SEVENTH FRAMEWORK PROGRAMME

Project Acronym:	ACAT
Project Type:	STREP
Project Title:	Learning and Execution of Action Categories
Contract Number:	600578
Starting Date:	01-03-2013
Ending Date:	28-02-2016

Deliverable Number:	D5.3
Deliverable Title:	Software and hardware architecture and integration.
Type (Internal, Restricted, Public):	PU
Authors :	L. Bodenhagen, J. A. Rytz, D. Chrysostomou, H. Langer, M. Tamosionaite
Contributing Partners:	SDU, AAU, UoB, UGOE

Contractual Date of Delivery to the EC:	03-03-2014
Actual Date of Delivery to the EC:	03-03-2014

Contents

Executive Summary

This deliverable provides an overview of the envisioned overall structure in ACAT as well as the three different demonstration platforms:

- the IASSES scenario and the Little Helper at AAU
- the ChemLab scenario at UoB
- a simulation platform at SDU

A description of the hard- and software architecture for these three platforms is provided in this document. The real-world platforms will utilize the ACAT structure directly to solve given tasks including novel ones, while the simulation platform also will be used for the generation of background knowledge directly as it allows for the massive exploration of potential solutions for task.

The three individual platforms are ready to operate, or close to being so, but will also be extended or modified in the future. The overall structure is currently under development and a more elaborated description of the finalized structure and its integration will be provided with D5.5 in month 30.

1 Introduction

This document describes the software and hardware architecture and integration. The deliverable will provide a report on the current status, methods and structures for software and hardware architecture as well as the status of the integration for the two scenarios.

We will introduce the ACAT system architecture and describe how the Action Kernels interface to the three different platforms in ACAT. This interface relates both to the execution of action kernel sequences and to the consolidation and incorporation of sensor motor feedback formatted in Action Data Tables (ADT's). The ADT's are described in deliverable D2.1.

2 ACAT Structure

Figure 1 shows an overview of the ACAT software system which has a general part and three platform specific parts. The general part concerns

- extraction of action kernel sequences from an instruction sheet and the ACAT database
- consolidation of sensor motor data received from different platforms

The general part of the ACAT system interfaces to the platform specific parts through:

- Formal Instruction Representation - that is sequences of parametrized action kernels.
- Action Data Tables - that is the feedback from executing action kernels and the encoded sensori motor data and SEC's (see deliverable D2.1).

The specific architectural parts are made up of three different platforms. Two of these are the ACAT integration platforms that are developed at AAU (IASSES scenario) and UoB (ChemLab scenario). The third platform is a simulated platform that enables execution and therefore collection of background knowledge, in other simulated environments. The three architectures are vastly different in design but are all able to interface to and benefit from the general ACAT data-structure.

2.1 Generalization representations

The ACAT system is designated for executing novel actions, hinted to a robot by a human readable instruction sheet which provides only very limited information. The robot needs to obtain missing information from the ACAT database (ACAT DB, See Fig. 1 top part). The ACAT DB will be structured in kernels, representing action verb categories.

The first entity that needs to be extracted from the database, given an instruction from the instruction sheet, is the action kernel corresponding to the action verb spelled in the instruction. However, in order to generalize when executing an (potentially novel) action, the so called background information is required. This includes information that is not explicitly stated in the instruction itself and does not have a straightforward association with action verb spelled in the instruction, but can be extracted from textual ontologies or previous robot execution of related actions. The organization of the background data structures for the ACAT project was described in D2.1. As the generalization process based on ACAT DB is among the research topics of the project, exact software architecture for the top part of the diagram in Fig. 1 is currently under development and will be presented in the follow-up deliverable D5.5 in month 30.

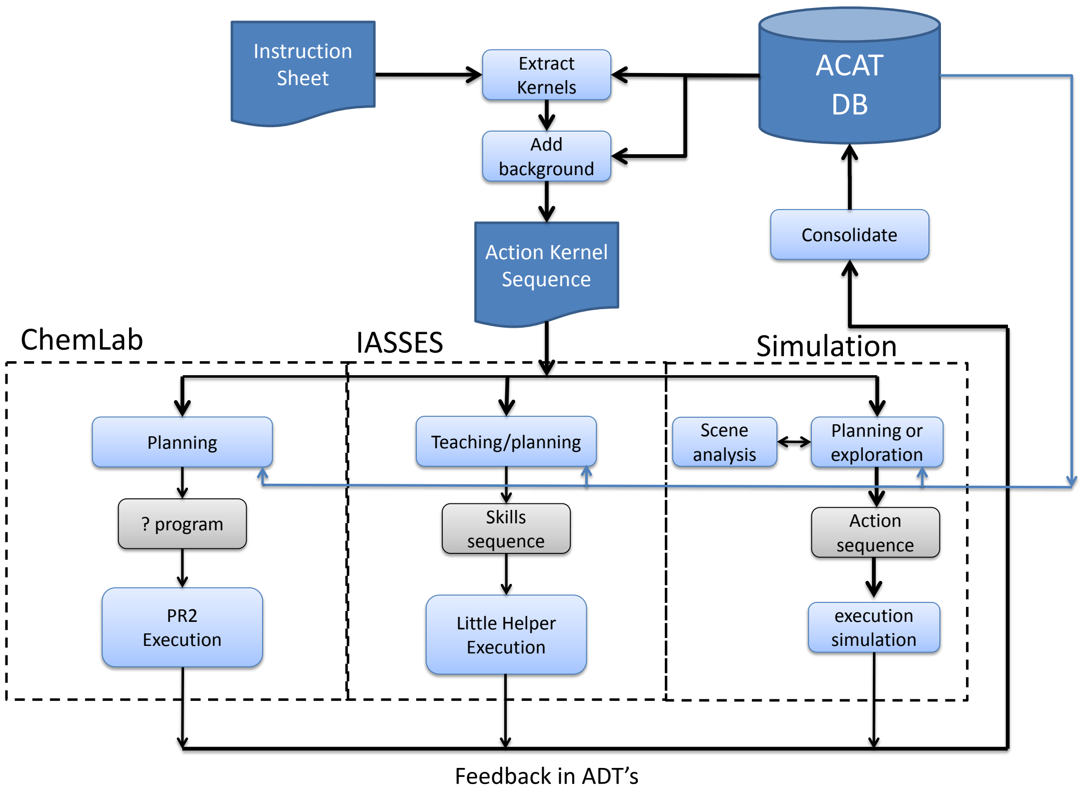


Figure 1: Block diagram providing an overview of the ACAT software system.

