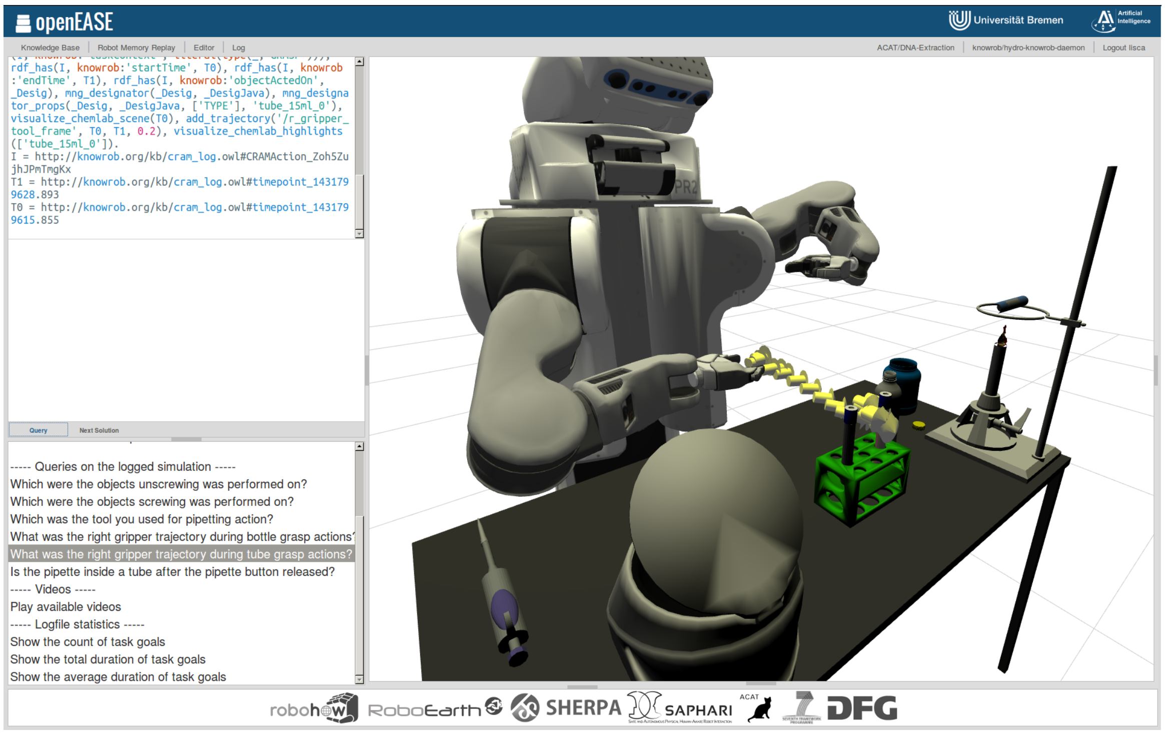
Figure 14: PR2 interacting (trough openEASE) with an ADT represented in KnowRob.

instruction. This plan has his parameters fully specified and it's ready to run in CRAM Execution Engine - in our case the *pipet* plan.

Internally the *pipet* plan calls other plans like *aspirate*, *dispense*, *screw*, *unscrew* (cf. Figure. 14). When the *unscrew* plan runs it calls other plans like *grasp* and different routines for retrieving from KnowRob various specific knowledge about the manipulations which the robot is instructed to perform. For instance the *unscrew* plan expects that the *unscrewing-trajectory* is stored into the *unscrew* ADT hence trough *query-ADT* routine it queries KnowRob for the *unscrewing-trajectory*. If the *unscrewing-trajectory* is returned then *unscrew* plan simply calls the *execute* routine for executing it. Contrary if the *unscrew* ADT doesn't contain the *unscrewing-trajectory* the *unscrew* plan calls the *compute-unscrewing-trajectory* routine which employs different interpolators which compute the *unscrewing-trajectory*. Furthermore this trajectory is forwarded to the *execute-trajectory* routine which simply executes it. Lastly, before returning, the *unscrew* plan saves the freshly computed *unscrewing-trajectory* into the *unscrew* ADT by calling the *update-ADT* control routine. While the *unscrew* plan is running the CRAM Execution Engine is logging various aspects of the execution process. All those aspects are semantically annotated. Once imported in *openEASE* these logs enable our robot and other robots to either reuse them or learn from them. For instance other robot required to grasp a test tube hold in a rack could query *openEASE* (cf. Figure 15) how to perform the reaching motion: "What was the right gripper trajectory during tube grasp actions?". As answer *openEASE* will return the trajectory of our robot's right gripper while reaching for the test tube.

```
(def-plan unscrew (object-part object-body)
 (grasp object-body)
 (grasp object-part)
 (let (unscrewing-trajectory (query-ADT "unscrew" object-part object-body))
  (if unscrewing-trajectory
   (executei-trajectory unscrewing-trajectory)
   (progn
    (let ((computed-trajectory (compute-unscrewing-trajectory
                                object-part object-body))
     (execute-trajectory computed-trajectory)
     (update-ADT "unscrew" computed-trajectory))))))
```

