

8<sup>th</sup> November 2013

Contract number: 287624

Dissemination Level: PU



# DELIVERABLE 3.3

## First Design and Technical Implementation of Computational Memory Architecture

**Author(s):** Joe Saunders, Kerstin Dautenhahn

**Project no:** 287624

**Project acronym:** ACCOMPANY

**Project title:** Acceptable robotiCs COMPanions for AgeiNg Years

<b>Doc. Nature (report, ...)</b>	Final Report
<b>Version</b>	4.0
<b>Actual date of delivery</b>	8 <sup>th</sup> November 2013
<b>Contractual date of delivery</b>	30 <sup>th</sup> September 2013
<b>Project start date</b>	1 <sup>st</sup> October 2011
<b>Project duration</b>	36 months
<b>Peer Review</b>	UVA

Project Acronym: ACCOMPANY

Project Title: **Acceptable robotiCs COMPanions for AgeiNg Years**

EUROPEAN COMMISSION, FP7-ICT-2011-07, 7th FRAMEWORK PROGRAMME  
ICT Call 7 - Objective 5.4 for Ageing & Wellbeing

Grant Agreement Number: 287624



## DOCUMENT HISTORY

Version	Date	Status	Changes	Author(s)
1.0	2013-10-12	Draft	First version	Joe Saunders
2.0	2013-10-21	Draft	Revisions by KD	Kerstin Dautenhahn
3.0	2013-08-11	Draft	Revisions after review from UvA	Joe Saunders

## AUTHORS & CONTRIBUTORS

Partner Acronym	Partner Full Name	Person
UH	University of Hertfordshire	Joe Saunders, Kerstin Dautenhahn
Review:	University of Amsterdam	Ben Kröse

## Table of Contents

<b>AUTHORS &amp; CONTRIBUTORS.....</b>	<b>2</b>
<b>Table of Contents .....</b>	<b>3</b>
<b>SHORT REPORT.....</b>	<b>5</b>
<b>Contents .....</b>	<b>6</b>
<b>1 Introduction.....</b>	<b>6</b>
<b>2 Background.....</b>	<b>6</b>
2.1 Re-ablement and Co-learning .....	7
2.2 Challenges.....	8
<b>3 Design and Implementation of the Computational Memory Architecture .....</b>	<b>8</b>
3.1 Existing Approaches for Robot Control Architectures .....	9
3.2 The Robot, the House, People and the Environment.....	9
3.2.1 Semantic memory – the ontology of the Robot House.....	10
3.2.2 Sensors.....	10
3.2.3 Sensor Abstraction .....	12
3.2.4 External Sensors and External Actions .....	13
3.2.5 Robot Capabilities.....	13
3.3 Procedural Memory - Behaviour Creation and Execution .....	15
3.3.1 Behaviour Creation.....	15
3.3.2 Specialist Planning .....	19
3.3.3 Behaviour Execution and Scheduling.....	19
3.3.4 Resource Management and Dealing with Conflicts.....	21
3.4 Dealing with Time .....	22
3.4.1 Temporal Relationships .....	22
3.5 Robot Teaching .....	24
3.5.1 Teaching Interfaces .....	24
3.5.2 Using the teaching system to create graphical interfaces between the user and the robot	25
3.5.3 Semi-technical Teaching Interface .....	25
3.5.4 Example of the Semi-Technical Teaching facility.....	26

3.5.5	User Teaching Interface .....	27
3.5.6	Example of the User Teaching facility .....	30
<b>4</b>	<b>Capabilities and Assessment.....</b>	<b>33</b>
4.1	Capability Assessment .....	33
4.2	Evaluation Criteria .....	35
<b>5</b>	<b>Usage of the CMA in Experimental Scenarios.....</b>	<b>38</b>
5.1	Other Users of the CMA .....	39
<b>6</b>	<b>Future Evaluation of the Teaching Component .....</b>	<b>39</b>
<b>7</b>	<b>Future Developments to Support learning .....</b>	<b>40</b>
<b>8</b>	<b>Appendix .....</b>	<b>45</b>
8.1	Complete set of Sensors .....	45
8.2	Typical Behaviours that have been tested with the UserGUI .....	48
8.3	Ancillary programs associated with Behavioural setup and Execution .....	49
8.3.1	Action Possibility Creator.....	49
8.3.2	Behaviour Viewer.....	50
8.3.3	Message Creator .....	50
8.3.4	House Simulators .....	51
8.3.5	Session Control .....	51

## SHORT REPORT

This document provides details of the implementation and design details of the initial computational memory architecture that has been implemented on the Care-O-Bot3<sup>®</sup>, the companion robot in the ACCOMPANY project.

We report on two major aspects of the proposed Care-O-Bot3<sup>®</sup> memory model which have been developed: the Semantic and Procedural Memory supporting the environmental ontology associated with the robot's environment and the behavioural control and scheduling mechanisms that allow for behaviour creation and execution on the robot.

One of the aims of the ACCOMPANY project is to provide aid and support for elderly people and allow them to reside for longer in their own homes by providing facilities not only for care, but also for companionship. It is envisaged that companionship would be supported by the dual functions of *co-learning* and *re-ablement*. In our previous deliverable (D3.2) we specified how our computational memory architecture was planned to support episodic, procedural and semantic aspects of memory and described the technical implementation of the *episodic* part of the architecture. In this report we focus especially on the design and implementation of the *semantic* and *procedural* aspects and how these are combined to form the behavioural control system for the Care-O-bot3<sup>®</sup> robot.

This document reports on 1) the background to this research and a definition of re-ablement and co-learning, 2) an explanation of the architecture itself, together with a general description of the environment, the robot and our ontological approach, 3) how the procedural aspects of memory and especially how behaviour execution, behaviour planning, resource management and temporal aspects of behaviours are supported, 4) how far the memory architecture meets the general requirements of a cognitive architecture and how it can be evaluated and assessed, 5) an outline of our experience of using the architecture in our experimental scenarios and our plans for formative evaluation, 6) the future extensions of the architecture to be carried out in the final phase of the project (Task 3.4), which concerns learning and adaptation.

## Contents

### 1 Introduction

One of the aims of the ACCOMPANY project is to provide aid and support for elderly people and allow them to reside for longer in their own homes by providing facilities not only for care, but also for companionship. It is envisaged that companionship would be supported by the dual functions of *co-learning* and *re-ablement*. In our previous deliverable (D3.2) we specified how our computational memory architecture was planned to support episodic, procedural and semantic aspects of memory and described the technical implementation of the *episodic* part of the architecture. In this report we focus especially on the design and implementation of the *semantic* and *procedural* aspects and how these are combined to form the behavioural control system for the Care-O-bot3<sup>®</sup> robot.

In order provide details of the implementation we divide the report into a number of sections describing: firstly, the background to this research and a definition of re-ablement and co-learning, secondly, the architecture itself, together with a general description of the environment, the robot and our ontological approach, thirdly, how the procedural aspects of memory and especially how behaviour execution, behaviour planning, resource management and temporal aspects of behaviours are supported, and finally how far the memory architecture meets the general requirements of a cognitive architecture and how it can be evaluated and assessed. We conclude this document by outlining our experience of using the architecture in our experimental scenarios and provide our plans for formative evaluation. The future extensions of the architecture to be carried out in the final phase of the project (Task 3.4), which concerns learning and adaptation, are then described.

### 2 Background

It is predicted that many countries worldwide are facing a demographics problem over the following decades. This is due to increasing life expectancy, leading to more elderly persons, combined with a decrease in the proportion of younger people providing support to the elderly. For example, it is predicted that in the European Union the number of people over 65 years will almost double (by 2060) and the number of people between 15-64 years will decrease by over 10% (Eurostats, 2013}. Health care costs are also rising (Przywara, 2010), therefore there has been a focus on the further use of assistive and robotic technologies as one possible solution to this issue.

Robotic companions have been suggested as an assistive technology to meet an ever ageing population. Our approach utilises a commercially available robot, the Care-O-bot3<sup>®</sup>, manufactured by Fraunhofer IPA (Reiser et al., 2009) sited in a fully sensorised house (a smart home which we call the *robot house*). The house itself is a normal British three bedroom house near the University of Hertfordshire and it was specifically chosen to be a realistic home environment rather than a modified scientific laboratory. One of the many challenges faced in such an environment is to provide a robotic companion that is not only useful as a physical and memory prosthetic but also provides active support in terms of *re-ablement* and *co-learning*.

## 2.1 Re-ablement and Co-learning

A concise definition of *re-ablement* is given by the Welsh Social Services Improvement Agency as ‘‘Support people ‘to do’ rather than ‘doing to / for people’ ’ (Welsh Social Services Improvement Agency, 2006). We envisaged that typical behaviours for a house robot might be, for example, to act as a memory aid and issue reminders, e.g. for medicine, to assist in fetch and carry tasks, and to provide monitoring facilities both for the house and for the person e.g. warning them that the fridge door has been left open or reminding them to take a drink if the robot has noticed nothing has been drunk for a long period. However in addition to the above, and to support re-ablement, the robot needs to provide motivation for the user to be more active, both physically and mentally. Thus the robot, rather than offering to do something *for* the user, may suggest that the person and the robot carry a task out *together* or that the user *does it themselves*. For example, the robot might suggest that contact is made with friends, bring relevant events (such as birthdays) to the user’s attention or even suggest that the user take a walk. These latter forms of co-operation designed primarily to attempt to avoid social isolation of the individual in the home.

*Co-learning* refers to where a person and a robot work together to achieve a particular goal. Typically a robot can provide help and assistance, but in return may also require help and assistance. While robot companion technology can be expected to advance significantly and become increasingly autonomous and competent over the next few years, robots will still have limitations in dealing with all the changes happening in a person’s life and living environment, and such learning and adaptation can be achieved with the help of the user. The user will typically teach the robot how to solve a problem, however the robot can also assist by suggesting to the person that it has particular capabilities which may prove fruitful (or indeed that it already knows how to address this particular problem). We believe that the co-learning approach will be useful as rather than treating the robot solely as a support mechanism for an

elderly person, and possibly disenfranchising the person from the problem being solved, both the robot and person find a solution together. The elderly person remains at the heart of the problem solving exercise and we want to avoid the person feeling 'bossed around' by a robot. A robot that needs help itself makes the relationships with the elderly person more symmetric in social terms, which is hoped to facilitate a person's interest and engagement in using the technological help

Thus the user-robot relationship is one of mutually beneficial support, assistance and companionship. Achieving these aims presents many issues for a companion robot. Simple scripting of interactions will not achieve the above aims due to the dynamics of the interaction and the key requirement of the robot to develop and learn.

## 2.2 Challenges

In order to achieve the aims outlined above we faced a number of challenges. Firstly, the disciplined integration of the robot house sensor network, the sensory capabilities of the robot itself, and the social memory aspects from the user themselves, into a common framework. We can call this step the creation of a 'robot house' ontology describing both the physical and semantic nature of the system as a whole. Secondly, a mechanism which allowed activities within the house, at both a sensory level and a more abstract contextual level to be joined as propositions (held as rules or preconditions) for resulting robot behaviours. Thirdly, a mechanism that would allow us to apply temporal constraints, where necessary, to such propositions. Fourthly, a facility to invoke actions on the robot, at both a primitive/actuator level or a more distant abstract level. Finally, the ability to be flexible in behaviour creation and scheduling. Given that our robot may be asked to carry out a large number of tasks, many of which may not be originally envisaged by the system designer, a flexible and 'easy to use' way of creating robot behaviours together with a mechanism for effectively scheduling such behaviours was required. Our goal was to make such facilities available to non-technical personnel such as the elderly persons themselves, carers or relatives.