3 Little Helper in IASSES

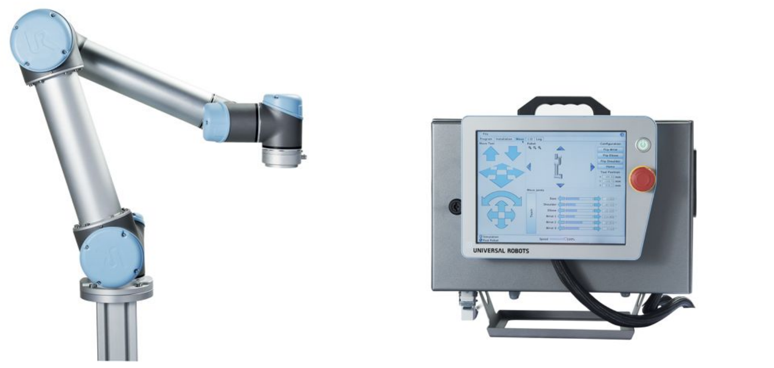
Little Helpers are a family of robots, built and maintained by the Robotics and Automation Group at Aalborg University. The latest member of the family, called Little Helper 4, has been built specifically for the need of the IASSES scenario of the ACAT project and includes a lightweight industrial manipulator, an adaptive 3-finger gripper, a hybrid stereo vision system and a fixed re-configurable platform. In the sections below, the general approaches regarding the software and hardware architecture of the developed system are presented.

3.1 Hardware architecture

This section introduces the hardware that is available for the successful completion of the IASSES scenario. Little Helper 4 is equipped with a UR5 manipulator, a 3-finger gripper by Robotiq, a reconfigurable platform and a mounting system that supports the pan tilt unit with the hybrid stereo vision system, the 3D depth sensor and the laser projector.

3.1.1 UR5 Manipulator

The UR5 manipulator from Universal Robots, illustrated in Figure 2a, is a 6-axis articulated industrial manipulator, with a payload of 5.0 kg, range of 850 mm and weight of 18.4 kg. The teach pendant contains the *Polyscope* graphical user interface software made specifically by Universal Robots offering an intuitive programming interface. The teach pendant and the UR robot controller are depicted in Figure 2b. The UR5 robot arm serves as the main manipulator of Little Helper 4 for the IASSES scenario.



(a) The UR5 robot manipulator (b) The teach pendant mounted on the UR's robot controller.

Figure 2: UR5 manipulator by Universal Robots, its teach pendant, and robot controller

3.1.2 Robotiq 3-finger Adaptive Gripper

A 3-finger adaptive gripper made by Robotiq (further mentioned as RQ3) is attached in the end effector of the UR5 manipulator. The RQ3 gripper is able to grasp a variety of objects by utilizing four basic grasping modes: basic, pinch, wide and scissor, as illustrated in Figure 3. Besides, the fingertips on each of the three fingers are interchangeable and can be customized if the grasping task has special requirements. For every mode, the position, speed, and force of the fingers can be controlled, either individually for each finger or as a system for all fingers. The maximum payload for the RQ3 gripper is 10.0 kg, but if the work piece is grasped using the fingertips, the maximum payload is decreased to 2.5 kg. The RQ3 gripper itself weighs 2.3 kg, thus 46% of the UR5 manipulator's lifting capacity is used.

3.1.3 Hybrid stereo vision system

The vision system on the Little Helper 4 is developed by SDU as part of WP4. The aim of the vision system is to be applicable for a wide range of tasks, enabling both high precision, e.g., for assembly tasks, but also reasonable high performance when searching the environment for a specific object. Therefore the sensor unit is composed of:

- two high resolution cameras, AVT Pike F-421C, for stereo vision,
- a pattern projector to enable dense stereo also at homogeneous areas,
- a RGBD-sensor, PrimeSense Carmine 1.09,
- a pan-tilt unit, FLIR PTU-D48 E, that enables the robot to observe different working areas.

The stereo images are processed with CoViS for generating a 3D reconstruction of the scene and analyzing it, e.g., by estimating the pose of objects when 3D object models are available. The Carmine sensor provides 3D information in terms of point clouds directly.

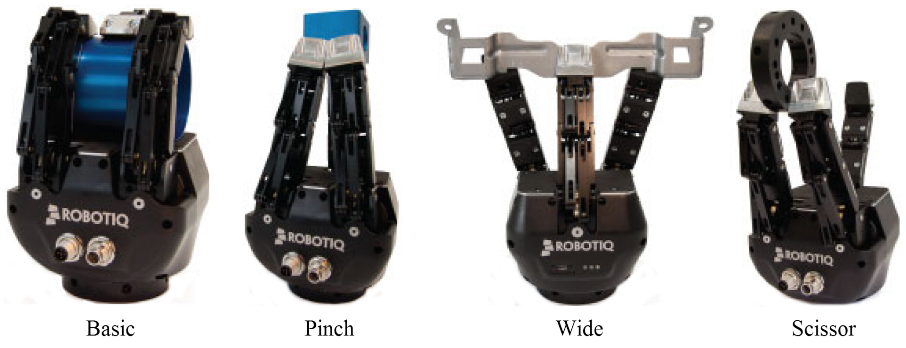


Figure 3: The four different grasping modes of RQ3 gripper.

3.1.4 Modular robot platform

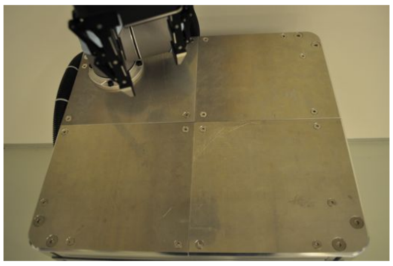
The following considerations had to be taken into account, prior the construction of the modular robot platform:

- The platform has to be relatively heavy, either by using heavier structure materials or by adding extra weights, to maintain stability and avoid tipping over when working around the limits of the manipulator's workspace.
- Shall be movable by a human operator while being essentially rigid.
- The contact points between the structure and the floor is of utmost importance.

