Deliverable number:	D2.1
Deliverable Title:	Background data structure
Type (Internal, Restricted, Public):	PU
Authors:	H. Langer, C. Landgraf, D. Nyga, M. Beetz, I. Markievicz, D. Vitkute-Adzgauskiene, M. Tamosiunaite, F. Wörgötter
Contributing Partners:	UoB, UGOE, VDU

Project acronym:	ACAT
Project Type:	STREP
Project Title:	Learning and Execution of Action Categories
Contract Number:	600578
Starting Date:	01-03-2012
Ending Date:	28-02-2015

Contractual Date of Delivery to the EC:	02-12-2013
Actual Date of Delivery to the EC:	02-12-2013

Content

1. Executive summary

This deliverable describes the data structures developed and used in ACAT. Two components are defined: 1) Robot execution-relevant data structures and 2) Symbolic, ontological data structures.

The following entities will be introduced in this document (ADT stands for Action Data Table):

1) Action Classes with one or more primitives each;
    a) ADTs for every primitive consisting of HEADER and SEQUENCE OF ACTION CHUNKS
        i. HEADER with basic action information;
        ii. SEQUENCE OF ACTION CHUNKS with action relevant parameters for each chunk.

2) Text-based ontology data structure consisting of
    a) ACTIONS, action classes and action parameters;
    b) OBJECTS and object parameters.

Furthermore, we discuss how these data structures are linked to each other so that the ACAT problem of compiling a human instruction sheet for robot use can be addressed.

2. Introduction

This deliverable is to provide the data structure for background knowledge in the ACAT project. By "background knowledge" we mean the shared knowledge humans have about the world which they would not explicitly mention in texts (here: instruction sheets) written for other humans. If a robot has to execute an instruction sheet written for humans, the required background knowledge for a robot consists of two components: symbolic knowledge (in the field of artificial intelligence this would be called common-sense knowledge, i.e. the collection of facts that an average human is expected to know) as well as the necessary symbol to signal link (control level knowledge) that the robot has to have to execute symbolic instructions.

The amount of "background" knowledge, as defined above, is huge as compared to the little knowledge provided in an instruction sheet in our ACAT scenarios. Consequently, when we talk about "background data structure for the ACAT project" we factually talk about the "*complete* data structure for the ACAT project". Thus, in this deliverable we will describe the *data structure(s)* used in ACAT, their links to each other, and some of the processes used to fill the data structures with their data. All these aspects (data structures, links and processes) remain in the core of the ACAT research and are likely to change in the course of the project.

Essentially we are dealing with two types of data structures:

1) Data structures for ontological knowledge, mostly from language sources (**Ontology**), which provide the link to the human instruction sheets and
2) Data structures for interfacing to the robot (so called Action-Data-Tables, **ADT**), which allows robot execution.

For brevity, we will use the abbreviations ”Ontology” and ”ADT” in the following.

Ontological knowledge is mainly symbolic, whereas the ADTs store also much sub-symbolic information, required to allow for robot execution of an action or action sequence.

The motivation for designing these data structures is that ACAT wants to arrive at a standardized (robot)-action representation, which can be used for AI-reasoning processes as well as for robotic applications at different embodiments. This allows us to address the core problem of ACAT: How to compile human instruction sheet into robot compatible execution language.

In the next sections we will first describe the ADTs before providing details about the Ontology. Only at the end we will discuss the links between both and some aspects on how to fill these data structures.

3. General Structure of the ACAT ADT library

ADTs are defined for a few generic action classes, which are of relevance for the ACAT scenarios, but we expect that there are not many more action classes existing (see below).

We distinguish essentially three layers, where the second layer defines the core structure of the ADTs:

1) **Action Classes** – defined by a sequence of standard chunks composing the action (see next);
2) **Action Data Tables** – for every chunk provides all parameters and function calls;
3) **Function-Specifiers** (for controllers) – for complex aspects like e.g. grasp definition provide the necessary structures and parameters (this can include feedback control parameters).

Here we provide all structures as tables (human readable format), though the general format that will be used in ACAT project is in OWL language. Rendering in OWL language will be provided for selected examples at the end of this document.

At the end of these sections we will also discuss error handling.

4. Action Classes

In general we assume that robot actions can be divided into several sequential primitives and a primitive is given by the fact that the robot will (or could) remove its hand at the end of a primitive.

Also note: Locate, grasp, pick up and release (put down) are always **parts** of an action. We define these by calling individual "controllers". Interspersed between those parts can be another action. For example: pick up spoon, stir with spoon in cup, put down spoon (the underlining refers to two action parts).

The following descriptions are coarse-grained. Specifics of how to actually specify this in detail will be given when discussing the Action Data Tables.

Now we define the Action Classes and their primitives. Examples come from our scenarios!

Null-Action

An action with one primitive, only! Wait for some time until some other process completes (E.g. “wait, hold”).

No action for some time

Homing Action

An action with one primitive, only! E.g. “Move robot actuator to a home position”

Move robot arm/hand to a home position

Hand-Only Action

Actions with hand and single object (e.g. “press button, pull up, pull down, screw on/off, open lid (without removing), close lid”)

Note: screw-on/off assumes that the screw sits on the target, if you have to move/remove it then one should consider this as an additional pick&place action.