## 3 Design and Implementation of the Computational Memory Architecture

In this section we describe our approach to designing the Computational Memory Architecture (CMA) together with a general description of the related work, the robot house and robot environment, sensors and sensor abstraction.



### 3.1 Existing Approaches for Robot Control Architectures

There are many and varied approaches to robot control architectures operating in complex and partially observable situations. Arkin extensively describes both behaviour based and deliberative approaches (Arkin, 1998) and provides a spectrum (from purely reactive to completely deliberative) of control issues that result. Many robot architectures also exploit research within the sphere of cognitive architectures (for a detailed survey see (Langley, Laird, & Rogers, 2009)). The need to cope in a timely fashion within uncertain environments, which would be typically experienced by robots in domestic and care environments, both reactive (Firby, 1987; Nilsson, 1994), planning approaches (Weser, Off, & Zhang, 2010; Off & Zhang, 2011) and combined approaches (often called three-level architectures (Gat, 1998; Martens, Prenzel, & Graser, 2007; Simmons & Apfelbaum, 1998)) have been attempted. Our approach takes inspiration from these and other approaches and attempts to combine cognitive aspects with ease of teaching within one integrated environment.

All of the approaches noted above have in common the need to link the environment with the robot, and typically to maintain some kind of knowledge representation layer. Robot actions and decisions must be made in real-time and the robot must have a way of deciding which actions to take in a given situation, sometimes when more than one set of actions are possible.

Abstraction of perception is also needed which, rather than focussing on low level sensory information, either experienced by the robot, or from a sensorised house (such as the robot house), the architecture should instead offer predictions of activity (e.g. making lunch). In our framework we describe a three-layer approach, based on a memory framework, where episodic, semantic and procedural memories can be constructed and implemented. Both reactive, temporal and state of the art planning elements (Nau, Cao, Lotem, & Munoz-Avila, 1999) are included in a uniform design.

### 3.2 The Robot, the House, People and the Environment

In ACCOMPANY we consider a typical care environment to be one where a person or persons remain in their own home. Our physical home ontology is modelled within a MySQL database, and all information arising from robot, house sensors (including abstract sensors) and user or robot locations in the physical home causes a real-time update to the database (see Figure 1 for a partial view of the database). We also, however, appreciate that such an approach is only one method for encoding such complexity. Other approaches (see for example, (RACE, 2011)), encode semantic information using RDF (Resource Description Framework) and ontological

information using OWL (Web Ontology Description Language). Although this approach provides some inferencing facilities, from our point of view, the use of a SQL database has the benefit of providing access to many tools for fast and relatively easy development, and is readily understood between project partners (who may not necessarily be experts in semantic web technologies).

One of the advantages of using a centralised information store is that house episodes can be modelled and tested without actually needing to be in the house by simply updating the sensor tables in the database. Episodic information, which consists of both images and sensory feedback during behaviour execution, can be accessed via GUI's allowing post-review of activities of the robot and the user (this was described in Deliverable D3.2). Procedural memory, which is here defined as the robot actions together with rules invoking such actions held as pre and post behavioural conditions, are also held as tables in the database. The rules themselves are encoded as SQL statements, generated by the behaviour teaching system, and refer back to the semantic information created by the sensor system.

### 3.2.1 Semantic memory – the ontology of the Robot House

We regard the house as one entity rather than as a collection of individual parts. In practise this means that the house sensor information is considered to be no different from robot sensor information, the sensory information derived from the occupants activities or from sensory abstractions (described in section 3.2.3). This provides the bedrock for the main focus of our work enabling co-learning and re-ablement by not artificially separating the robot, user (or indeed the house) as separate entities but rather focus the generation of behavioural activity (which in our case is via the robot, but in an ambient home could be via actuation of household devices) on the complete system.

This approach requires a number of pre-requisites, firstly a disciplined method of maintaining both current and historical sensory information, secondly, a way of creating 'abstract' sensors which can use existing facilities for sensor interrogation and finally, a way of dealing with temporal aspects of sensory information (described in section 3.3.4).

### 3.2.2 Sensors

The home itself consists of physical sensors, user and robot locations and objects. In the University of Hertfordshire robot house there are over 50 'low level' sensors ranging from electrical (fridge door open, microwave on etc.), to furniture (cupboard door and drawers open

etc.), to services (such as toilet flushing, taps running etc.) and pressure devices (sofa or bed occupied). Sensory information from the robot is sent to the database or for high throughput, is acquired via ROS messaging (Willow Garage, n.d.). User locations are obtained via ceiling mounted cameras (part of work package 4) and described in (Hu, Englebienne, & Krose, 2012), and robot locations via ROS navigation (Willow Garage, n.d.). House locations are labelled and encoded as map co-ordinates in the database and organised hierarchically, for example, ‘sofa location’ is part of the ‘living room’ which is part of the ‘robot house’. Each sensor also has ancillary information concerning how it should be interpreted, For example, a bathroom tap sensor value is interpreted as a moving average of values, whereas a light switch is simply a boolean interpretation. The full set of sensors and their interpretations are shown in the Appendix 8.1.

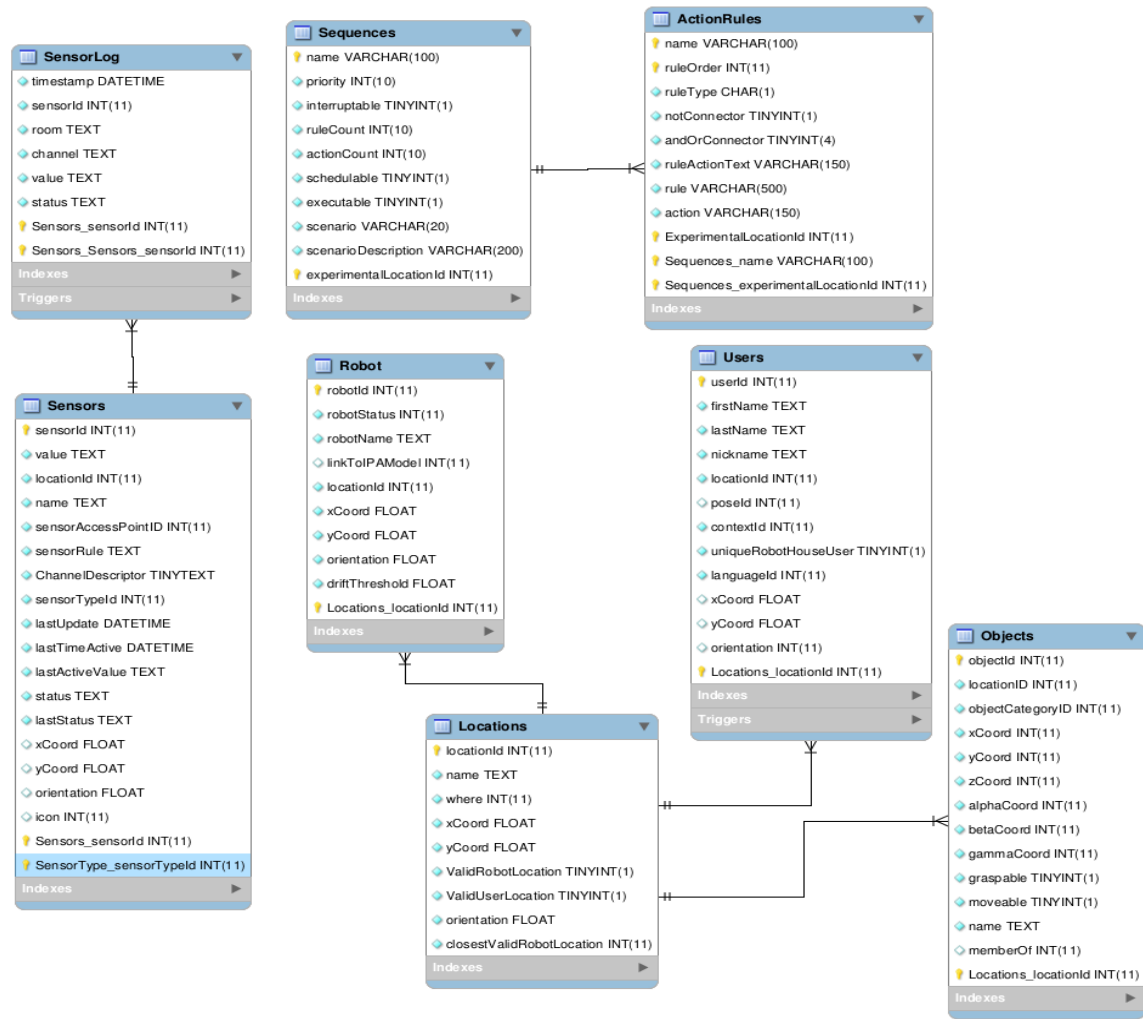


Figure 1. A partial view of the robot house ontology represented in a MySQL database. Three sets of tables are shown. The first, shown on the left, is a sensor log and sensor table containing current and historical sensory information. The second, at the top, are the main tables used for behavioural encoding. The bottom set of tables show the relationships between robot, users, objects and locations. The actual database contains considerably more detail and complexity with around 50 tables in total.

### 3.2.3 Sensor Abstraction

*Abstract* sensors are used to define or infer knowledge about the house or activities within the house at a higher semantic level. We define two classes of abstract sensors. The first we call *context* sensors. Context sensors are updated via a rule based context analysis system derived

from HRI experiments (described in (Duque, Dautenhahn, Koay, Willcock, & Christianson, 2013)). This provides contextual information such as, e.g. ‘User Preparing Evening Meal’ . Thus the sensor ‘User Preparing Evening Meal’ would be set ‘on’ if a given set of propositions were true (for example, fridge has been opened recently, it is after 4pm, kitchen light is on etc.) and set ‘off’ otherwise. Secondly, in order to cope with on-going events in the house which are not reflected by the environmental sensors a set of ‘predicate’ sensors can be created by the teaching system. Predicate sensors typically reflect on-going *actions* and are used in order to cope with events in the house which are not reflected by the physical environmental sensors. For example, a sensor with the label ‘User has been reminded to turn off the TV’ might be set if the robot has given such a reminder to the user typically and would be used to ensure that the reminder was not repeated.