Taking all the above features into account, we constructed the first prototype of Little Helper 4, initially as a 3D prototype and then as a physical one. The structure is based on extruded aluminium profiles, since they are modular, while still being able to maintain the structure relatively rigid. Considering the experience derived from the construction of the three previous Little Helpers we knew that a great amount of stability can be provided by the top and bottom aluminium plates. As a result, their thickness is 10 mm and 5 mm, respectively.

The structure is designed with modularity in mind. Thus, the first prototype is created with a separate upper and lower structure that are connected with a set of bolts. The upper structure contains all the electronic components as a 24 VDC/4 A power converter used as power supply to the RQ3 gripper. For the needs of the vision system of the project, the pan-tilt unit, including vision hardware, is mounted on a pole on Little Helper 4. So, three of the four corners of the application specific area are interchangeable as depicted in Figure 4a.

An essential task is to make the mobile platform stable, such that it maintains the same position after the robot calibration. It has been experienced that the casters on Little Helper 3, which are made of a soft rubber material, is the source of minor instability of the platform. Therefore we chose to mount the Little Helper 4 platform on lifting casters, as shown in Figure 4b. These are made of PA nylon, which is a harder material and each of them can withstand 2000N. This makes them ideal for a heavy platform. The casters can switch between being normal heavy duty casters or used as fixed feet. The final result of the construction of Little Helper 4 as a 3D prototype, a first physical robot and the most recent prototype with the vision system attached, are illustrated in Figure 5a, 5b and 5c, respectively.


(a) Application specific area divided into tiles on Little Helper 4.

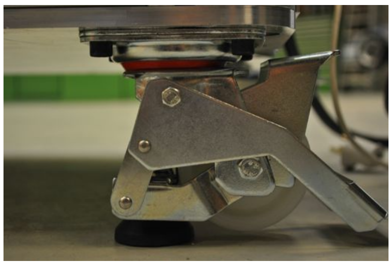
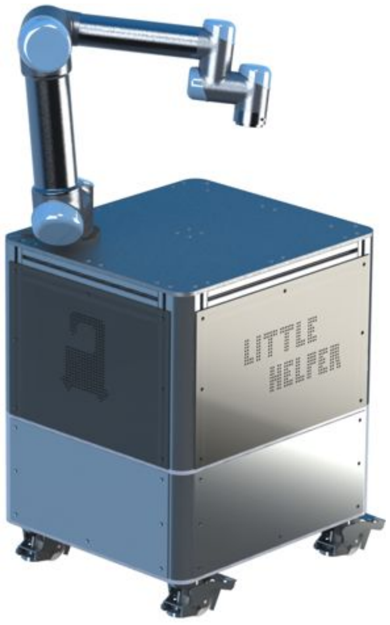
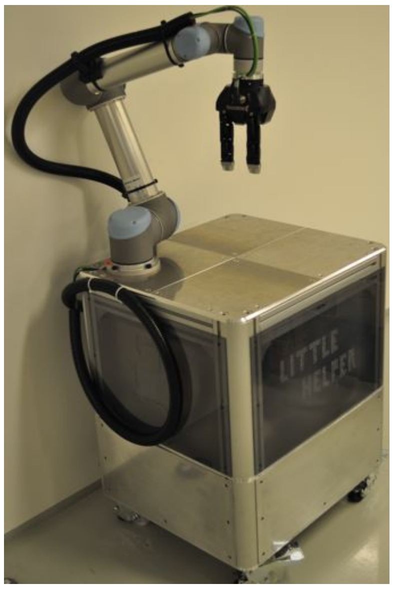

(b) Lifting casters on Little Helper 4 in fixed feet position.

Figure 4: The top mounting platform and the caster wheels of Little Helper 4.



(a) Final version of Little Helper 4 in CAD


(b) First physical prototype of Little Helper 4

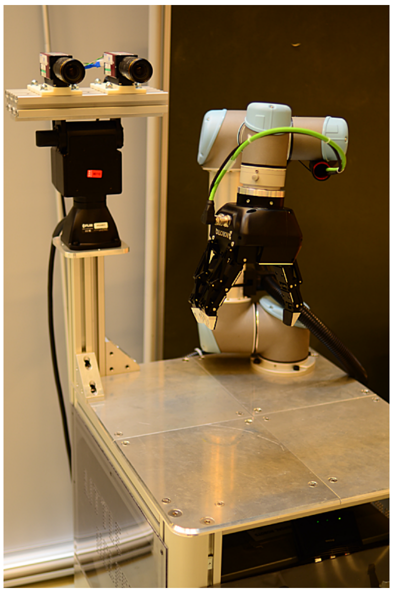

(c) Little Helper 4 with pole and stereo vision system attached

Figure 5: Little Helper 4 as the initial 3D model, initial physical prototype and latest prototype with vision system attached.

3.2 Software architecture

The main scope of the software framework of Little Helper 4 for the needs of the ACAT project and the IASSES scenario, is to maintain modularity and flexibility across hardware layers and different user commands coming from the ADTs. It must be made possible to incorporate other types of hardware, replacing the current one, with none or minor changes to the software framework. Moreover, it should offer the possibility to cope with uncertainties in case of unknown environments and be able to be reprogrammed on the fly, easily by the user. The three main pillars of our software framework are the device primitives, skills and tasks. An overview of these three main components is given below and more extensive descriptions can be found in [8], and [6]. Besides, an illustrated guide to our different layers is shown in Figure 6.