Locate target object
Act at target

Handling-Action

(e.g. “shake, invert”)

Locate source object
Pick up object
Act
Put down object

Tool Action

Actions with primary tools. (e.g. “stir with a spoon, screw with a screwdriver, hammer with a hammer”)

Locate tool
Pick up tool
Locate target object
Act at target
Remove and Put down tool

Tools-with-Movers-Action

Actions with an additional mover (for example ‘gravity’ or motors): “Pouring” (gravity action!) or systems with transmissions, gears, motors. (Power tools). (e.g., “pour, add a drop, drop something, discard (throw away), mix with mixer)”.

Locate tool
Pick up tool
Locate target object
Act on source such that “mover” does its job on target
Remove and Put down tool

Pick and Place Action

Pick and Place (e.g. “put down, insert [peg and hole], put together, put on top, put into, essentially all variants of putting together and pushing together”).

Locate source object
Pick up object
Locate target object
Put down source object at target

5. Action Data Tables - ADTs

Here we describe in detail how the action data tables are organized. Every primitive has its own ADT! Not always all entries in an ADT will have to be filled, though (see example below). All information is here given in human readable format. For the use with robots these tables are translated in OWL code (an example is given at the end of this document).

An ADT divides into header and sequence of action chunks.

HEADER: The header describes the action type, then the objects with which the action is performed. Next the Semantic Event Chain (SEC) table of the respective action is provided. The SEC defines the sequence of action chunks thus dividing an action into yet smaller temporal entities.

SEQUENCE OF ACTIION CHUNKS: The sequence provides for every chunk given by a SEC all action-relevant details. For example: start and end points of the movement, trajectory and force profiles, pose relationships, grasps, etc.

Next we show how an ADT looks like. The HEADER is everything above the olive-green divider. The SEQUENCE OF ACTION CHUNKS is beneath the divider.

Action Data Table (ADT) for Action = "Name"		
HEADER		
Name	**Description**	**Type**
Action	Action type (name, same as above, verb)	string (Word)
Source object	Object type (name of object, noun)	string (Word)
Target object	Object type (name of object, noun)	string (Word)
Source Object Descriptors	Robotics relevant object description	CAD, or reduced description, incl. "object anchor" frame[1].
Target Object Descriptors	Robotics relevant object description	CAD, or reduced description, incl. "object anchor" frame.
Anchor Points (SEC)	Semantic Event Chain for this action which **defines the chunks of an action**	SEC table
Precondition Check	Assessed status of required preconditions for action as specified by SEC (similar in meaning to success variables)	True/False
Function specifier: Locate	A function call to the locate function specifier (if required)	function(parameters[2])
Locate Success specifier	Confidence in detecting an object + decision detected or not	real value + yes/no
SEQUENCE OF ACTION CHUNKS		
SEC chunk 1		
Start(1)	Move source object **from** there	3D coord
Target(1)	Move source object **to** here	3D coord
Orientation(1)	Desired orientation of source object (or hand/finger) at target.	3D Orientation Vector
Task space trajectory(1)	This is the set of coordinates in task space that define the trajectory (execution methods may differ)	Set of 3D coords
Force(1)	Force(profile) for contact of source with target object	Set of 3D coords
Grasp(1)	If required for this action: Grasp specifier for grasp or ungrasp	function (parameters)
Grasp success specifier(1)	If grasp: value for grasp stability (0=failure)	real value
Additional(1)	As required	As required
Success specifier(1)	Overall success to reach the desired end state + indicators what was unsuccessful (SEC level achieved or not, trajectories match to planned or not, forces ok or not, poses ok or not, etc.)	A set of 4 indicators
SEC chunk 2		
Start(2)=Target(1)	Move source object **from** there	3D coord
Target(2)	Move source object **to** here	3D coord
Orientation(2)	Desired orientation of source object (or hand/finger) at target	3D Orientation Vector
Task Space Trajectory(2)	This is the set of coordinates in task space that define the trajectory (execution methods may differ for partners)	Set of 3D coords
Force(2)	Force for contact of source with target object	Set of 3D coords
Grasp(2)	If required for this action: Grasp specifier for grasp or ungrasp	function (parameters)
Grasp success specifier(2)	If grasp: value for grasp stability (0=failure)	real value
Additional(2)	As required	As required
Success specifier (2)	Overall success to reach the desired end state + indicators what was unsuccessful (SEC level achieved or not, trajectories match to planned or not, forces ok or not, poses ok or not, etc.)	A set of 4 indicators
More chunks if needed		

Note, these tables are not specific w.r.t the actual execution methods. E.g. Trajectories are specified in task space and can be generated "in any way" by a user.

[1] Objects need to come with one specific coordinate and orientation on which the CAD (or other) description "hangs". This coordinate is to be used to define the complete system for different robotic installations in their specific task space.

[2] Here parameters are grasp or locate parameters.

Also, we do – on purpose – not use "pose" as a descriptor but instead "goal" and "orientation" (which are the two aspects that constitute pose). The reason is that for many actions "orientation" can remain rather vaguely specified only.

6. Function-Specifiers and Trajectory and Force Descriptors

We use two types of function specifiers:

1) Locate Process Specifier
2) Grasp Specifier

Locate Process Specifier

Here we will place parameters used for scene investigation in order to locate an object. These contain camera set-up specifiers needed for specific object search, the segments of the images where an object is expected to be, etc.