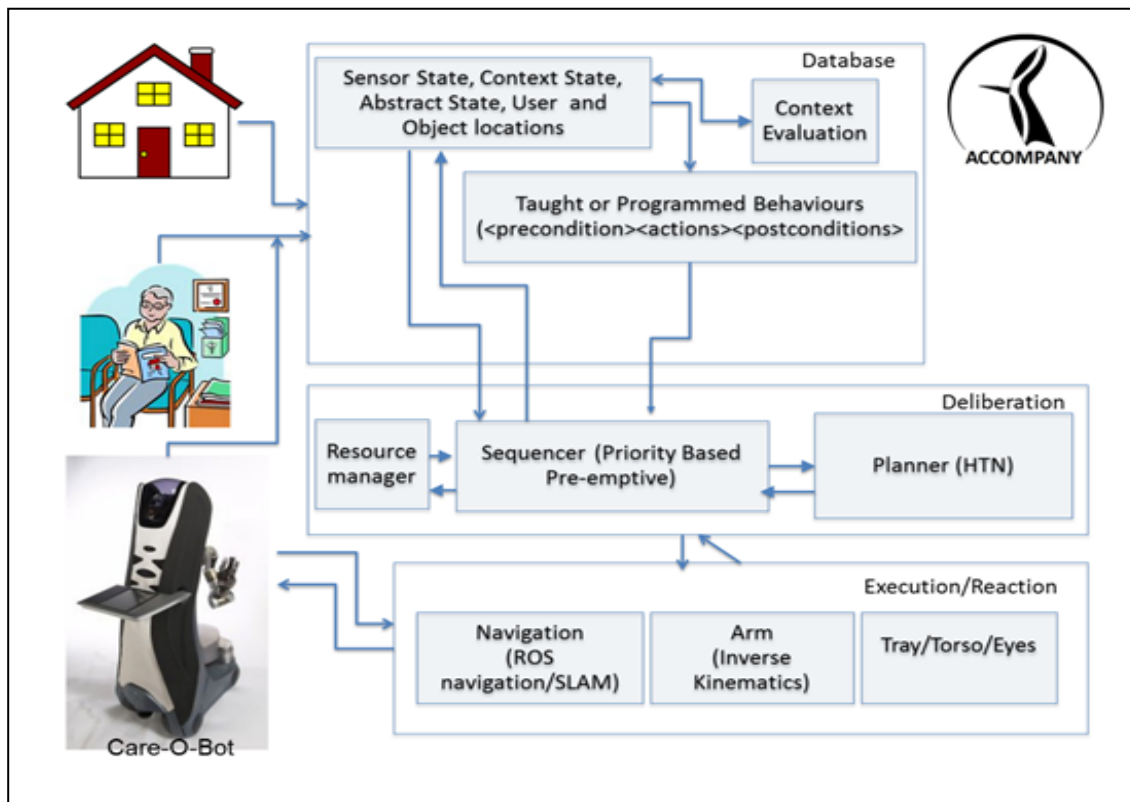
### 3.2.4 External Sensors and External Actions

The sensor system (both physical and abstract) provides a standardised way of creating and encoding information and therefore gives the possibility of associating sensors with other, possibly external, events. For example, by polling an external weather service it would be possible to set the value of a previously created abstract sensor. This could then be checked by a behaviour which might suggest to the user that this was a good day for a walk, or to do some gardening. This is one way that the idea of re-ablement can be operationalised.

External actions can also be executed, e.g. calling a text messaging (SMS) service. For example, a behaviour that checks whether the bed pressure sensor had been active for more than 12 hours and that there had been no activity in the kitchen might then send a text message to the users’ careworker suggesting that the person might need assistance to get out of bed.

### 3.2.5 Robot Capabilities

For this work we use the Care-O-bot3<sup>®</sup> robot (Reiser et al., 2009) (see picture in figure 2) which uses ROS navigation (a form of SLAM) (Willow Garage, n.d.) using its laser range-finders to update a map of the house in real-time and can thus navigate to any given location whilst avoiding obstacles and replanning routes. Similarly the robot is equipped with facilities for manipulating the arm, torso, ‘eyes’, robot LED’s, tray and has a voice synthesiser to express given text. High level commands are sent via the ROS ‘script server’ mechanism and interpreted into low level commands by the robot software. Typical commands would be for example, ‘raise tray’, ‘nod’, ‘look forward’, ‘move to location x’, ‘grab object on tray’, ‘put object x at location y’, ‘say hello’ etc.



**Figure 2 :** A high level view of the control architecture for the Care-O-bot R within the robot house. The architecture is based on a typical three-layer approach comprising of deliberation (sense/plan), execution and reaction (act). The deliberative layer makes decisions based on real-time information from the central database. This information is composed of updates from the physical house sensors and updates made via the deliberation or context awareness rule-based programs to abstract sensors. The reactive layer operates primarily in a tight control loop in areas such as navigation and arm kinematics.

### 3.3 Procedural Memory - Behaviour Creation and Execution

One of our major goals was to be able to generate and execute behaviours on the robot. Generation of behaviours is considered to be a task that would be on-going, due to changing living conditions, and be carried out by users of the system themselves. Behaviour encoding and behaviour execution are intimately tied and we have drawn inspiration from research on cognitive architectures (for a detailed survey see (Langley et al., 2009)) and especially behavioural control outlined in Freed (Freed, 1998) and Nilsson (Nilsson, 1994, 2001) together with work on Hierarchical Task Planners (HTN's) (see, for example, (Nau et al., 1999)).

Each of these approaches is based on what is sometimes called ‘sketchy’ or partial planning, whereby outline plans for execution are generated with the details gradually being filled in as the behaviour executes. Our behaviours operate in a similar way where conditions can be set which cause lower level behaviours, pre-set plans or operational primitives to execute. The lower level behaviours can in turn further refine the task requirements. At each step the current abstract and environment sensory states are checked so to ensure that the behavioural goal is still achievable.

#### 3.3.1 Behaviour Creation.

Behaviours are not pre-programmed, but instead *taught* to the system via one of two GUI interfaces (Saunders et al., 2013). The first GUI is a semi-technical interface allowing for direct access to all types of sensors and all types of robot actions (described in section 3.5.3), the second interface is more restricted and generates behaviours based on templates (described in section 3.5.5). This latter GUI trades behavioural generation complexity and expressiveness against ease of use and it is the interface which we would expect to be used by end users and the one which will be formatively evaluated during the project.

Behaviours that are generated follow the familiar pattern (similar to Nilsson's T-R formalism (Nilsson, 1994)) of evaluating propositions (as *pre-conditions*), followed by execution of robot *actions* and updating of *post-conditions* (in a similar way to the add/delete lists of planning systems). Pre-conditions can evaluate any form of sensory information including both real sensors (set by environmental changes) and abstract sensors (context/predicate sensors set by post-condition updates). Post-condition updates are used to change abstract sensors, however physical sensors cannot be directly updated. An example of a behaviour would be as follows:

```
BEHAVIOUR: doorbell

IF the doorbell has rung in the last 20 seconds    (sensor pre-cond)
AND the user has not yet been informed             (predicate sensor pre-cond)
THEN
    send the robot to the user location in the house (action)
    make the robot say 'Someone at the door'        (action)
    update the database to signal that
        the user has been informed                 (update predicate sensor post-cond)
```

#### Example 1. A typical behaviour

If the set of preconditions are true, then the actions are executed, including the post-condition update. Actions, as well as being used to control the robot, can also be used to execute other behaviours, which again can have pre-conditions and actions. Sequential sets of actions can also be called, and these would be set-up as behaviours with no pre-conditions. Careful arrangement of behaviours allows the action of post-condition setting to fire other behaviours and thus ‘fill in’ the details of the sketchy plan.

Within each behaviour, each pre-condition is automatically encoded by the teaching GUI into SQL statements. See the example below:

```
IF the doorbell has rung in the last 20 seconds    (sensor pre-cond)
AND the user has not yet been informed             (pred. sensor pre-cond)

resolves to the SQL statements...

SELECT * FROM Sensors WHERE sensorId = 302 AND lastStatus = "On" AND
lastUpdate+INTERVAL 20 SECOND >= NOW()

SELECT * FROM Sensors WHERE sensorId = 701 AND value = 'notInformed'
```

#### Example 2. SQL Statements representing pre-conditions

The SQL statements shown in Example 2 check whether sensor number 302 (the doorbell) held on the sensor table was active (lastStatus = “on”) at any time within the last 20 seconds, and



whether sensor 701 (an abstract sensor called “user has been informed”) has the value ‘notinformed’.

Execution of each of the SQL statements will return a row if the conditions are currently true. Thus, if a row is returned, then that pre-condition is deemed to be true, otherwise false.

When building sets of pre-conditions the user, via the GUI, has the opportunity to link them with a conjunctive AND or a disjunctive OR. The conjunction or disjunction of statements results in a final boolean decision. If this final decision is true then the robot actions are executed. Example 3 shows the primitive actions that would be executed in the “doorbell” behaviour. Firstly, calling the navigation system to move the base, then making the robot say something and finally setting an abstract sensor.

base,0,[4.329:0.837:51],999,wait	(moves the robot to the user position)
speak,0,Someone at the door	(makes the robot say ‘someone at the door’)
cond,701,informed	(sets sensor 701 to the value ‘informed’)

**Example 3. The set of actions for the ‘doorbell’ behaviour**

A complete set of robot actions, including setting of post-conditions and calling other behaviours is shown below.

Action	Result
<b>Base &lt;desired location&gt;</b>	<i>Moves the base the desired location using ROS navigation. For multi-room movements this may call a separate SHOP2 planning domain.</i>
<b>Speak &lt;text&gt;</b>	<i>Makes the robot say &lt;text&gt;</i>
<b>Tray &lt;position&gt;</b>	<i>Moves the tray to a position. Typically up or down or folded.</i>
<b>Light &lt;colour&gt;</b>	<i>Set the robot LED’s to a particular colour.</i>
<b>Sleep &lt;seconds&gt;</b>	<i>Robot will pause for a number of seconds.</i>

<b>Torso &lt;movement&gt;</b>	<i>Make the robot torso move. Typical movements include “nod”, “shake”, “bow”.</i>
<b>Eyes &lt;position&gt;</b>	<i>Move the “eyes” forward or back.</i>
<b>Head &lt;position&gt;</b>	<i>Move the head to a particular position.</i>
<b>Arm &lt;position&gt;</b>	<i>Move the arm to a specified position e.g. “above tray”, “folded” etc. This will always call a SHOP2 planning domain to yield a plan of safe way-points.</i>
<b>GUI &lt;label&gt;</b>	<i>Present the UH-GUI with choices on the user tablet and await a response</i>
<b>Cond &lt;value&gt;</b>	<i>Set a abstract sensor to a particular value</i>
<b>APoss &lt;value&gt;</b>	<i>Set an action possibility for the SienaGUI to a particular value. The SienaGUI will display action possibilities if this value exceeds a threshold.</i>
<b>Expression &lt;value&gt;</b>	<i>Set the robot expression displayed on the SienaGUI to a value (e.g. happy, sad etc.)</i>
<b>Sequence &lt;behaviour name&gt;</b>	<i>Recursively calls the behaviour execution system to execute a behaviour.</i>

**Table 1. The list of actions that can be executed. Actions in blue are concerned with the user interface (either the SienaGUI or the UH-GUI). The action in red is the way in which other behaviours are fired.**

The actions shown in Table 1 are typically executed directly. However there are instances where a sub-plan would be invoked (for example, for the “base” command, if the move location was another room). This sub-plan would invoke a separate planning HTN module (SHOP2). A diagram of the overall system is shown in Figure 2.

### 3.3.2 Specialist Planning

Planning, in this context, means using specialist plans coded separately as SHOP2 planning domains (Nau et al., 1999)). Our general approach is to plan only when needed and when necessary. By “planning” we mean that we explicitly use planning domains created for some particular task. Where such detailed planning is required the appropriate planning domains therefore need to be created.

For example, if the Care-O-bot arm needs to be moved we need to construct a safe set of “way-points” between the arm start-location and the arm end-location. If we were simply to send the end-location to the robot controller, the result might be that the arm attempts to move through the body of the robot. To avoid this we construct, within a SHOP2 planning domain, the appropriate planning rules and fill the domain with current environment details. When the “arm” action is called we execute this planning domain to yield the safe set of way points.

We consider that creating planning domains to be too complex for end-user involvement and therefore we pre-code these. Note however that the overall behaviour of the system is driven primarily by the environmental conditions via environmental sensors or from context or predicate sensors by the behaviour execution system. Behaviours are explicitly scheduled. However, there are instances where more detailed planning is required (e.g. the “arm” example above or in a multi-room robot navigation).