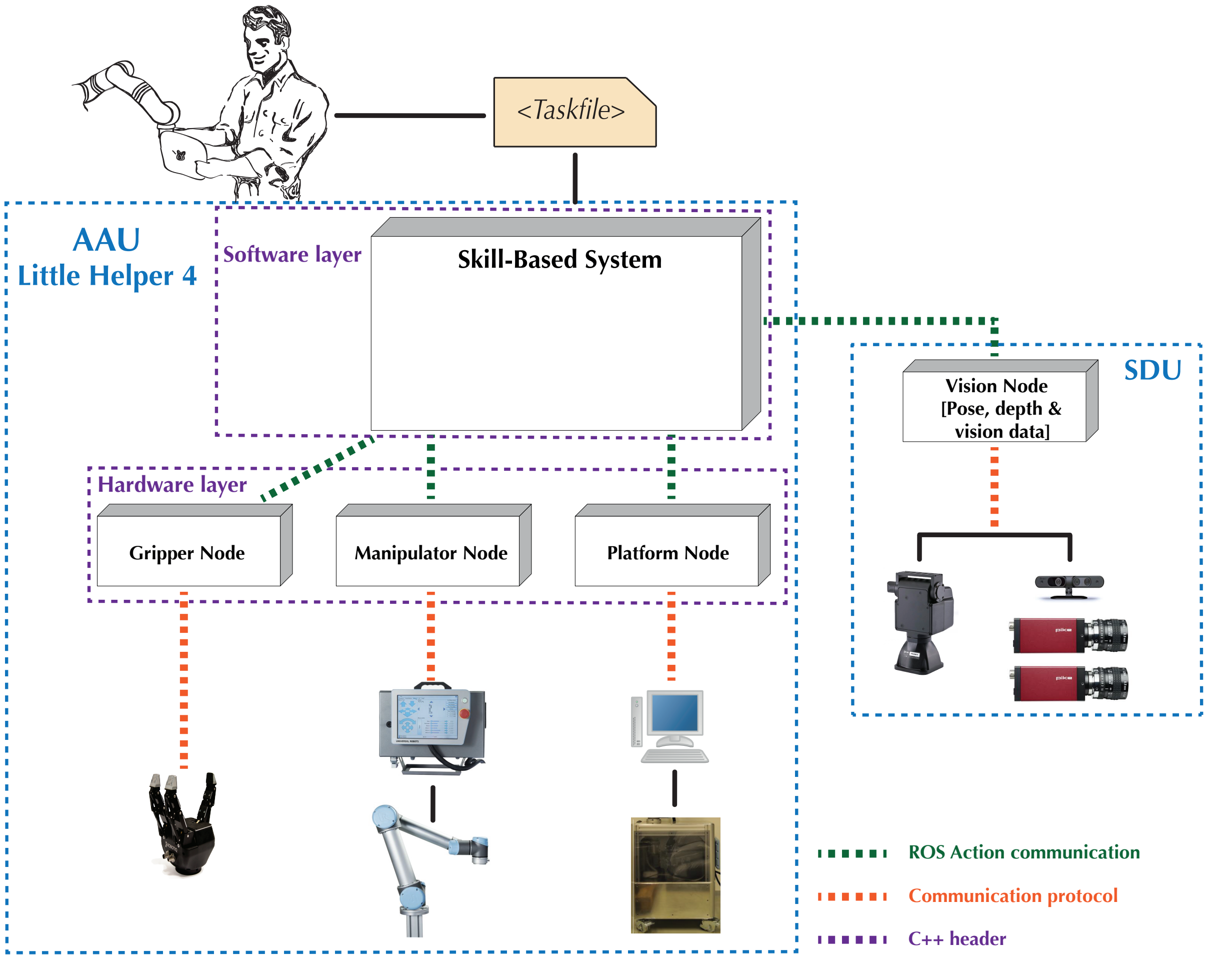


Figure 6: An overview of the software and hardware layers developed by AAU and SDU for the IASSES scenario.

3.2.1 Device primitives

Device primitives can be described as the basic motions and functionalities of the different hardware components used in Little Helper 4. They are the lowest level of hardware abstraction and act as an API between the hardware layer and the skill library. Moreover, they are developed utilizing the communication protocols available for the specific hardware component used for the IASSES scenario.

3.2.2 Skills

One of the key elements in bringing robot instruction from a robot-specific programming language to a higher abstraction level is skills. First of all, skills can represent the foundation of the task and, thus, the building blocks used by the operator. Moreover, they are defined as a higher-level abstraction of functionalities and motions of the individual devices of the robot. A skill utilizes these device primitives as motions combining them with sensor input, advanced mathematics and advanced robotics in order to achieve a production-related goal. That is, a skill is an object-oriented capability of all Little Helpers, i.e., "pick", "place", "move", etc. A thorough description can be found in [7].

One of the most challenging implementations in ACAT project and specifically for the IASSES scenario is the "translation" of the skills implemented in the previous Little Helpers using a KUKA lightweight manipulator to the Little Helper 4 that uses the UR5 arm. The increased difficulty for such endeavor lies in the different device primitives and hardware layers used by the two manipulators. In order to implement the task file for a simple pick and place scenario the following sequence of skills have already successfully imported from the framework used with the KUKA manipulator to Little Helper 4 with UR5 manipulator.

- *MoveTo* skill is created to move the manipulator to a point, with joint movement.
- *MoveToL* skill is created to move the manipulator to a point, with linear movement.
- *Pick* skill is based on a pick point, where a pick has to be executed. The approach and leaving point are calculated based on the pick point and the specified approach and leaving direction. Calculations are done either with respect to the basis of the base or tool frame of the manipulator. The movement towards the pick point and leaving point is made as linear movements, whereas the movement towards the approach point is in joint movements. The gripper is opened at the beginning of the skill and closed after the manipulator reaches the pick point.
- *Place* skill is based on a place point, where placing of a work piece has to be conducted. Similar to Pick skill, the approach and leaving point are calculated based on the pick point and the specified approach and leaving direction. Calculations are done either with respect to the basis of the base or tool frame of the manipulator. The movement towards the place point and leaving point is made as linear movements, whereas the movement towards the approach point is in joint moments. The gripper is kept closed at the beginning of the skill and opened after the manipulator reaches the placing point.
- *PickwithVision* skill uses a depth sensor to locate rotational symmetrical objects, as the rotor cap used in the IASSES scenario, and then picks them up using a small number of taught parameters. More implementation details for this advanced skill can be found in [1]. This is an initial effort where a carmine depth sensor used and tested to check the manipulator's response. The main implementation of this skill will be provided by SDU where it will be enhanced with pose estimation information produced by combining vision data obtained by a stereo vision system and depth data delivered by a carmine sensor, while all of them will be controlled by a pan tilt unit.