Grasp Specifier

For the different grasps it may make sense to specify a few default tables, where we will only vary the parameters. Hence:

Name	**Description**			**Value**
Power (params)	Pose	Handpose to source obj.	Matrix	Specifier
	Size	Width of hand opening	Numerical	
	Fource	Fource to graps	Numerical	
Pinch(params)	Pose	Handpose to source obj.	Matrix	Specifier
	Size	Width of hand opening	Numerical	
	Force	Force to grasp	Numerical	
Etc.	Etc.			Specifier

Trajectory

We put the trajectory information into the ADT as a task space coordinate sequence, as this is the general format every user can provide. For execution (re-use) the saved trajectory can be re-parameterized up to individual user demands (e.g. using DMP (Ijspeert et al, 2002, Kulvicius et al., 2012) or splines, or any other useful implementation).

We describe trajectory by indicating position and orientation values (when required) as well as force profiles (when required).

Force

Here force profile is specified, as a sequence of force values (3D sequence). Again, users can re-parameterize the force profiles for re-use at their own convenience.

7. Filling the ADTs and Reactive Control Mechanisms

This deliverable does not deal with these aspects but they are being addressed through major research efforts in ACAT to be reported elsewhere.

8. Example for a set of ADTs for one action from a given class

Let us assume we have an instruction from our scenario: "Place the rotor cap on top of the fixature". The instruction imposes the following actions (which we will analyze in detail below):

1) Locate rotor cap
2) Pick up rotor cap
3) Locate fixature
4) Put rotor cap on fixature

This is an example from the Action Class: Pick and Place Action given by

Locate source object
Pick up object
Locate target object
Put down source object at target

Specifically, we will provide here ADTs for the four primitives defined above. In blue font human-readable entries for concrete parameter values are given.

Graphical visualization of ADT conceptual model with two action entities ("Locate rotor cap" and "Pick up rotor cap") is given in Fig. 1. In addition, the OWL code for the first two actions ("Locate rotor cap" and "Pick up rotor cap") is provided in the Appendix.

LOCATE (source object – rotor cap)

Name	**Description**	**Type**
Action	Locate	string (Word)
Source object	Rotor Cap	string (Word)
Target object	Void	string (Word)
Source Object Descriptors	CAD or similar description of tool (required to define the grasp) for the Rotor Cap	CAD, or reduced description, incl. “object anchor” frame.
Target Object Descriptors	Void	CAD, or reduced description, incl. “object anchor” frame.
Anchor Points	Void	SEC table
Precondition Check	Void	True/False
Function specifier: Loacte	A function call to the function specifier of locate	function(parameters)
Locate success specifier	Confidence in detecting an object + decision detected or not.	real value + yes/no
SEC-determined Sub-Parts (index is determined by the anchor points as given by the SEC)		
Void		

PICK UP OBJECT (rotor cap)

Name	**Description**	**Type**
Action	**Pick up Object**	string (Word)
Source object	*Transferred:* Rotor Cap	string (Word)
Target object	Void	string (Word)
Source Object Descriptors	*Transferred:* CAD or similar description of the Rotor Cap	CAD, or reduced description, incl. “object anchor” frame.
Target Object Descriptors	Void	CAD, or reduced description, incl. “object anchor” frame.
Anchor Points	Semantic Event Chain for pick up object (hand,object) 0 1 1 (object,table) 1 1 0	SEC table
Precondition Check	Assessed status of required preconditions for action as specified by SEC above.	True/False
Locate Function specifier	Void	function(parameters)
Locate success specifier	Void	real value + yes/no
SEQUENCE OF ACTION CHUNKS		
SEC shunk 1 **(hand goes from home to grasp)**		
Start(1)	Start position of robot hand: Home	3D coord
Target(1)	Target position for grasp: Middle of the Rotor Cap	3D coord
Orientation(1)	Desired pose of tool versus hand for grasping: As required for pinch perpendicular to the Rotor Cap	3D Orientation Vector
Task Space: Trajectory(1)	Trajectory in task space for the source object: Default straight or as demonstrated	Set of 3D coords
Force(1)	Force for contact of source with target object: Grasp force	set of 3D coords.
Grasp(1)	Pinch perpendicular to the Rotor Cap	function (parameters)
Grasp success specifier(1)	Value for grasp stability (0=failure)	real value
Additional(1)	As required. Void	As required
Success specifier(1)	Overall success to reach the desired end state + indicators what was unsuccessful (SEC level achieved or not, trajectories match to planned or not, forces ok or not, poses ok or not, etc.)	An set of 4 indicators
SEC shunk 2 **(grasped object is lifted)**		
Start(2)	Start position of the chunk (=Target(1))	3D coord
Target(2)	End position of the lift. Default: 1 mm up or as demonstrated.	3D coord
Orientation(2)	Desired pose of the lift. Default same as at the start or as demonstrated.	3D Orientation Vector
Task Space: Trajectory(2)	Trajectory in task space. Straight if no obstacles	Set of 3D coords
Force(2)	Force for lifting	set of 3D coords.
Grasp(2)	Void	function (parameters)
Grasp success specifier(2)	Void	real value
Additional(index)	Void	As required
Success specifier(2)	Overall success to reach the desired end state + indicators what was unsuccessful (SEC level achieved or not, trajectories match to planned or not, forces ok or not, poses ok or not, etc.)	An set of 4 indicators

LOCATE (target object - fixature)