Although such planning complexity *could be coded* as multiple behaviours within the existing execution framework of the CMA, this would lead to an overly complex set of behavioural rules. By separating these pre-coded planning domains for particular tasks, and calling them only as necessary within behaviours that require them, the overall complexity is reduced.

We use an open source state-of-the-art HTN (Hierarchical Task Network) planner (SHOP2 (Nau et al., 1999)) to cope with these situations (in fact we use the JSHOP2 java-based version). We follow the approach described by Hartanto (Hartanto, 2011) and Off and Zhang (Off & Zhang, 2011), in that each planning domain is individually coded in the lisp-like syntax of SHOP2 and called when the high level action is required. JSHOP2 returns the planning actions as robot actions. After each action execution we recall the planning component just in case the environment has changed between actions.

### 3.3.3 Behaviour Execution and Scheduling.

We use a priority-based pre-emptive scheduling system to control behaviour execution. Behaviours are created with a scheduling priority (as an integer) defined by the user. When the

robot is running, the scheduling system checks all of the preconditions of all of the behaviours (in a manner similar to Nilsson (Nilsson,2001)). If the combined (conjunction or disjunction) set of pre-conditions of a behaviour become true, then that behaviour *becomes available* for execution. Then, the highest priority behaviour is executed first (however see section 3.3.4 below). Priority ties result in a random choice of behaviour for execution. Note that due to continual checking of all behavioural pre-conditions, behaviours may become valid or invalid for execution as the currently executing behaviour operates. In this manner, the set of environment, context and predicate sensors drive behaviour execution. Some behaviours can also be set as non-interruptible, for example if a behaviour was reporting on a critical house event - such as the bathroom taps being on for a long time.

Name	Priority	Int	Exec	Behaviour	Rule/Action	Order	Description	And/Or	True/False
resetAPDrinkReminderIcon	91	No	No	s2-CheckUserDrinking	Rule	0	ZUYD Cup is Full AND has been in this state for more than 60 seconds	*	True
resetDrinkReminder	91	No	Yes	s2-CheckUserDrinking	Rule	1	::\$11: userShouldDrink is NotReminded	*	True
resetShouldDrinkReminder	91	No	No	s2-CheckUserDrinking	Rule	2	::\$19: cupLocation is Table	*	False
resetDoorBellReminded	91	No	No	s2-CheckUserDrinking	Action	3	SET ::\$11: userShouldDrink TO Reminded		
gotoFrontDoorLeftCondition	90	No	No	s2-CheckUserDrinking	Action	8	Change Action Possibility ::204: You really should have a drink now (Dutch) to 1		
gotoSofaLeftCondition	90	No	No	s2-CheckUserDrinking	Action	9	Execute sequence 'sadGUI' on ::3: Care-O-Bot 3.6		
gotoChargingStationCondition	90	No	No	s2-CheckUserDrinking	Action	10	Execute sequence 'attractAttention' on ::3: Care-O-Bot 3.6		
gotoCurrentUserLocationCondition	90	No	No						
gotoFridgeCondition	90	No	No						
s2-SurprisedToBasic-Timeout	60	No	No						
s2-HappyToBasic-Timeout	60	No	No						
s2-SadToBasic-Timeout	60	No	Yes						
FridgeAlert	50	Yes	No						
s2-UserHasDrunk	30	No	No						
s2_doorbell-ignore	30	No	No						
s2_doorbell-answer	30	No	No						
s2_gotoChargingArea	30	No	No						
s2_depositCup	30	No	No						
s2_waitAtDoor	30	No	No						
s2_gotoFridge	30	No	No						
s2_collectDrinkAtFridge	30	No	No						
s2-CheckUserDrinking	30	No	No						

Evaluated sequence: s2\_gotoFridge -> result is NOT executable.  
 Evaluated sequence: s2\_collectDrinkAtFridge -> result is NOT executable.  
 Evaluated sequence: s2-CheckUserDrinking -> result is NOT executable.  
 Evaluated sequence: s2\_doorbell -> result is NOT executable.  
 Evaluated sequence: s2\_drinkWarning -> result is NOT executable.  
 Evaluated sequence: sleep -> result is executable.  
 Evaluated sequence: s2-UserHasDrunk -> result is NOT executable.  
 Evaluated sequence: resetDrinkReminder -> result is executable.  
 Evaluated sequence: s2-CheckUserDrinking -> result is NOT executable.

Test Planner    Enable Debug    cu    1    2    3    UhrhUser:localhost

**Figure 3.** The behaviour scheduler. The right side shows the set of behaviours. A green box indicates that the behaviour is available for execution. Individual behaviours can be shown on the left box. The status of each pre-condition is shown and indicated by a green box. An execution status box is shown at the bottom of the screen.

### 3.3.4 Resource Management and Dealing with Conflicts.

There are occasions where resource conflicts can occur. For example, consider a situation where the robot needs to inform the user that the fridge door is open. The robot might say ‘the fridge door is open’ and then proceed to the kitchen. Another behaviour might be for the robot to react to the doorbell by saying ‘the doorbell rang’ and then proceeding to the door. If the fridge door was open and the doorbell rang at around the same time the highest priority behaviour would be executed, to the exclusion of the other behaviour. In this example the fridge door behaviour would “win” and the user would never be informed about the doorbell. To overcome such situations we extend the behavioural pre-condition and execution rules with a *resource manager*. This checks the likely resources that any behaviour requires, as well as the preconditions. Behaviours become eligible for execution only if their pre-conditions are true and there are no resource conflicts with higher priority behaviours. For example, consider the following three behaviours:

BEHAVIOUR ‘FridgeAlert’ PRE-CONDITION: The fridge has been open for 5 minutes ACTIONS: Say The fridge is open Robot proceeds to the fridge RESOURCE: base
BEHAVIOUR ‘Doorbell’ PRE-CONDITION: The doorbell rang ACTIONS: Say The doorbell rang Robot proceeds to the door RESOURCE: base
BEHAVIOUR ‘Doorbell-noMove’ PRE-CONDITION: The doorbell rang ACTION: Say The doorbell rang RESOURCE: none

In the example behaviours above, all three would normally be eligible for execution, and the higher priority “FridgeAlert” would execute. However, using the resource manager results in the “FridgeAlert” locking the “base” resource (this is the robot navigation mechanism). Thus “Doorbell” is no longer valid for execution. However, “Doorbell-noMove” is still valid as it has no resource requirements. The result is that the both the “FridgeAlert” and “Doorbell-noMove”

behaviours will execute in parallel, resulting in the user being informed that the fridge is open and that the doorbell rang and the robot proceeding to the fridge (but not to the door).

### 3.4 Dealing with Time

Interval-based reasoning, often based on Allen's temporal logic (Allen, 1983) (although for an alternative logic also see (Morchen, 2006)), within 'smart homes' has been previously considered by Sciavicco (Sciavicco, 2010) where a logical analysis is undertaken and is used to show that "interval-based relations are particularly adequate to express typical smart homes constraints". Augusto and Nugent (Augusto & Nugent, 2004) provide a language for expressing ECA (event-condition-action) rules (which are conceptually similar to our behavioural rules) that employ temporal relationships and apply these within a smart home environment using an active database. They specifically apply their formalism to emergency type situations and demonstrate that such analyses increase the possibility of complex event detection. Jakkula and Cook (Jakkula & Cook, 2007) also consider how temporal relations could be learned in a smart home environment, and they suggest that learning of such relations would be of much benefit, especially in relation to elderly persons. In our work, we are primarily concerned with robot behaviours within the home, therefore many of the decisions on which behaviours execute will be based on real-time considerations of sensor readings and their current or historical values. Our approach takes a pragmatic view of dealing with temporal relationships and provides a simple mechanism to apply such relationships in a practical way. Our future work also includes research into learning such relationships from occupant activity within the house.

#### 3.4.1 Temporal Relationships

We constructed our sensor tables to reflect both the event driven nature of the system and to allow immediate decisions on sensory durations. We created two tables, the first holding a historical log of all sensor value changes, and a second table holding one row per sensor with the current value and the previous value (if different) of that sensor. Thus each sensor event, for example, a light switch being turned on, creates a time-stamped single row in the sensor log table which then triggers the update of the immediate state/time and previous state/time of that sensor on the sensor table. By constructing the sensor table in this way fairly complex decisions based on the temporal rules can be constructed. Take for example a doorbell sensor. This type of sensor is 'on' only for a short period of time. Although we would normally, when constructing a behaviour using the doorbell, say "If the doorbell rings, then make the robot say 'someone at the door' ". However, due to the short period of sensory activity of the doorbell, we would need a more precise definition, for example, "If the doorbell rang within

the last ‘n’ seconds, then make the robot say ‘someone at the door’ ” . Given such considerations we defined three basic conditions:

1. *is the value of sensor X equal to Y now?*
2. *has the value of sensor X been equal to Y for time period `now - T`?*
3. *was the value of sensor X equal to Z at any time during time period `now - T`?*

where X is a sensor, Y is the current real-time value of sensor X, Z is the previous value of sensor X and T is a time unit (e.g. seconds, minutes, hours, days)

The first condition is simply a real-time reaction to a sensor value, for example, ‘Is the microwave on?’ . The second condition adds a duration to the current sensor state, e.g., ‘Has the microwave been on for 30 minutes?’ . The final condition considers the containment of the condition during some time period, for example, ‘Was the microwave previously on at any time during the last 2 hours?’ . Clock-based decisions can also be added. For example, ‘is the current time greater or equal to C’ and ‘is the current time between C and D’ (where C and D are clock times in hours, minutes and seconds). This allows for further complexity, for example, assume that the house occupant likes to sleep in the afternoon, but wants to be woken by the robot if they sleep longer than 1 hour. The following rule could be used:

*“If the sofa has been occupied for 1 hour, at any time between 1pm and 5pm, make the robot say ‘It’s time to wake up’ ”.*

Using the userGUI this would translate to the following behavioural template (note that the function *spBetweenTimeCheck* is a SQL stored procedure which checks time intervals and returns a single row if the check is true).

```
SELECT * FROM Sensors WHERE sensorId = 15
      AND value = 0 and lastUpdate+INTERVAL 3600 SECOND <= NOW()
CALL spBetweenTimeCheck('13:00:00','17:00:00')
speak,0,it's time to wake up           (makes the robot say 'It's time to wake up')
```

#### Example 4. Checking for timed intervals.

The sensor table, by holding two timed events, can immediately yield duration information useful for real-time actions. The sensor log table, on the other hand, can then be used via our ACCOMPANY Deliverable 3.3 Report

context analysis program, to carry out a deeper analysis of events (this is also discussed by Sciavicco (Sciavicco, 2010) in the context of sleeping). For example, we may want to know if the user is having lunch. This might be expressed as a series of behavioural pre-conditions such as, dining room chair occupied, time is between 12am and 2pm, the fridge has been opened in the last 30 minutes etc. However, if the user vacated the chair temporarily, the ‘having lunch’ context may still be true, but using only real-time sensor information as pre-conditions would yield the opposite result. In this context an analysis of the sensor log, biased with appropriate behavioural parameters (for example, lunch typically takes 30 minutes) would avoid the ‘noise’ inherent in the activity. Furthermore, by considering ‘having lunch’ as simply a new sensor, all of the previous temporal rules can then be applied. Thus we could ask ‘has the occupant had lunch today?’ .

These encoding facilities can therefore cope with a wide range of possibilities and capture information related to both current activity, past activity and used to encourage socially desirable and non-passive activities in the house occupant. This latter function is primarily set through the creation of predicate sensors from contextual rules.