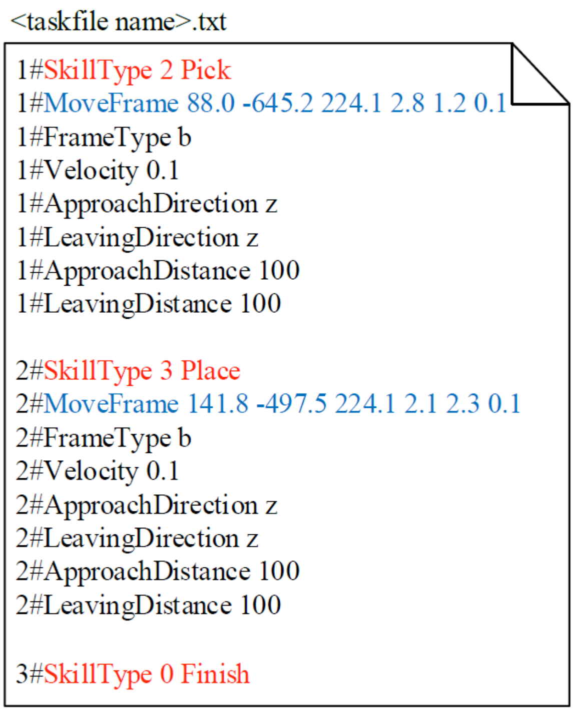
Hardware Specific Skills
Hardware specific skills are skills that utilize specific hardware functionalities as the ones described below:

- *PlaceOnTosimple* skill is based on a Place skill, where the UR5 manipulator uses the build-in force function, to use force as a goal, for termination and not a position. The force mode is ended after the waiting time, and the RQ3 gripper is opened.
- *TeachPointStart* skill activates force mode with compliance set to zero in all directions and rotations. This sets the UR5 manipulator free to be moved around by the user and remains in force mode until the TeachPointEnd skill is called.

- *TeachPointEnd* skill ends the force mode. The TeachPointEnd skill returns the current position of the UR5 manipulator, unaffected of a prior TeachPointStart call.
- *ShowForce* skill is used to show observers the capabilities of the build-in force control, in the UR5 manipulator. The skill moves the UR5 manipulator to a safe position, before activating force mode until a user sends a signal to end the force mode.

3.2.3 Task level

A task can be generally described as a whole process, that contains an overall goal, e.g., pick up a rotor cap from conveyor belt from station A. They are composed of a sequence of instantiated skills where each of them is parametrized to the specific task. In the implementation used for ACAT and the IASSES scenario, this would be encoded by a file containing the sequence of skills and the parameters needed for the correct execution of each skill. A task file can offer the possibility to have a set of measurable state variables that are interchangeable during the execution of the skill sequence. A short example of how a short task file using two skills for a simple pick and place task is shown in Figure 7.



<taskfile name>.txt

```
1#SkillType 2 Pick
1#MoveFrame 88.0 -645.2 224.1 2.8 1.2 0.1
1#FrameType b
1#Velocity 0.1
1#ApproachDirection z
1#LeavingDirection z
1#ApproachDistance 100
1#LeavingDistance 100

2#SkillType 3 Place
2#MoveFrame 141.8 -497.5 224.1 2.1 2.3 0.1
2#FrameType b
2#Velocity 0.1
2#ApproachDirection z
2#LeavingDirection z
2#ApproachDistance 100
2#LeavingDistance 100

3#SkillType 0 Finish
```

Figure 7: Example of a pick and place task file.

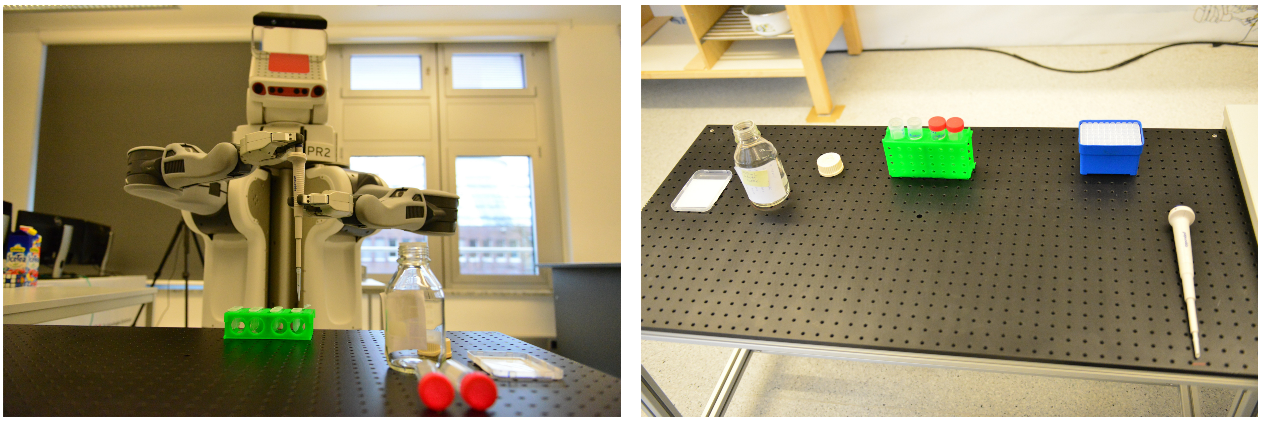

(a) (b)

Figure 8: a) the PR2 holding a pipette; b) some of tools in the ChemLab scenario.

4 The ChemLab scenario

The ACAT ChemLab scenario is the process of DNA extraction from a sample. The process involves the handling of liquids (pouring, decanting, etc.) and usage of standard laboratory equipment such as jars of different size and shape, filter cartridges, and a centrifuge. In order to be successful the process has to be executed under the required constraints (temperature, time schedule, etc.) stated in the respective lab protocol.

4.1 ChemLab Hardware Architecture

In the ACAT ChemLab scenario two different robot platforms are used:

- a Willow Garage PR2,
- an omni-directional mobile robot with two KUKA LWR-4+ arms.

In our first experiments we used the PR2 platform, see Figure 8a where we show the PR2 pipetting. The PR2 is a two-armed, mobile 'personal' robot platform developed by the Menlo Park-based company Willow Garage. The PR2 has an omni-directional base and two arms with 7 degrees of freedom, supported by a torso with one DOF. Each arm also has one adaptable gripper, that is able to pinch and grasp small objects with up to 150 N. It also has pressure sensors on the fingertips to regulate the grasping force and a three-axis accelerometer to detect contacts. Each arm has a payload of 1.8 kg. The robot has a sensor package that includes stereo vision cameras, a tilting laser, and a head-mounted Microsoft Kinect sensor. This robot is the main development platform for ROS (Robot Operating System), now developed by the OSRF (Open-Source Robotics Foundation).