Name	Description	Value
Action	**Locate**	string (Word)
Source object	Fixature	string (Word)
Target object	Void	string (Word)
Source Object Descriptors	CAD or similar description of tool (required to define the grasp) for the Fixature	CAD, or reduced description, incl. “object anchor” frame.
Target Object Descriptors	Void	CAD, or reduced description, incl. “object anchor” frame.
Anchor Points	Void	SEC table
Precondition Check	Void	True/False
Locate Function specifier	A function call to the function specifier of locate	function(parameters)
Locate success specifier	Confidence in detecting an object + decision detected or not	real value + yes/no
SEC-determined Sub-Parts (index is determined by the anchor points as given by the SEC)		
Void		

PUT DOWN (rotor cap on fixature)

Name	Description	Type
Action	**Put down Object**	string (Word)
Source object	*Transferred:* Rotor Cap	string (Word)
Target object	Name of Target Object: Fixature	string (Word)
Source Object Descriptors	*Transferred:* CAD or similar description of Rotor Cap	CAD, or reduced description, incl. “object anchor” frame.
Target Object Descriptors	Reduced descriptor for target (surface, is-free, etc.): CAD or similar description of Fixature	CAD, or reduced description, incl. “object anchor” frame.
Anchor Points	SEC for put down (Hand,R.Cap) 1 1 0 (R.Cap,Fixature) 0 1 1	SEC tabe
Precondition Check	Assessed status of required preconditions for action as specified by SEC above.	True/False
Locate function specifier	Void	function(parameters)
Locate success specifier	Void	real value + yes/no
SEQUENCE OF ACTION CHUNKS		
SEC chunk 1 (actual put down)		
Start(1)	*Transferred*: (= former target position after previoius action (Target(2) in prev. table))	3D coord
Target(1)	Target position for put down: Position of fixature	3D coord
Orientation(1)	Desired pose of the put down. Default same as at the start.	3D Orientation Vector
Task Space: Trajectory(1)	Trajectory in task space. Default movement of putting from above if no obstacles	Set of 3D coords
Force(1)	Force for putting down	set of 3D coords.
Grasp(1)	Void	function (parameters)
Grasp success specifier(1)	Void	real value
Additional(1)	Void	As required
Success specifier(1)	Overall success to reach the desired end state + indicators what was unsuccessful (SEC level achieved or not, trajectories match to planned or not, forces ok or not, poses ok or not, etc.)	An set of 4 indicators
SEC chunk 2 (retract hand)		
Start (2)	=Target(1)	3D coord
Target(2)	target position of the hand, home default	3D coord
Orientation(2)	Desired pose of the hand: home default	3D Orientation Vector
Task Space: Trajectory(2)	Trajectory for bringing hand home, default straight (or comes from demonstration)	Set of 3D coords
Force(2)	Void	set of 3D coords.
Grasp(2)	Un-grasp	function (parameters)
Grasp success specifier(2)	Success specifier (0=failure)	real value
Additional(2)	Void	As required
Success specifier(2)	Overall success to reach the desired end state + indicators what was unsuccessful (SEC level achieved or not, trajectories match to planned or not, forces ok or not, poses ok or not, etc.)	An set of 4 indicators

The figure below shows a graphical representation of the above example.