## 3.5 Robot Teaching

One of our goals in this work was to allow co-learning, whereby robot and user work together to achieve a goal. To achieve this we approach the problem in two ways. Firstly, by directly teaching the robot, via a GUI, what has to be achieved and when it should happen and secondly, via the robot (or the house) learning what is happening and learning to adapt itself to those conditions. The learning objective is future work for the final year of the ACCOMPANY project and described in section 7. In the following sections, we describe our current teaching interfaces.

### 3.5.1 Teaching Interfaces

In order to create behaviours the user as a minimum would need to specify what needs to happen (the actions of the robot) and when those actions should take place (setting pre-conditions). We provide two levels of interface (in addition to direct programming of behaviours by robotic experts). The first allows direct entry of behaviours by specifying rules explicitly based on sensor values (see Figure 4. The semi-technical teaching interface), and choosing actions on the robot including the setting of post-conditions via predicate sensors. Hierarchical scaffolding of behaviours is also possible by choosing existing behaviours as actions. We envisage that this



facility would be used by 'semi-technical' persons generating sets of behaviours for the first time.

### 3.5.2 Using the teaching system to create graphical interfaces between the user and the robot

Note that the semi-technical GUI teaching facility also allows behaviours to be created that have direct interaction with the user via an automatic GUI generation facility. The automatic GUI generation can operate in two modes - the first is where the robot prompts the user (via a tablet computer) for a response and then waits for the user actions (we call this the *UH GUI*). This was the GUI demonstrated during the first review meeting for robot control in Accompany scenario 1. The UH-GUI operates by associating behaviours with buttons on a screen presented to the user. The second is to provide 'action possibilities' and changes to an empathic mask (we call this the *SienaGUI*). The SienaGUI is described in ACCOMPANY workpackage 2 and detailed in (Stienstra, Marti, & Tittarelli, 2013; Marti & Stienstra, 2013a, 2013b). The SienaGUI was used for robot and user interaction in Accompany scenario2.

### 3.5.3 Semi-technical Teaching Interface

The semi-technical teaching interface is shown in Figure 4. It operates by associating sensory pre-conditions as rules, with actions for execution.

The sensory 'sets' range from user and robot locations, environmental sensors to abstract sensors and time constraints. When a sensor is selected, the appropriate sensor interpretation is also presented. Thus the hot water tap presents a temperature setting, whereas the light switch only presents on or off. On selecting a sensor and the appropriate value the user has the opportunity to group the rules via AND or OR settings. On the right of the screen the choice is displayed to the user.

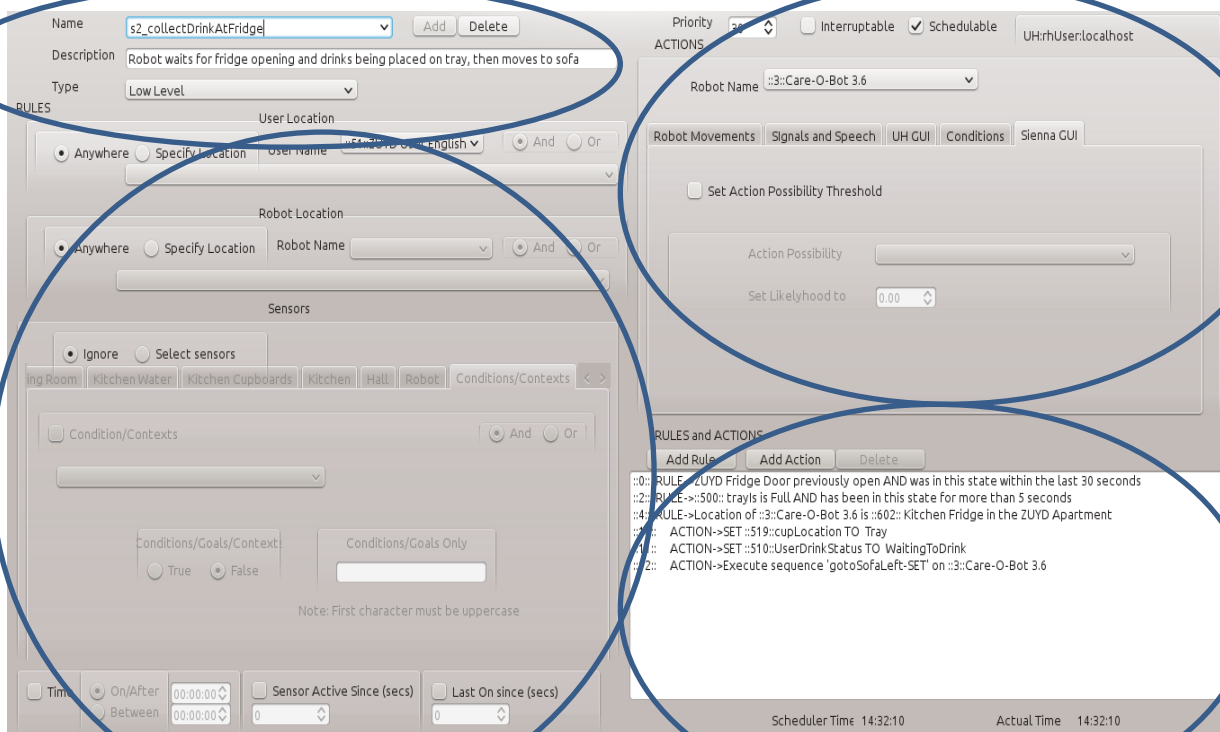
Note that the behavioural rules can apply to individual users, and to individual robots. Also, where necessary, messages can be generated in an appropriate language. This is based on the user preferences.

Robot actions and the setting of post-conditions can be similarly selected and displayed. The final behaviour is shown on the bottom right of the screen and the interpretation is shown in non-technical language.

### 3.5.4 Example of the Semi-Technical Teaching facility

This defines the label and description of the behaviour

This describes the behaviour priority, whether it can be pre-empted, and whether it is scheduled. Unscheduled behaviours can be used as complex primitive operations. Each tab shows a set of robot primitive actions or existing task sequences



The set of rules is defined here, including rules for user and robot location, environmental sensor values, abstract sensor values and temporal constraints. Each tab represents a collection of sensors

The final complete behaviour is shown here. This gets automatically translated into SQL rules, post-conditions and robot operations

Figure 4. The semi-technical teaching interface

### 3.5.5 User Teaching Interface

The second facility (see figure 3) takes away much of the complexity of the former by automatically generating many of the sub-behaviours required to operationalize the system. The cost of this simplification is a loss of generality; however it is compensated for by ease of use.

Taking an example of a user wanting to be reminded to take their medicine at 5pm. In the semi-technical teaching GUI we would need to associate each precondition with the appropriate sensor, including the predicate sensors. To some extent this requires a logical approach akin to creating simple programs or planning domains. In the userGUI, a less complex facility, the user need only specify *what is important* (that the reminder be given at a particular time) and the background system automatically generates the appropriate predicate pre-conditions to ensure the behaviour is generated accordingly.

This is possible as the majority of behaviours envisaged tend to follow a common template, and we exploit this template to generate the appropriate conditional logic. Appendix 8.2 contains a list of typical behaviours that have been set up and tested using the userGUI.

In this manner much of the cognitive load is removed from the user and left to the behaviour generation system. Co-learning is operationalized by allowing the robot to provide details of its existing sets of skills that can then be exploited by the user.

Taking, for example, the behaviour, “Remind me to take my medicine after 5pm”. If this simple behaviour were to be coded in the Semi-technical interface two behaviours would be required:

- 1) The first behaviour would need to check that the time was after 5pm and that the user had not been already reminded. If these conditions are true then the robot carries out its normal procedure of setting appropriate LED colours, saying “It’s time for medicine” and setting an abstract sensor to indicate that the user has been reminded.
- 2) At some point later, in this example after midnight, the ‘user reminded’ sensor will need to be reset so that it can fire the next day.

Thus two behaviours need to be created, and careful alignment of reminder rules need to be inserted.

A picture of the behaviours is show in Figure 5.

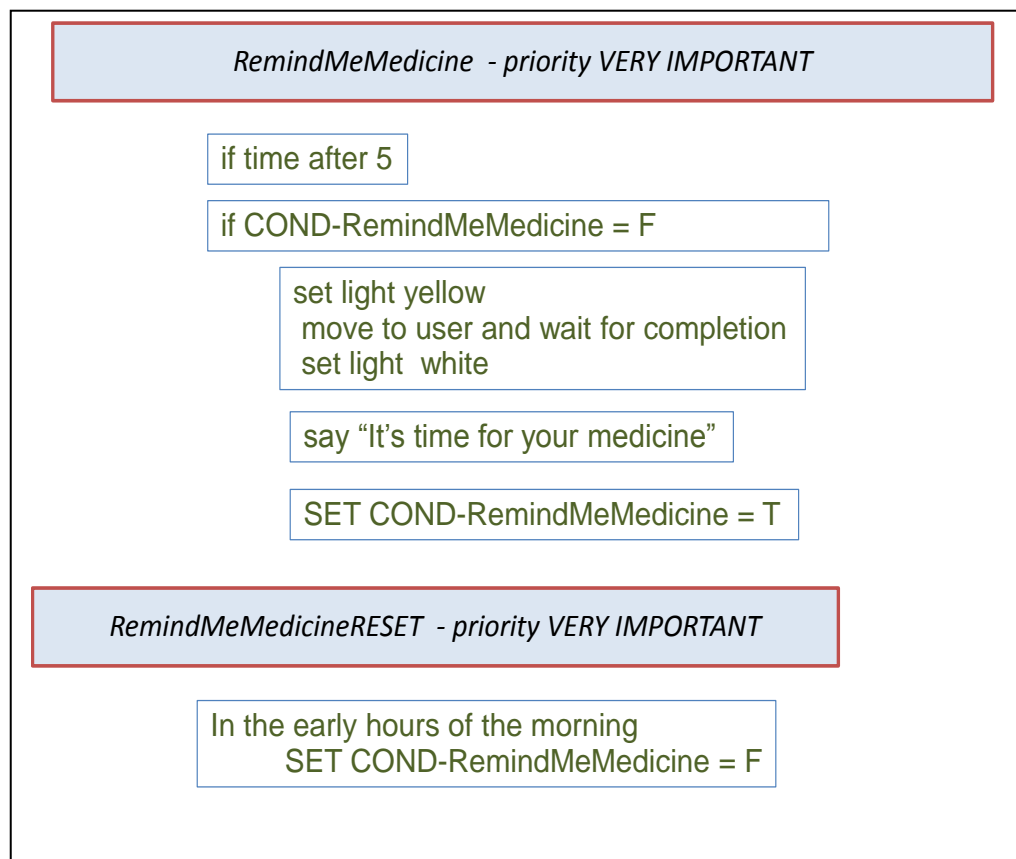


Figure 5. Setup of a simple behaviour

However, for a user teaching interface this kind of behavioural logic is difficult. Instead, behaviours can be created by following a fixed template. Thus the requirements of the user would be straightforward – WHAT to do – remind me to take medicine, WHEN to do it – at 5pm.