The robot with KUKA LWR-4+ has similar capabilities, but the arms have a payload of 14 kg and we have two different grippers to adjust to different objects. The robot is developed to be compatible from the software point of view as much as possible with the PR2. This makes it straightforward to exchange software developed on one robot to the other.

In our first experiments we used standard laboratory equipment (see also Figure 8b), including:

- an Eppendorf pipette,

- small pipette tips (stored in a box),
- bottles of different sizes and shapes,
- test tubes (stored in a test tube rack).

The PR2 supports a lot of open-source software packages, including the open robot operating system ROS developed by Willow Garage together with the Stanford Artificial Intelligence Laboratory, open libraries for different purposes, e.g., vision (OpenCV), and the 3D Point Cloud Processing Libraries (PCL).

4.2 ChemLab Software Architecture

UoB has developed Probabilistic Robot Action Core (PRAC) models [13], which are action-specific probabilistic first-order knowledge bases representing generic types of events in the real world and executable robot actions, respectively. Such an action core pattern consists of the inter- and intra-conceptual relations between all entities that are affected by a particular action verb and therefore can be seen as a probabilistic relational model of action parametrization.

Nyga and Beetz [13] have shown that PRAC models can also include linguistic features and hence can be learned from natural language, which is a focal aspect of ACAT. In addition, they are well-suited for automated inference of underspecified action parameters. Thus, PRAC models can be used to provide the necessary parameters for instantiating generic action plan schemata into executable robot plans implemented in a concurrent reactive plan language, such as CRAM (cf. Figure 9 for the basic structure and [5] for further details) .

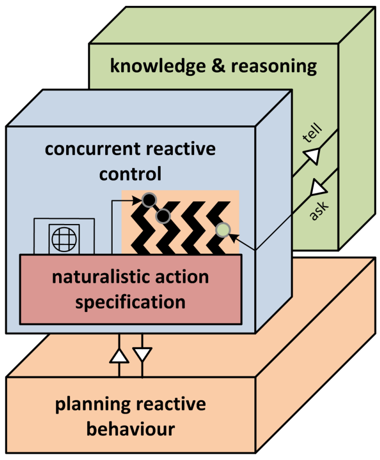

knowledge & reasoning
concurrent reactive control
tell
ask
naturalistic action specification
planning reactive behaviour

Figure 9: Cognitive Robot Abstract Machine (CRAM)

Its core component is a probabilistic first order logic knowledge base represented as a Markov logic network [14]. The content of this knowledge base is the result of a linguistic analysis pipeline (see below) applied to instruction texts, e.g., those especially collected for the ChemLab partition of the ACAT text corpus or to other English instruction texts, e.g. cooking recipes from websites such as wikiHow.com.

The basic components of the linguistic analysis are:

- a linguistic preprocessing and syntactic analysis system which includes as a major component the Stanford Parser [10, 12]. This parsing system provides syntactic information about the

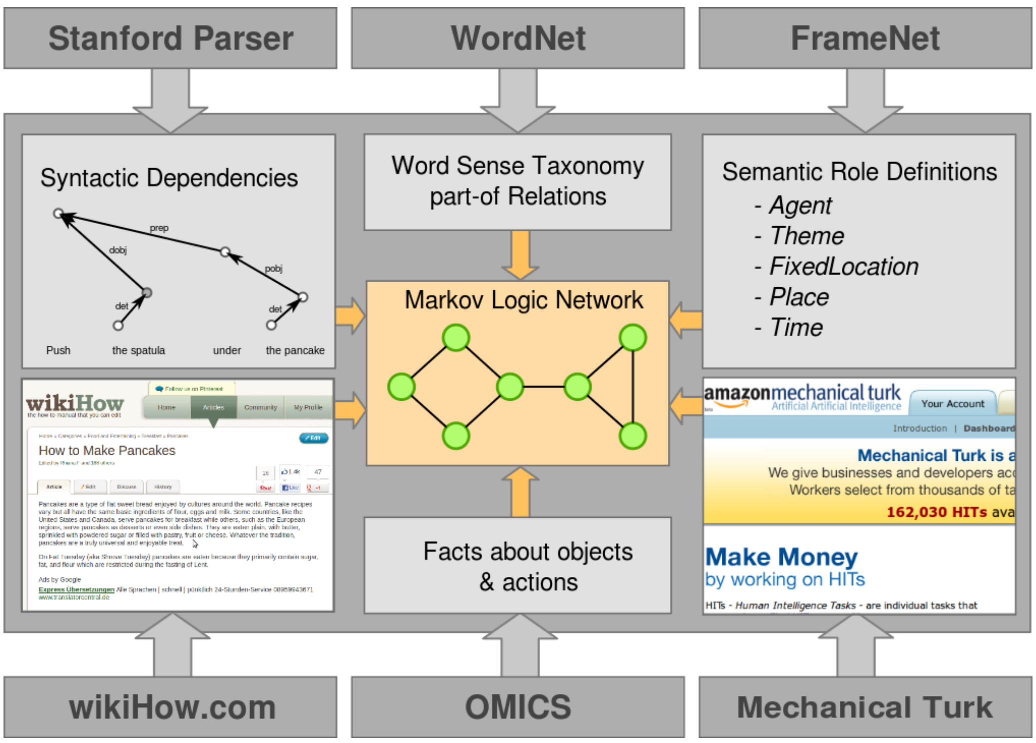


Figure 10: The schematic architecture of the ChemLab PRAC model.

sentences in the input documents both in the form of labeled phrase structure trees and dependency graphs.

- WordNet, a large-scale lexical semantics knowledge base of English which we used as a source of semantic relations such as hyperonym/ hyponyms (IS-A), PART-OF, etc. These relations can be used to connect an input word to its appropriate place in taxonomies and partonomical (mereological) hierarchies.

- FrameNet, a lexical database providing detailed information on semantic roles, which is of particular importance for the analysis of (action) verbs and their semantic relationship to the noun phrases directly involved (e.g., the verb's subject, object, etc.).

The probabilistic parameters of the model have been estimated using data created by human annotators at Amazon's Mechanical Turk. The architecture also allows for the integration of additional knowledge sources, such as the OMICS (Open Mind Indoor Common Sense) database.