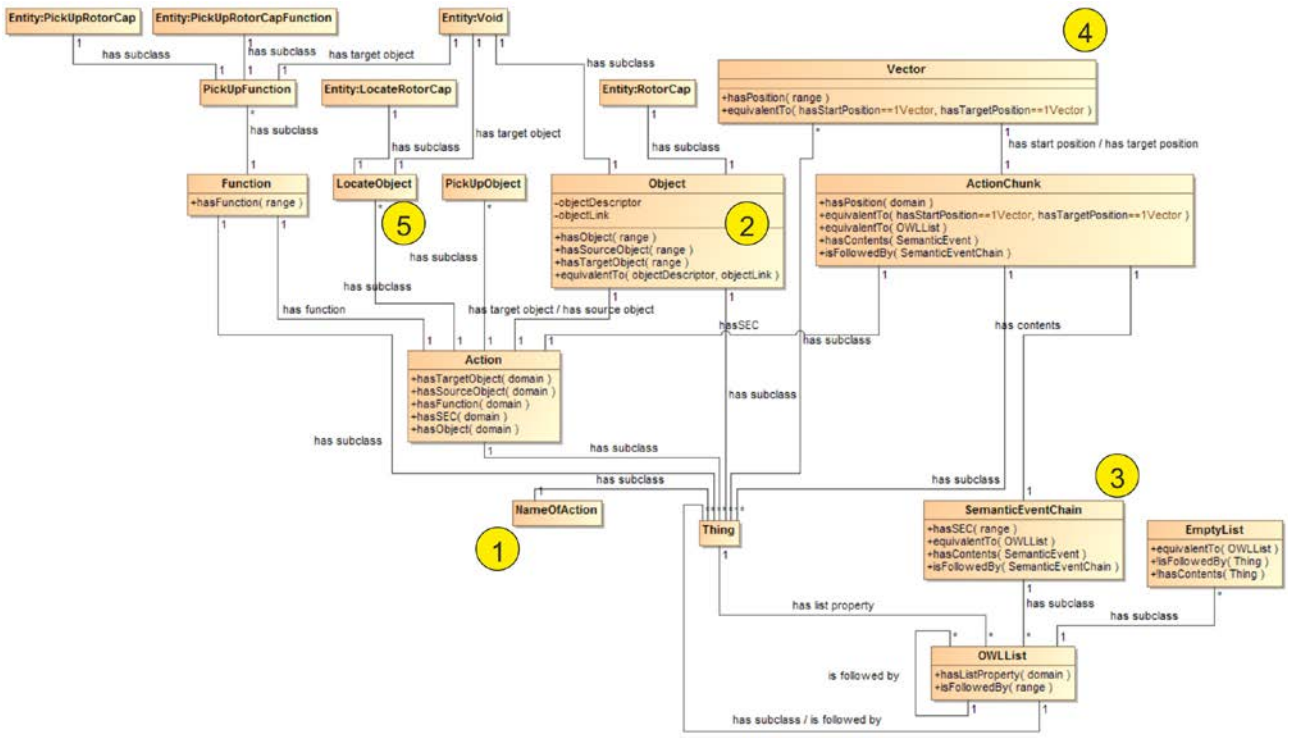


Fig. 1 Graphical visualization of ADT conceptual model with two examples of action entities ("Pick up rotor cap" and "Locate rotor cap"). Numbers show link points to the text-based ontology conceptual model (Fig. 2). The OWL-List component is taken from Drummond et al (2006).

9. Interim Summary

Up to this point we have described that part of the data structures which will allow for robot execution of different actions from our action classes.

The following components had been introduced:

1) Action Classes with one or more primitives each
    a. ADTs for every primitive consisting of HEADER and SEQUENCE OF ACTION CHUNKS with several chunks
        i. HEADER with basic action information.
        ii. SEQUENCE OF ACTION CHUNKS with action relevant parameters for each

All ADT parts come with their own success specifiers, which can be used to trigger corrective actions. ADTs are coded in OWL.

All (required) information will have to be entered – manually or automatically – into the ADTs (see Section 11).

In the next sections we will introduce the starting point for the Ontology.

10. Ontology

In Fig. 2 the conceptual model of the text-based ontology is presented. The presented action ontology model assigns an appropriate action synset for each action as well as action details required for execution that can be obtained from textual sources. The action synset contains verbs, prepositional verbs and phrasal verbs, having the same sense. We divide actions into action classes as described in section 4.

Each action can be parameterized with modifiers, which can affect action execution speed (e.g. wash slowly, wash quickly), method (e.g. wash carefully, wash safely), repeating sequence (e.g. wash once, wash twice, wash repeatedly, wash again) or the sequence of steps (e.g. wash first, wash then). We plan to include this information as action properties, but at the current stage we are not yet decided on the exact structure of property definition and only some of those properties, like "Time" and "Movement trajectory" were included in the current version of the conceptual model. Note, here we talk only about information coming from textual sources, and corresponding (complementary) information coming from robotic implementations will be provided in the ADTs described above.

In the current version of the conceptual model each action class has the following set of environment elements: objects, tools, time span, location, object materials, etc. (see Fig. 2). The process of extracting the environment elements is organized in three steps: 1) text preprocessing and building the glossary of possible environment elements; 2) obtaining rules (search patterns) for action environment element classification; 3) classifying the action environment elements by their roles. Each action environment element can be described by its properties. Defined ontology formalizes knowledge from various e-learning resources: online books, scientific literature, etc.

The list of the most common verbs for the chemistry domain (as obtained in the collected text corpus for the CHEMLAB scenario of ACAT) can be divided into three main groups of verbs: specific actions from chemistry laboratory domain (e.g. distill, extract, cool, reduce), verbs describing part of actions (e.g. place, reach, take), multiword actions (e.g. make distillation, get cool, start stirring). In the ACAT project will make an attempt to transform more complex, domain-specific action verbs into a sequence of simpler actions that have been described in section 4.