The underlying conditional logic, the light displays associated with the robot actions, and the reset of the reminder conditions can all be automatically generated. A typical template used by the userGUI (in this case for reminders) is as follows:

Firstly, take information from the userGUI to create various abstract sensors.

```
Create the following abstract sensors::  ReminderTime = t      (e.g. 5pm)
                                         Cond-Reminder = TRUE
                                         Cond-Remind-again = FALSE
                                         textItem  e.g. "Have you taken your medicine?"
                                         repeatAfter = n  (e.g. 60 seconds)
```

Then create behaviours based on the following template:

Create 3 behaviours:

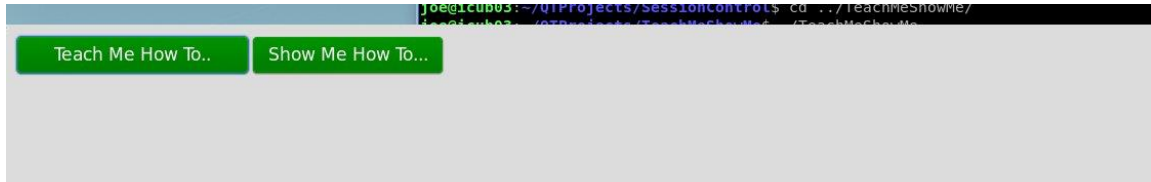
- 1) **ReminderX-reset:**      If NOW between midnight and t  
                                    and  
                                    Cond-Reminder = FALSE  
  set Cond-Reminder = TRUE  
  set Cond-Remind-again = FALSE
  
- 2) **ReminderX:**                If NOW >= t  
                                    and  
                                    Cond-Reminder = TRUE  
  say "text item"  
  set Cond-Reminder = FALSE  
  set Cond-Remind-again = TRUE
  
- 3) **Remind-Again:**            If Cond-Remind-again = TRUE  
                                    and  
                                    Cond-Remind-again set within repeatAfter sec  
  say "textitem"  
  set Cond-Remind-again = FALSE  
  (reset Cond-Remind-again for repeat)

Where other types of robot actions are required (in addition to the 'say' action above), these are added as necessary. Similarly additional pre-conditions can also be included.

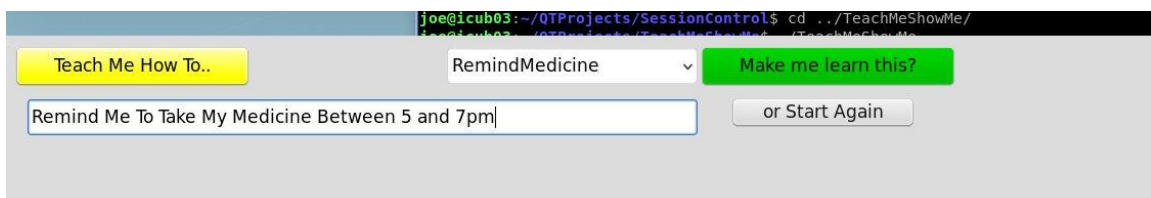
Where diary items are not required, a simple generation of pre-conditions and actions can be generated.

The following section shows how the user enters information into the userGUI to create behaviours.

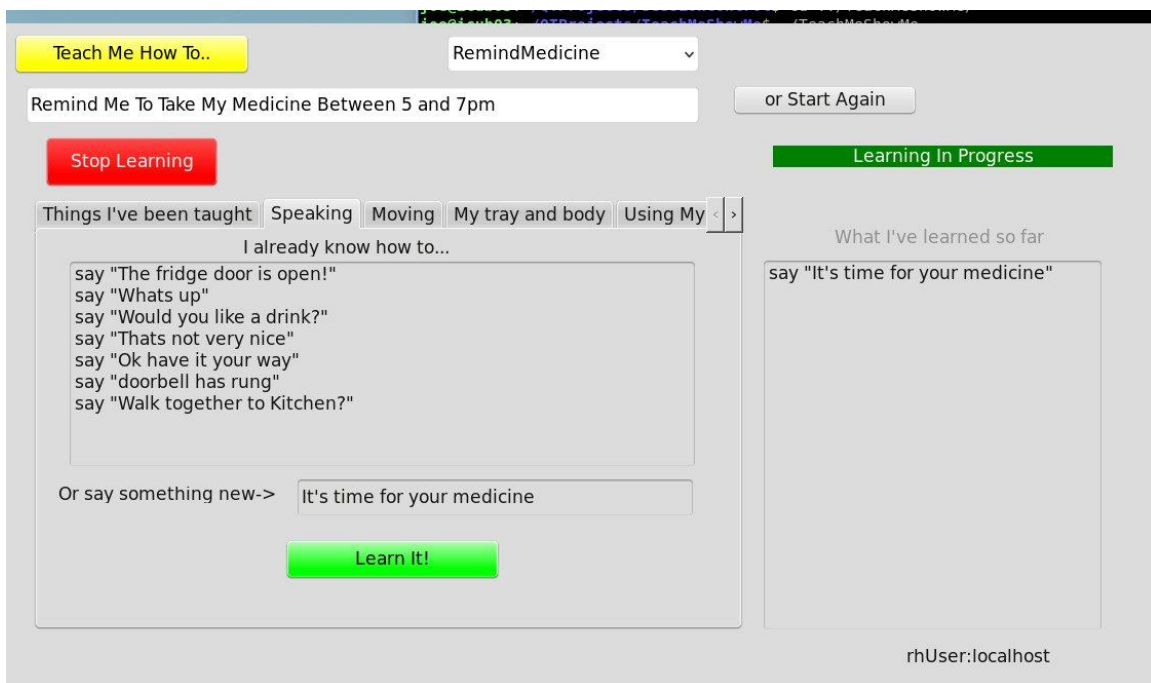
### 3.5.6 Example of the User Teaching facility



User is presented with the “TeachMe-Show Me” screen and presses “Teach me”.

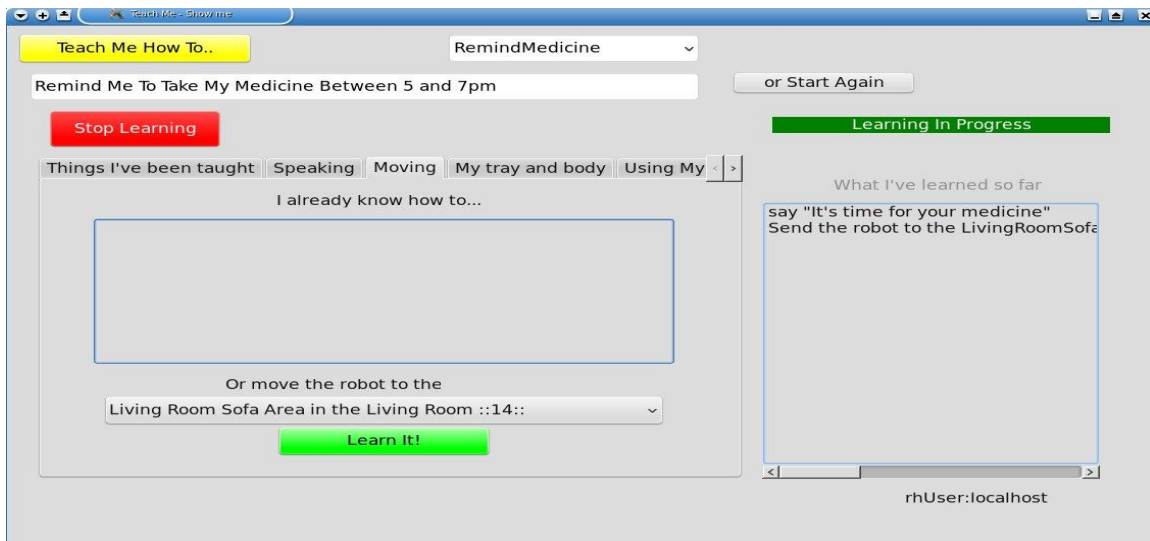


The user enters a text description of their requirement and clicks the “make me learn this” button.

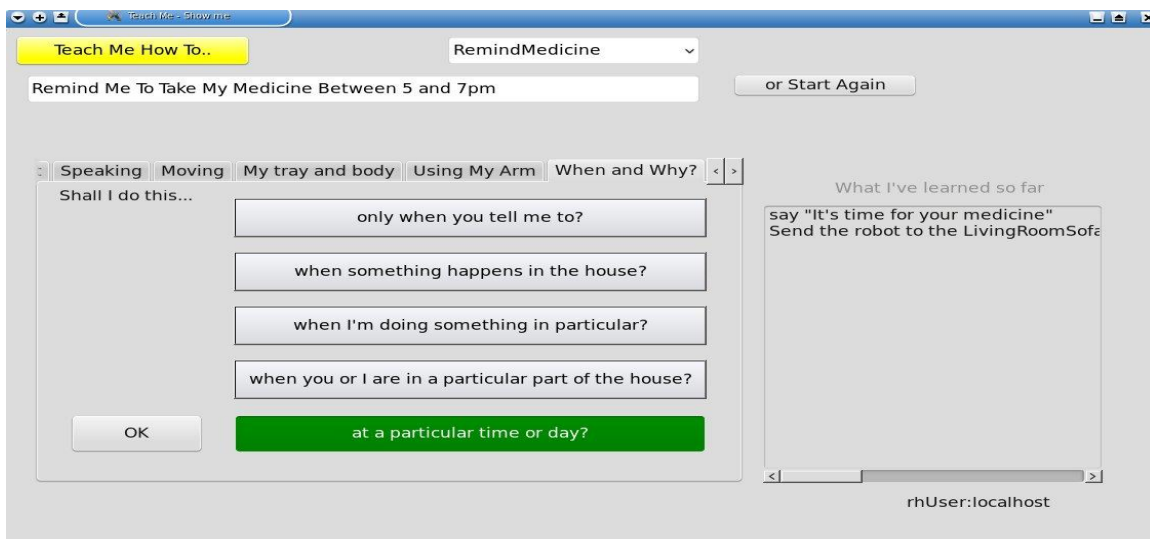


The “WHAT to do” screen appears. The tabs show the set of actions the user can teach the robot, and also shows what the *robot already knows* (the “I already know how to...” box).

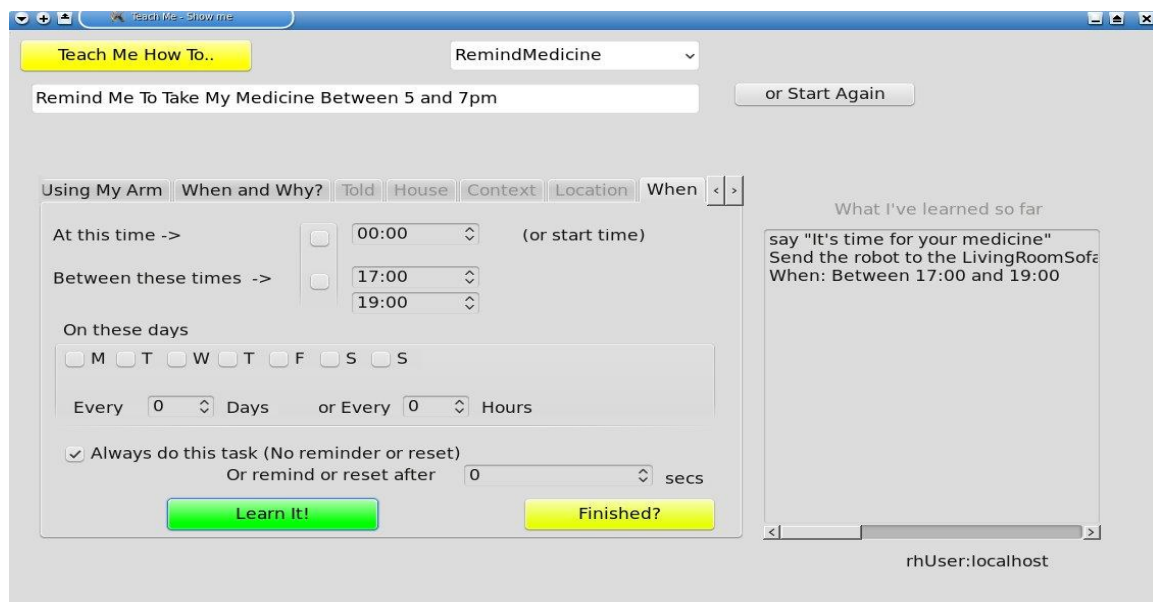
Clicking the “learn it” button causes the action to be saved into the “behaviour box” on the right.