The PRAC subsystem is part of the CRAM-based robot architecture which uses the concept of structured reactive controllers (SRCs [4]). An SRC is a collection of concurrent control routines that specify routine activities and can adapt themselves to non-standard situations by means of planning. The controllers are called reactive because they can respond immediately to asynchronous events and manage concurrent control processes. They are called structured because they use expressive control abstractions to structure complex activities. Structured reactive controllers execute three kinds of control processes: handling of standard situations, monitoring for non-standard situations, and adapting to non-standard situations [3, 2].

5 Simulation platform

The simulation platform for ACAT, developed at SDU, is still a work in progress and this section will describe the predicted architectural overview and the current status on methods that use the simulation.

In essence, the simulation platform is used to

- generate background knowledge. The simulation can be used for both computation of knowledge such as grasp databases and for massive execution of action sequences and the generation of experience formed in ADT's.
- do action optimization through exploration. The simulation allows optimization of actions by exploring variations in action parametrization.
- do optimization and validation of robot cell design. The simulation can be used in the scene analysis component to validate or even optimize the design of the robot workcell in accordance to its task.

5.1 Virtual platform

The virtual platform can be thought of as a virtual replacement of a real hardware platform, which allows for the generation of background data on a large variety of robot systems. The dynamic simulation that is used was first presented in [9]. It has later been modified and is still being developed in order to simulate advanced robotics processes.

The virtual platform is loaded through a dynamic workcell description used by RobWork. The description allows for the combination of various robot designs, sensors, controllers and actuators. The interface to the virtual platform is defined through Robot Operating System (ROS) and the experience produced on the virtual platform will be converted to ACAT ADT's and feed back to the ACAT database.

The action execution system that is implemented on top of the virtual platform is based on the RobWork framework for control, planning and modeling, and CoViS for pose estimation. However, the components used in ACAT are still in early development and will be described in more detail in D5.5 in month 30.

5.2 Direct Background generation

The virtual platform enables the execution of action sequences directly, however for some tasks, such as grasping, background data can be more efficiently computed using dedicated exploration of the task. At SDU several data-driven grasp planning methods have been developed, see example in [11]. These rely on dynamic, kinematic and geometric models of the problem and allow the generation of databases of high quality grasps.

In ACAT these methods will be extended such that they can be used with the underactuated gripper from Robotiq. This gripper is used in the IASSES scenario. Figure 11 illustrate grasps generated on two different objects of the IASSES scenario. The figure also visualizes the Robotiq 3-finger gripper used in the simulation.

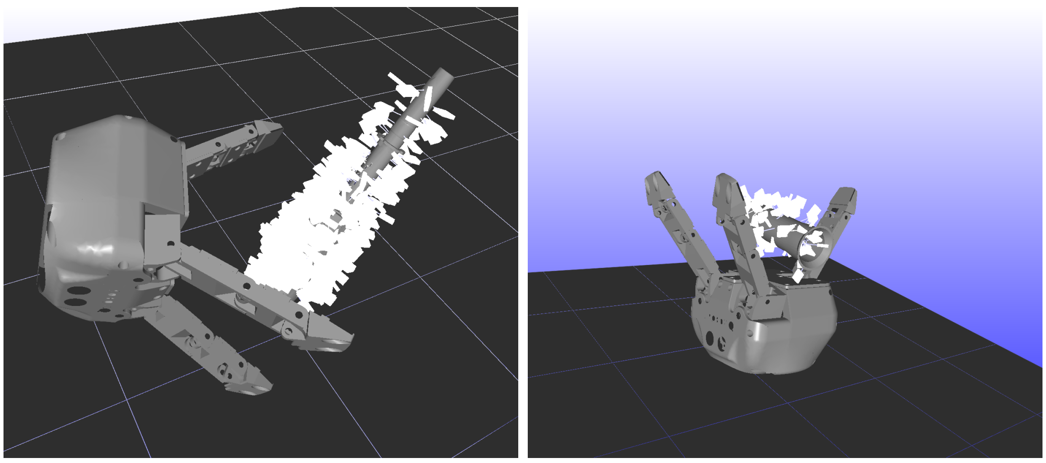


Figure 11: Grasp generated using dynamic simulation and biased grasp target sampling.

5.3 Scene analysis - gripper performance

A method has been developed to use the simulator to evaluate how well a gripper will perform in a given task. A paper describing this method has been submitted to IROS 2014, see [16]. We will only briefly describe the motivation for such a metric here.

Industrial applications such as those demonstrated in the IASSES scenario typically use simple gripping technologies adapted to the few parts that the system should handle. These gripping technologies require significant manual labor from expert users to design the shape of the gripper surfaces. This conflicts with the ACAT idea of automatically compiling human readable worksheets into robot programs since these programs will require a specific robot embodiment. A method that automatically computes gripper designs will minimize the manual labor involved in programming the robot system to a new task.

Our current work on scene analysis [16] is a step toward automatic computation of gripper designs. It focus on the automatic evaluation of a specific gripper design as depicted in Figure 13. The gripper design is evaluated in a specific task context such as table or conveyor belt picking (see figure 12). By evaluating individual grasp successes in dynamic simulation we are able to extract three quality measures of a complete gripper design.

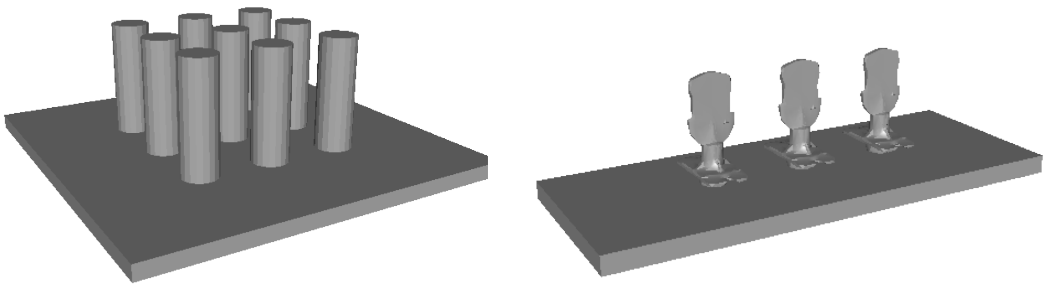


Figure 12: Two simulated scenes: Table picking (left) and conveyor dolt pickking (right).

The gripper evaluation method can also serve as a validation of the action kernel sequence, by analyzing if the current gripper is sufficiently successful in grasping relevant objects. This and the automatic gripper designer is left as future work in ACAT.