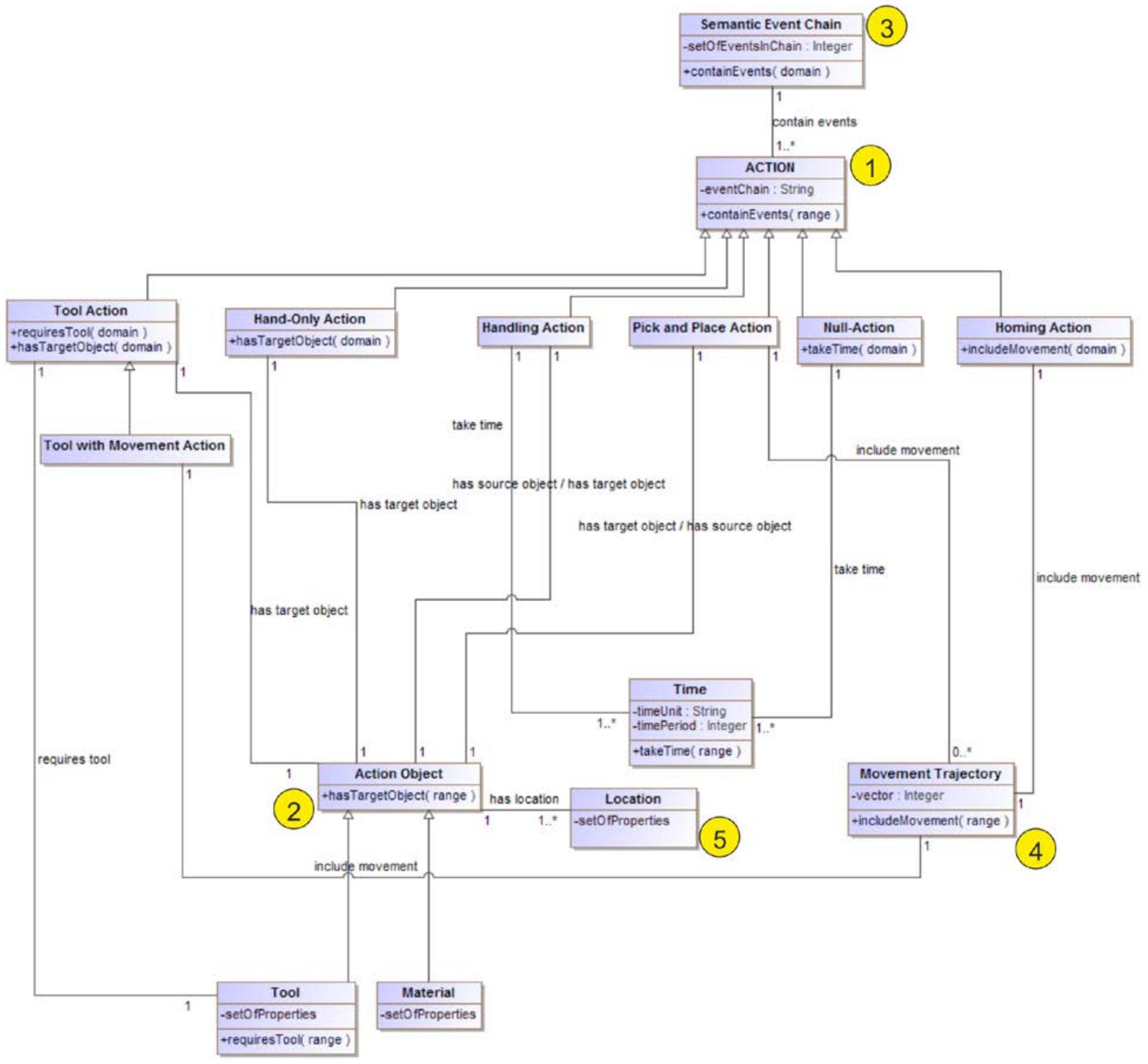


Fig. 2. The conceptual model of the ACAT text-based ontology. The numbers show link points to the ADT model (see Fig. 1)

11. Linking the text based ontology and Action Data Tables for addressing the ACAT problem

ADT and text-based ontologies are presented in Fig. 1 and Fig. 2. By corresponding numbers in those figures we show the links between the two models. These links will be used for applying reasoning throughout the two structures. Ontologies can be merged both, on the conceptual model level and by considering individual action instances. Merging on the conceptual level involves processes that can be used for checking the similarity of concepts. Comparing instances depends on checking the similarity of the instances through textual string metrics and semantic knowledge. In Figs 1,2 we show only the links at the conceptual model level.

In the following we give an example how the structures defined above (ADTs and text-based ontologies) can be used for execution of new instructions previously not executed by a robot.

In the next stage of the ACAT project a set of Action Description Tables (ADT) will be filled. This includes learning by demonstration, learning from simulation, as well as simple pre-programming, when necessary. When – through such processes – the ACAT system has accumulated a large enough action set (e.g. has filled in data structures for 5 or 10 instructions), generalization and re-use procedure can be started. If the new instruction is "similar enough" to the instructions for which the ADTs have been filled, we can try to re-use information from those tables. The approach for information from ADT re-use is illustrated in Fig. 3.

Here we give an example, which is just illustrative, to show the essence of information re-use we are currently implementing.

Let us assume that ATDs exist for the following instructions. (We add "K" to the numbers below to indicate this is a "known" instruction):

K1) Place the rotor cap on top of the fixature;
K2) Remove the test-tube from the holder;
K3) Shake the bottle;
K4) Insert the test-tube into the centrifuge;
K5) Remove the magnet from the magnet holder.
K6) Put a rotor cap on the table

Let us assume that the new instructions that robot needs to execute (and never has executed before) are the following ("N" for "New"):

N1) Insert the magnets into magnet-holes
N2) Remove the rotor from the press.
N3) Put a bottle on the sideboard.