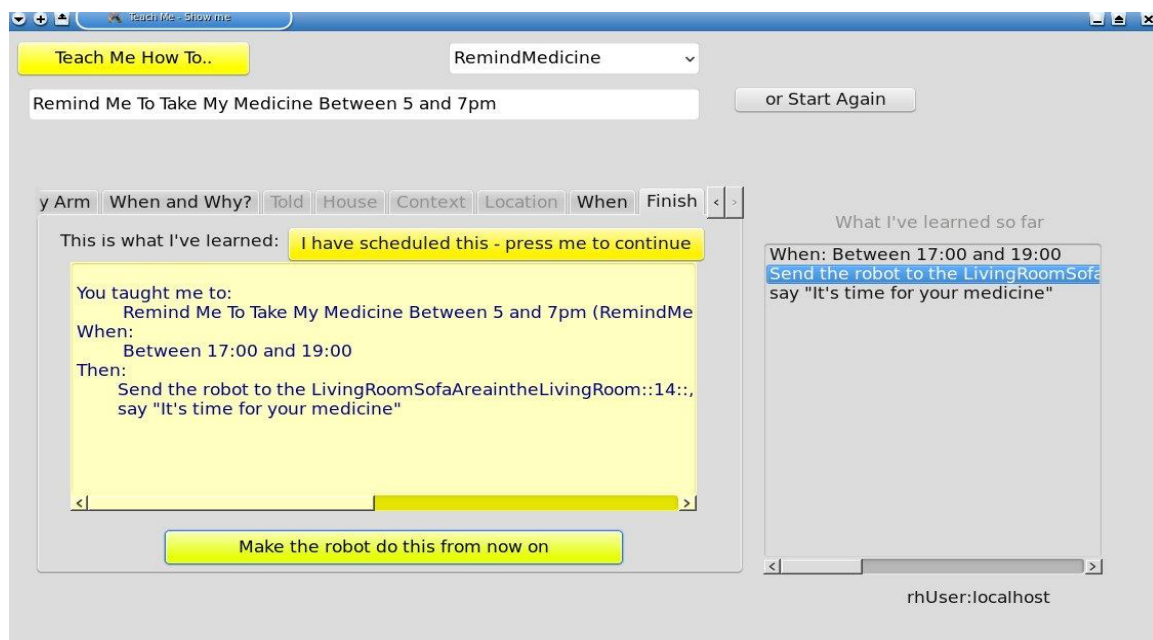
The user clicks the “Moving” tab. The robot has not been previously taught this and so the “I already know how to do this...” box is empty. The user chooses to select a known positions from the drop down box at the bottom. This gets added to the behaviour screen on the right.



The “WHEN To Do IT” screen appears, and the “particular time of day” option is chosen. Note that multiple options can be chosen at this point.



Because the user chose the WHEN option as a particular time of day, a diary function appears. The appropriate times are entered. The user then clicks the "finished" button.



The complete behaviour is displayed to the user. They have the opportunity to go back and change it. Once finished the behaviour is sent to the scheduler.



## 4 Capabilities and Assessment

Langley et al., (Langley et al., 2009) provide a general set of capabilities and assessments for cognitive architectures and one of our associated aims was to assess how well the memory architecture (the CMA) described in this document meets these capabilities and assessments. A summary of how the CMA meets the capability criteria is shown below in Table 2 and a summary of how the CMA meets the assessment criteria is shown in Table 3.

### 4.1 Capability Assessment

The table below compares Langley, Laird and Rogers criteria for Cognitive Architectures against the facilities provided by the CMA.

**Table 2. Cognitive Systems Capabilities and the CMA**

Cognitive Architectures: Capabilities	Cognitive Architectures: Requirements	How does CMA support this?
<b>Recognition and Categorization</b>	Recognises events and familiar patterns both static and dynamic. Categorisation and recognition are linked to perception and can operate on abstract mental structures.	<i>Recognition via behavioural pre-conditions. Dynamic time based events recognised. Abstract mental structures supported by abstract sensors. Teaching facilities allow encoding for recognition of new events.</i>
<b>Decision making and Choice</b>	Decision making, should be able to select from alternatives and support conflict resolution	<i>Decision making supported by behaviours, pre-emptive priority-based scheduling and resource management.</i>
<b>Perception and Situation Assessment</b>	Perception should sense and interpret situations.	<i>Perceptions via real-time updates from environmental and robot sensors. Interpretation by context-analysis.</i>
<b>Prediction and Monitoring</b>	Learning of predictive models and responding to	<i>Predictive modelling not yet</i>

	changes from monitoring these models.	<i>supported by CMA.</i>
<b>Problem Solving and Planning</b>	Support decision making, execution monitoring and re-planning	<i>Decision making guided by behavioural execution but also on lower level planning via HTN domains.</i>  <i>Execution monitoring operates at both levels.</i>
<b>Reasoning and Belief Maintenance</b>	Relationships between beliefs and inferring new beliefs.	<i>Reasoning and inference supported by context analysis and setting of context and predicate sensor values based on environmental and existing context and abstract sensors.</i>
<b>Execution and Action</b>	Motor skills should be encoded, action pre-plans available, reactive and deliberative skills split and the ability to learn new skills available.	<i>Procedural memory supports primitive actions and hierarchically encoded actions.</i>  <i>Planning domains that generate actions are pre-coded.</i>  <i>New skills can be taught (or in future work learnt).</i>
<b>Interaction and Communication</b>	Dialogues must be supported between agents. Transformation into appropriate communication medium. Use of natural language in communication and learning of changes to communication mechanisms.	<i>Dialogues supported both for sequential direct action and for parallel emphatic dialogue between robot and (human) via GUI based interfaces.</i>  <i>Natural language is not supported.</i>
<b>Remembering, Reflecting</b>	Remembering behavioural	<i>Episodic memory facilities</i>

<b>and Learning</b>	episodes, explaining and learning from such episodes.	<i>provide retention of episodes. Explanation and learning not currently supported.</i>
---------------------	---	---

## 4.2 Evaluation Criteria

Langley et al., (Langley et al., 2009) describe a set of criteria that could be used to evaluate cognitive systems. In the Accompany project, our research is on providing robot services to elderly persons within a sensorised home and the focus is on the use of a cognitive architecture to provide support for this research, rather than development of a cognitive architecture as an end in itself. However, key issues, such as scalability and efficiency, are still of prime importance. The following table provides an indication of how well CMA meets the evaluation criteria.

**Table 3. Cognitive Systems Evaluation and the CMA**

Cognitive Architectures: Evaluation category	Cognitive Architectures: Evaluation Detail	How does CMA compare?
<b>Generality</b>	Can the architecture operate and support requirements and tasks in a wide variety of diverse environments?  Versatility - how difficult/how much effort to construct new tasks?	<i>CMA operates in a specific environment (within a sensorised house and using a service robot). In this sense it is not designed as a general purpose system. However, our approach could be generalised by excluding the physical sensors and robot. i.e. simply use the abstract sensors to symbolically hold concepts .  Construction of new tasks is relatively easy and tasks can be constructed with minimum effort.</i>
<b>Rationality</b>	What percentage of time do behaviours satisfies execution criteria?	<i>From our experimental scenarios there have been no instances where the behaviour set does not satisfy</i>

	Are there formal measures of behavioural rationality?	<p><i>the criteria of the scenarios.</i></p> <p><i>Although not part of the Accompany research, formal measures of rationality of behaviours within the CMA are currently being carried out in another project (Trustworthy Robot Assistants project, EPSRC, UK funded).</i></p>
<b>Optimality</b>	Does the architecture support best value functions, optimality and bound rationality.	<p><i>Currently CMA does not have optimality features.</i></p>
<b>Efficiency and Scalability</b>	<p>Are there time and space constraints?</p> <p>Is real-time scalability influenced by task difficulty and complexity?</p> <p>Has the architecture been measured over a range of problems?</p> <p>Is efficiency inversely related to scalability</p> <p>Can the architecture learn over time and does this effect scalability (e.g. more options - more choice - slower operation ).</p>	<p><i>There are no known physical time or space constraints in the current architectures.</i></p> <p><i>Scalability is currently evaluated via our behavioural scenarios.</i></p> <p><i>Both of the scenarios developed in Accompany Year 1 and year 2 have suffered no scalability problems even when behavioural complexity has increased.</i></p> <p><i>Further testing of the architecture in year 3 may yield more information on this topic.</i></p> <p><i>The current CMA does not automatically learn over time; however this is a research issue for year 3.</i></p>
<b>Reactivity and Persistence</b>	<p>Is there adequate speed of response on given recognise-act cycle.</p> <p>Does the architecture suffer</p>	<p><i>Currently there is no 'speed of response' issues on the recognise-act cycle.</i></p> <p><i>The architecture attempts to</i></p>

	<p>from the frame problem?</p> <p>Is there persistence of goals? Does tight reactivity lose sight of longer term goals?</p>	<p><i>ground its perceptions in environmental reality. The use of a complex world model is avoided. However the robot operates in a relatively closed world and frame problem issues are unavoidable.</i></p> <p><i>There is a clear separation between reactivity and deliberation. Reactive processes are generally limited to navigation which encompasses obstacle avoidance. The use of the resource manager generally avoids loss of goals persistence; however the current experimental scenarios are not complex enough to fully test this for longer term goals.</i></p>
<b>Improvability</b>	<p>Additions by the programmer or user?</p> <p>Are additions possible?</p> <p>Are there any metrics for performance gain vs. programmer/end user effort?</p>	<p><i>Additions and creation of behaviours can be carried out at all levels of expertise (from programmer to end-user), however there is a general loss of functionality and expressiveness due to the compromise with ease-of-use.</i></p> <p><i>There are no automatic metrics for performance gain vs development effort. However, a formative evaluation of the user teaching interface is due in year 3 of the Accompany project.</i></p>
<b>Autonomy and extended operation</b>	<p>Can the system avoid failing when encountering the</p>	<p><i>The behavioural scheduling system can cope with limited</i></p>

	<p>unexpected?</p> <p>Does the system have the ability to ask for help?</p>	<p><i>failure of non-critical components (for example the robot torso not moving) however critical component failure (such as navigation) will cause the CMA to report errors and stop scheduling behaviours.</i></p> <p><i>The CMA can ask for help via behaviours created for this purpose, and in general will report and question technical errors. However, there is no general purpose facility for 'asking for help'.</i></p>
--	---	--

## 5 Usage of the CMA in Experimental Scenarios

Scenarios have been prepared and behaviours created which are based on realistic scenarios derived from user requirements (Lehmann et al., 2013). In the summer of 2013 the CMA outlined above was employed and technically evaluated using scenarios derived from Work Package 1. This involved 11 elderly persons interacting with the robot in a fetch-and-carry task. The robot was also instructed to check that the person has been drinking adequately over the period (to avoid de-hydration). This scenario employs the SienaGUI (part of work package 2), and fully integrates person location tracking (part of work package 4), arm kinematics, and a cup fluid level sensor designed at the University of Hertfordshire. In total around 50 behaviours were created using the semi-technical interface described in section 3.5.3.