Listing 2: The *unscrew* plan.



```
rdf_has(I, knowrob:'startTime', T0), rdf_has(I, knowrob
:'endTime', T1), rdf_has(I, knowrob:'objectActedOn',
_Desig), mng_designator(_Desig, _DesigJava), mng_designa
tor_props(_Desig, _DesigJava, ['TYPE'], 'tube_15ml_0'),
visualize_chemlab_scene(T0), add_trajectory('/r_gripper_
tool_frame', T0, T1, 0.2), visualize_chemlab_highlights
(['tube_15ml_0']).
I = http://knowrob.org/kb/cram_log.owl#CRAMAction_Zoh5Zu
jhJPmTmgKx
T1 = http://knowrob.org/kb/cram_log.owl#timepoint_143179
9628.893
T0 = http://knowrob.org/kb/cram_log.owl#timepoint_143179
9615.855
```

----- Queries on the logged simulation -----
Which were the objects unscrewing was performed on?
Which were the objects screwing was performed on?
Which was the tool you used for pipetting action?
What was the right gripper trajectory during bottle grasp actions?
What was the right gripper trajectory during tube grasp actions?
Is the pipette inside a tube after the pipette button released?
----- Videos -----
Play available videos
----- Logfile statistics -----
Show the count of task goals
Show the total duration of task goals
Show the average duration of task goals

Figure 15: Chemlab in openEASE

5 Simulation platform

The simulation engine, facilitated at SDU, is developed continuously. Recently the simulator has been enhanced to simulate more complex actions, composed by several smaller action chunks. These actions resemble the actions found on instruction sheets as they are used in the IASSES scenario. The simulation of these actions facilitates the learning structural knowledge [2], considered being background knowledge. Utilizing the structural knowledge associated with a task enables the system to tune the algorithms to the task at hand, e.g. by limiting the search space for object poses during object recognition ((Figure 16) shows results when a line constraint is learned online). By tailoring the functionality to the specific task both speed and robustness is improved.

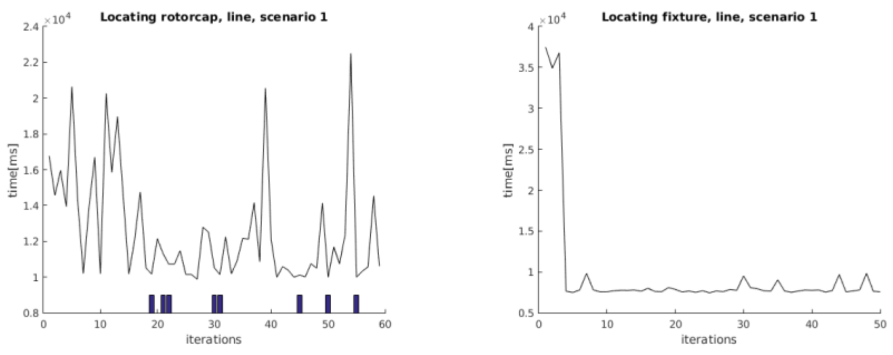


Figure 16: Execution times for pose estimation of rotor caps (left) and spots on the fixture (right) when learning and applying a line constraint. Errors are indicated by boxes on the horizontal axis.

6 Conclusion

In this deliverable, we have provided a summary of the changes related to the soft- and hardware architecture in ACAT. All platforms are now operational and have been adjusted based on the experience made throughout the project.

References

[1] Michael Beetz, Lorenz Mösenlechner, Moritz Tenorth, and Thomas Rühr. Cram – a cognitive robot abstract machine. In *5th International Conference on Cognitive Systems (CogSys 2012)*, 2012.

[2] Jimmy Alison Jørgensen, Nadezda Rukavishnikova, Norbert Krüger, and Henrik Gordon Petersen. Spatial Constraint identification of parts in SE3 for action optimization. In *IEEE International Conference on Industrial Technology (ICIT)*, 2015.

[3] L. Kiforenko, A. Glent Buch, L. Bodenhagen, and N.Krüger. Object Detection using Categorized 3D Edges. In *Seventh International Conference on Machine Vision (ICMV)*, 2014.

[4] Gheorghe Lisca, Daniel Nyga, Ferenc Balint-Benczedi, Hagen Langer, and Michael Beetz. Towards robots conducting chemical experiments. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015 (accepted).

[5] Daniel Nyga and Michael Beetz. Everything robots always wanted to know about housework (but were afraid to ask). In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October, 7–12 2012.

[6] Moritz Tenorth and Michael Beetz. KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. *International Journal of Robotics Research (IJRR)*, 32(5):566 – 590, April 2013.