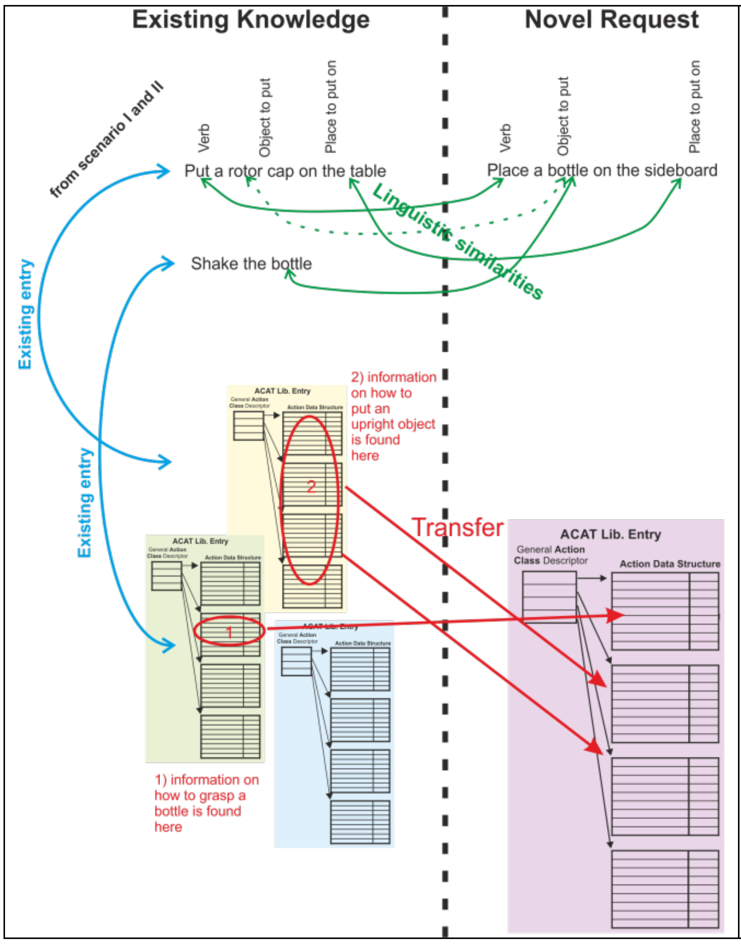
We can now consider and explain several possible aspects of generalization and re-use. Re-use is implemented by exploiting language similarities, as well as by cross-inferences.

Cross-inference can take the shape of one-to-one comparisons of following kind: If the object was ever grasped following any known instruction and if the geometrical shape of the new object (in a new instruction) is similar enough (by some metric) then the system can attempt to re-use the previously used grasp (with not too many re-parameterizations). While this was just one possible way for re-use based on cross-inference, other similar inferences can be considered, too.

If for any new instruction there is more than one known instruction that relate to it we can try to select the one which is most similar considering all aspects in the ADTs as well as with respect to language (ontological) analyzes.

In general we find that re-use can be centered on the action verb in the instruction. If we e.g. need to execute a new instruction N2: "Remove the rotor from the press", and we have two known instructions K1 and K5, where "remove" was performed, we can select the most similar one and then it may be possible to infer sub-aspects, e.g. trajectories, etc.

Not all the actions suggested by the given replacement scheme will be exactly correct. Here we plan designing procedures how a human operator can be included into the decision process, where human either accepts suggestions given by the systems or does tuning of the suggestions provided by the system.



Existing Knowledge
Novel Request
from scenario I and II
Verb
Object to put
Place to put on
Put a rotor cap on the table
Place a bottle on the sideboard
Linguistic similarities
Shake the bottle
Existing entry
Existing entry
2) information on how to put an upright object is found here
ACAT Lib. Entry
General Action Class Descriptor
Action Data Structure
Transfer
1) information on how to grasp a bottle is found here

Concretely:

1) **Existing Knowledge:** Assume that several such ADTs have been designed (left side of figure).
2) **Novel Request Processing:** Give a novel action request (place bottle on sideboard).
    a. **Enquire text-based ontologies** for information for similarities between existing data and novel request (top part of figure).
    b. **Transfer** all similarity-confirmed data into the novel-request data structure (red arrows, not all shown).
    c. **Correct and complete** filling in the novel-request data structure by any means (not displayed).

Fig. 3. General scheme of information re-use from the Acat database.

12. Conclusions

Background knowledge for a robot comes in two forms: symbolic knowledge and action execution (control level) knowledge.

Symbolic knowledge is provided in the form of ontologies and action execution level knowledge is provided in the form of action description tables, containing parameters required for execution. The set of parameters needs to be a compromise between different groups, as currently different robotic systems (also in different partner groups) use different ways to describe robotic execution. Here we use Semantic Event Chain based chunking and task space description as general grounds. However, the data structures are not final and the development of those data structures will continue throughout the ACAT project.

We have marked the linking points between text-based ontology and the action execution level knowledge (ADTs), however more linking possibilities may emerge in future during the testing and ongoing development of the here suggested action structures, both on textual and control ends.

References

Aksoy, E.E., Abramov, A., Dellen, B., Ning, K., Dörr, J. and Wörgötter, F. (2011) Learning the semantics of object action relations by observation. Int. J. Robotics Res. (IJRR), 30, 1229-1249.

Drummond, N.; Rector, A.; Stevens, R.; Moulton, G.; Horridge, M.; Wang, H. & Sedenberg, J. (2006), Putting OWL in Order: Patterns for sequences in OWL. , *in* 'OWL Experiences and Directions (OWLEd 2006).

Ijspeert, J.A., Nakanishi, J. and Schaal, S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots, in Proc. 2002, IEEE Int. Conf. Robotics and Automation, 1398–1403.

Kulvicius, T., Ning, K., Tamosiunaite, M. and Wörgötter, F. (2012) Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. IEEE Trans. on Robotics (IEEE T-RO) 28(1), 145-157.