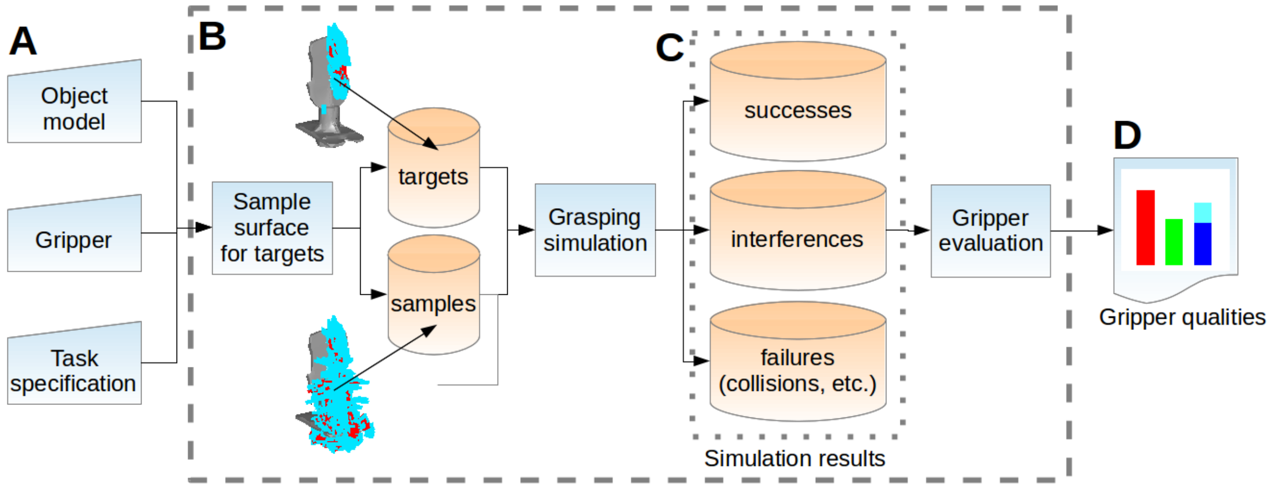


Figure 13: The process of automatically computing the quality of a gripper in a specific task context for grasping a specific object.

5.4 Cluster for massive exploration

The ACAT cluster at SDU (see task 1.3 in the ACAT DoW) is not directly part of the hardware architecture. However, as it serves as a computational platform in the simulations used in ACAT we include a small description of it here.

The cluster at SDU currently consists of ten computers connected through a 1 gigabit ethernet network. The computers are equipped with Intel(R) Core(TM) i7-3770 CPU @ 3.40 GHz (quad core and hyperthreading), 32 GB memory, one NVIDIA NVS 310 graphics card and 4 TB storage each. Currently they are running Ubuntu 12.04 and uses version 2.4.16 of TORQUE (Tera-scale Open-source Resource and QUEue manager [15]) to dispatch the user specified jobs between the computers.

6 Conclusion

A description of the hard- and software architecture for the three ACAT platforms has been given. The three platforms are ready to operate (UoB ChemLab and the SDU simulation platform) or close to being so (AAU IASSES). They willl be further extended during the ACAT project.

References

[1] R. Andersen, L. Nalpantidis, V. Kruger, O. Madsen, and T. Moeslund. Using robot skills for flexible reprogramming of pick operations in industrial scenarios. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2014.

[2] M. Beetz, D. Jain, L. Mosenlechner, M. Tenorth, L. Kunze, N. Blodow, and D. Pangercic. Cognition-enabled autonomous robot control for the realization of home chore task intelligence. *Proceedings of the IEEE*, 100(8):2454–2471, Aug 2012.

[3] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, Thomas Ruhr, and M. Tenorth. Robotic roommates making pancakes. In *In 2011 11'th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 529–536, Oct 2011.

[4] Michael Beetz. Concurrent reactive plans. anticipating and forestalling execution failures. *Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence*, 1772, 2000.

[5] Michael Beetz, Lorenz Mosenlechner, and Moritz Tenorth. CRAM—A Cognitive Robot Abstract Machine for everyday manipulation in human environments. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on. IEEE*, 2010.

[6] S. Bøgh, M. Hvilshøj, M. Kristiansen, and O. Madsen. Identifying and evaluating suitable tasks for autonomous industrial mobile manipulators (AIMM). *International Journal of Advanced Manufacturing Technology*, 61(5–8):713–726, 2012.

[7] S. Bøgh, O. Nielsen, M. Pedersen, V. Kruger, and O. Madsen. Does your robot have skills? In *Proceedings of the 43rd International Symposium on Robotics*. VDE Verlag GMBH, 2012.

[8] M. Hvilshøj, S. Bøgh, O. Nielsen, and O. Madsen. Multiple part feeding – real-world application for mobile manipulators. *Assembly Automation*, 32(1):62–71, 2012.

[9] Jimmy Alison Jørgensen, Lars-Peter Ellekilde, and Henrik Gordon Petersen. RobWorkSim - an open simulator for sensor based grasping. In *Joint 41st Int. Sym. on Robotics (ISR)*, 2010.

[10] Dan Klein and Christopher D. Manning. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, pages 423–430, 2003.

[11] Dirk Kraft, Lars-Peter Ellekilde, and Jimmy Alison Jørgensen. Automatic grasp generation and improvement for industrial bin-picking. In Florian Röhrbein, Germano Veiga, and Ciro Natale, editors, *Gearing up and accelerating cross-fertilization between academicand industrial robotics research in Europe:*, volume 94 of *Springer Tracts in Advanced Robotics*, pages 155–176. Springer International Publishing, 2014.

[12] Marie-Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *LREC 2006*, 2006.

[13] D. Nyga and M. 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)*, pages 243–250, Oct 2012.

[14] Matthew Richardson and Pedro Domingos. Markov Logic Networks. In *Machine Learning*, 2006.

[15] Garrick Staples. TORQUE resource manager. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 8, New York, NY, USA, 2006. ACM.

[16] Adam Wolniakowski, Konstantsin Miatliuk, Norbert Krüger, and Jimmy Alison Rytz. Automatic evaluation of task-focused parallel jaw gripper design. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2014: Expanding the Societal Role of Robotics, Chichargo, Illinois, USA, Sept. 14-18, 2014*, 2014 (submitted).