Appendix: OWL code for selected action descriptions

Here we show the OWL code for the first two entries LOCATE K.1.1 and PICK UP K1.2

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY list "http://www.co-ode.org/ontologies/list.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<rdf:RDF xmlns="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#"
     xml:base="http://www.semanticweb.org/lnd/ontologies/2013/10/acat"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:list="http://www.co-ode.org/ontologies/list.owl#">
    <owl:Ontology rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat">
        <owl:imports rdf:resource="http://www.co-ode.org/ontologies/lists/2008/09/11/list.owl"/>
    </owl:Ontology>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasFunction -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasFunction">
        <rdfs:domain
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#ActionChunk"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Function"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasObject -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasObject">
        <rdfs:domain
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#ActionChunk"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasPosition -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasPosition">
        <rdfs:domain
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#SemanticEvent"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasSEC -->

    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasSEC">
        <rdfs:domain
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#ActionChunk"/>
        <rdfs:range
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#SemanticEventChain"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasSourceObject -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasSourceObject">
        <rdfs:domain
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#ActionChunk"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object"/>
        <rdfs:subPropertyOf
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasObject"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasStartPosition -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasStartPosition">
        <rdfs:subPropertyOf
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasPosition"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasTargetObject -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasTargetObject">
        <rdfs:domain
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#ActionChunk"/>
        <rdfs:range rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object"/>
        <rdfs:subPropertyOf
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasObject"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasTargetPosition -->

    <owl:ObjectProperty
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasTargetPosition">
```

```
        <rdfs:subPropertyOf
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasPosition"/>
    </owl:ObjectProperty>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Data properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#objectDescriptor -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#objectDescriptor">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#objectLink -->

    <owl:DatatypeProperty
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#objectLink">
        <rdfs:domain rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Classes
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Action -->

    <owl:Class rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Action"/>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#ActionChunk -->

    <owl:Class rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#ActionChunk">
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasFunction"/>
                <owl:onClass
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Function"/>
                <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasSEC"/>
                <owl:onClass
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#SemanticEventChain"/>
```

```
                <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasSourceObject"/>
                <owl:onClass
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object"/>
                <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasTargetObject"/>
                <owl:onClass
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object"/>
                <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Function -->

    <owl:Class rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Function"/>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#LocateObject -->

    <owl:Class rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#LocateObject">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#ActionChunk"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasTargetObject"/>
                <owl:hasValue
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Void"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object -->

    <owl:Class rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#objectDescriptor"/>
                        <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
                        <owl:onDataRange rdf:resource="&xsd;string"/>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#objectLink"/>
                        <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
                        <owl:onDataRange rdf:resource="&xsd;string"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpFunction -->

    <owl:Class rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpFunction">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Function"/>
    </owl:Class>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpObject -->

    <owl:Class rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpObject">
        <rdfs:subClassOf
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#ActionChunk"/>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasTargetObject"/>
                <owl:hasValue
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Void"/>
            </owl:Restriction>
        </rdfs:subClassOf>
    </owl:Class>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#SemanticEvent -->

    <owl:Class rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#SemanticEvent">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasStartPosition"/>
                        <owl:onClass
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector"/>
                        <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#hasTargetPosition"/>
                        <owl:onClass
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector"/>
                        <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#SemanticEventChain -->

    <owl:Class
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#SemanticEventChain">
        <owl:equivalentClass>
            <owl:Class>
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="&list;OWLList"/>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="&list;hasContents"/>
                        <owl:allValuesFrom
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#SemanticEvent"/>
                    </owl:Restriction>
                    <owl:Restriction>
                        <owl:onProperty rdf:resource="&list;isFollowedBy"/>
                        <owl:allValuesFrom
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#SemanticEventChain"/>
                    </owl:Restriction>
                </owl:intersectionOf>
            </owl:Class>
        </owl:equivalentClass>
    </owl:Class>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector -->

    <owl:Class rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector"/>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Individuals
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->




    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#LocateRotorCap -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#LocateRotorCap">
        <rdf:type
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#LocateObject"/>
        <hasSourceObject
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#RotorCap"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpRotorCap -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpRotorCap">
        <rdf:type
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpObject"/>
        <hasFunction
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpRotorCapFunction"/>
        <hasSEC
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapSEC1"/>
        <hasSourceObject
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#RotorCap"/>
        <hasTargetObject
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Void"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpRotorCapFunction -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpRotorCapFunction">
        <rdf:type
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickUpFunction"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapEvent1 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapEvent1">
        <hasStartPosition
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector1"/>
```

```
        <hasTargetPosition
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector1"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapEvent2 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapEvent2">
        <hasTargetPosition
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector1"/>
        <hasStartPosition
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector1"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapSEC1 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapSEC1">
        <list:hasContents
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapEvent1"/>
        <list:hasNext
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapSEC2"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapSEC2 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapSEC2">
        <list:hasContents
rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#PickupRotorCapEvent2"/>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#RotorCap -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#RotorCap">
        <rdf:type rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object"/>
        <objectDescriptor rdf:datatype="&xsd;string">&quot;CAD for rotor
cap&quot;</objectDescriptor>
        <objectLink rdf:datatype="&xsd;string">&quot;Rotor Cap&quot;</objectLink>
    </owl:NamedIndividual>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector1 -->

    <owl:NamedIndividual
rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Vector1"/>



    <!-- http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Void -->

    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Void">
        <rdf:type rdf:resource="http://www.semanticweb.org/lnd/ontologies/2013/10/acat#Object"/>
        <objectDescriptor rdf:datatype="&xsd;string">&quot;&lt;void&gt;&quot;</objectDescriptor>
        <objectLink rdf:datatype="&xsd;string">&quot;&lt;void&gt;&quot;</objectLink>
    </owl:NamedIndividual>
</rdf:RDF>



<!-- Generated by the OWL API (version 3.4.2) http://owlapi.sourceforge.net -->
```