Previously, scenario 1 (demonstrated in real-time at the ACCOMPANY Review in December 2012) the user was reminded to take medicine at a particular time (medical re-ablement), asked to accompany the robot to the kitchen (a form of physical re-ablement), gave a reminder about medicine 10 minutes after the first reminder (warning reminder), warned that the fridge had been left open (safety warning), and suggested to the user that they watched TV together (social partner). In total around 30 behavioural components were created using the approach outlined above.

In both of the scenarios described the behaviours, scheduling, execution and planning components all operated successfully.

## 5.1 Other Users of the CMA

Usage of the 'non-technical' interface was used to create an interaction with the Care-O-bot by two 'conceptual' artists residing in the University of Hertfordshire robot house. This was activated during an artistic event held at the robot house in May 2013 (Lehmann at al., 2013). This operated successfully and in real-time in a house of around 20 people.

## 6 Future Evaluation of the Teaching Component

The teaching component described above (the userGUI) is due to be formatively evaluated during year 3 of the Accompany project.

The evaluation procedure will attempt to employ all aspects of the CMA including using the episodic memory visualisation component (an extended version of what was presented in D3.2), the teaching component and the execution scheduler.

The experimental scenario is as follows:

- Participants (max. 20) will be recruited from University staff and students.
- Each experiment will involve 1 participant.
- One of the set of behaviours shown in Appendix 8.2 is chosen for to test the teaching interface. For example, “Whenever I am in the kitchen, come to the kitchen and wait outside with tray raised”.
- This will be used to train the participant in the use of the userGUI teaching system.
- Once taught the participant will be shown how to test the behavior and additionally how to use the episodic viewer to review the behavioral actions of the robot.

- The participant will then be asked to create a similar, but more complex behavior, test it, modify if necessary, and use the episodic viewer. For example, “If I am sitting on the sofa, and the doorbell rings, come to me and say ‘there is someone at the door’”.
- Metrics of time taken to set-up the behavior, the number of test-verify cycles will be recorded.
- Final post-experimental questionnaires will be prepared asking the participant about the general usability of the system and their experience interacting with it, their views on the general use of the teaching facility, the episodic viewer and whether such tools would be useful to them in using the robot. A short interview will complete the experiment.

A post-experimental quantitative analysis of the metrics of the teaching episodes and a qualitative evaluation of the questionnaires and interviews will be prepared and results documented in a conference or journal paper.

This formative evaluation is due to take place in January/February 2014 at the Robot House at the University of Hertfordshire.

## 7 Future Developments to Support learning

In the sections above we described the *teaching* component of *co-learning*. In this section we will outline the *learning* part. We approach this problem by considering two aspects: firstly, how can the robot learn through interaction with a human partner, and secondly, how can the robot learn through its experiences in the house. Both of these areas were described and supported with a literature review presented in deliverable D3.1.

In dealing with the first issue, learning from interaction, we are planning a series of experiments with a new implementation of the Interaction History Architecture (IHA) (Mirza et al., 2007). IHA was originally developed at the University of Hertfordshire by Mirza et al., in 2007, and later extended with a short term memory function by Broz et al. (2009).

IHA is a mechanism where grounded sensorimotor histories are used to learn behavior sequences. When the robot executes its behavior, it creates a memory of past “experiences” based on its sensors, encoders, and internal variables over a short temporal window. Each



experience is associated with the behavior the robot was carrying out and a reward value based on properties of the experience. These experiences are recorded in memory and recalled using a metric of information distance which is used to select the most similar experience compared to its current state. New actions are selected probabilistically based on the reward value.

This existing memory architecture was previously deployed on humanoid robots but has never been deployed or tested with a 'service' robot such as the Care-O-bot, and it has never been used in assistive technology scenarios. Proof of concept experiments are planned in the third year of the Accompany project. These will be based on e.g. allowing the Care-O-bot to learn to follow a person at a specific distance and if within this distance to look at the person. The robot will learn these overall behaviors (following, stopping at safe distances and looking up) entirely through interaction with the user and not via any explicit teaching mechanism.

The second issue, where the robot learns through its experiences in the house, will be based on work by Saunders et al., (2007) which was previously outlined in Deliverable 3.2. A series of experiments will be designed where the typical daily routines of the house occupant are modeled (such as having lunch). In this example (lunch) the user might request the robot to join them and stand next to the lunch table with its tray raised (via for example the 'squeeze me interface – see workpackage 2) . The robot would learn from these experiences to then do this automatically by extracting the appropriate temporal rules and pre-conditions from its sensory experiences. This aspect of learning is due to be investigated in year 3 of the project.

## References

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11), 832 – 843.

Arkin, R. (1998). Behavior-based robotics. Mit Press.

Augusto, J. C., & Nugent, C. D. (2004). The use of temporal reasoning and management of complex events in smart homes. In *Proc. of the 16th european conference on artificial intelligence (ECAI 2004)* (pp. 778 – 782).

F. Broz , H. Kose-Bagci , C. L. Nehaniv and K. Dautenhahn "Learning behavior for a social interaction game with a childlike humanoid robot", *Proc. Social Learn. Interact. Scenarios Workshop, Humanoids*, 2009

Duque, I., Dautenhahn, K., Koay, K. L., Willcock, I., & Christianson, B. (2013). Knowledge-driven user activity recognition for a smart house, development and validation of a generic and low-cost, resource-efficient system. In *ACHI 2013, the Sixth International Conference on Advances in Computer-Human Interactions* (pp. 141 – 146).

Eurostats. (2013). Population projections. Online database.  
<http://epp.eurostat.ec.europa.eu/statisticsexplained>.

Firby, R. J. (1987). An investigation into reactive planning in complex domains. In *Proc. AAAI-87*. AAAI.

Freed, M. (1998). Managing multiple tasks in complex dynamic environments. In *Proc. 15th International Conf. on Artificial Intelligence* (pp. 921 – 927). AAAI Press.

Gat, E. (1998). On three-layer architectures. In *Artificial intelligence and mobile robots*. MIT Press.

Hartanto, R. (2011). A hybrid deliberative layer for robotic agents: Fusing DL reasoning with HTN planning in autonomous robots. Springer Berlin Heidelberg.

Hu, N., Englebienne, G., & Krose, B. J. A. (2012). Bayesian fusion of ceiling mounted camera and laser range finder on a mobile robot for people detection and localization. In *Proceedings of IROS workshop: Human Behavior Understanding* vol. 7559. Lecture notes in computer science (pp. 41-51).

Jakkula, V., & Cook, D. J. (2007). Learning temporal relations in smart home data. In *Proc. 2nd Int. Conf. on Technology and Ageing*, Canada.

Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2), 141-160.

Lehmann, H., Syrdal, D., Dautenhahn, K., Gelderblom, G., Bedaf, S., & Amirabdollahian, F. (2013). What can a robot do for you? - evaluating the needs of the elderly in the UK. In *Proceedings of the 6<sup>th</sup> International Conference on Advances in Computer-Human Interactions*. Nice, France.

Lehmann, H., Walters, M. L., Dumitriu, A., May, A., Koay, K. L., Saez, J., Syrdal, D. S., Wood, L., Saunders, J., Burke, N., Duque, I., Christianson, B. & Dautenhahn, K. (2013), Artists as HRI Pioneers: A Creative Approach to Developing Novel Interactions for Living with Robots, in *Proc. Int. Conf. Social Robotics, (ICSR13)*.

Martens, C., Prenzel, O., & Groser, A. (2007). The rehabilitation robots FRIEND-I II: Daily life independency through semi-autonomous task-execution. In *Rehabilitation robotics (pp. 137 – 162)*. I-Tech Education and Publishing (Vienna, Austria).

Marti, P., & Stienstra, J. (2013a). Engaging through her eyes: embodying the perspective of a robot companion. In *Proceedings of the 18th International Symposium on Artificial Life and Robotics, AROB 2013.*, Daejeon, Korea.

Marti, P., & Stienstra, J. (2013b). Exploring empathy in interaction: Scenarios of respectful robotics. *Journal of Gerontopsychology and Geriatric Psychiatry*, 26(2), 101 – 112.

Mirza, N. A.; Nehaniv, C. L.; Dautenhahn, K. & te Boekhorst, R. Grounded Sensorimotor Interaction Histories in an Information Theoretic Metric Space for Robot Ontogeny *Adaptive Behaviour, SAGE Publications, 2007, 15, 167-187*

Morchen, F. (2006). A better tool than Allen's relations for expressing temporal knowledge in interval data. In *Proc. the 12th ACM SIGKDD int. conf. on knowledge discovery and data mining*, Philadelphia, PA, USA.

Nau, D., Cao, Y., Lotem, A., & Muñoz-Avila, H. (1999). SHOP: Simple Hierarchical Ordered Planner. In *Proc. IJCAI-99* (p. 968-973).

Nilsson, N. J. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 158.

Nilsson, N. J. (2001). Teleo-reactive programs and the triple-tower architecture. *Electronic Transactions on Artificial Intelligence*, 5, 99 – 110.

Off, D., & Zhang, J. (2011). Multimodal integration processes in plan-based service robot control. *Tsinghua Science and Technology*, 16(1), 1-6.

Przywara, B. (2010). Projecting future health care expenditure at European level: drivers, methodology and main results. In European economy. European Commission, Economic and Financial Affairs.

RACE. (2011). EU integrated Project RACE Robustness by Autonomous Competence Enhancement. 24 <http://www.project-race.eu/>.

Reiser, U., Connette, C., Fischer, J., Kubacki, J., Bubeck, A., Weisshardt, F., et al. (2009). Care-obot<sup>®</sup> creating a product vision for service robot applications by integrating design and technology. In *Intelligent Robots and Systems*, 2009. IROS. IEEE/RSJ International Conference (pp. 1992 – 1998).

Saunders, J. Nehaniv, C.L., Dautenhahn, K. and Alissandrakis, A.: 2007, Self-imitation and Environmental Scaffolding for Robot Teaching, *International Journal of Advanced Robotics Systems*, 4:1, 109-124, ISSN 1729-8806

Saunders, J., Burke, N., Koay, K. L., & Dautenhahn, K. (2013). A user friendly robot architecture for re-ablement and co-learning in a sensorised homes. In *Proc. 12th European Conf. Advancement Assistive Technology in Europe*, (AAATE13).

Sciavicco, G. (2010). Using interval-based reasoning in smart homes. In *Proc. of the 4th international*

*conference on ubiquitous computing and ambient intelligence* (pp. 307 – 314).

Simmons, R., & Apfelbaum, D. (1998). A task description language for robot control. In *Proceedings of the conference on intelligent robots and systems (IROS)*.

Stienstra, J., Marti, P., & Tittarelli, M. (2013). Exploring dynamic expression in human-system interaction. In *Proceedings of CHI 2013*, Paris.

Welsh Social Services Improvement Agency. (2006). Demonstrating improvement through reablement. <http://www.ssiacymru.org.uk/reablement>, Last referenced 23rd November 2012.

Weser, M., Off, D., & Zhang, J. (2010). HTN Robot planning in partially observable dynamic environments. In *Proc. Int. Conf. Robotics and Automation (ICRA)*. Anchorage, Alaska, USA: IEEE.

Willow Garage. (n.d.). <http://www.ros.org/wiki/>, Last referenced 30th May 2013.

## 8 Appendix

### 8.1 Complete set of Sensors

The sensor table from the ACCOMPANY database is shown below. Three columns are shown including “sensorRule” , which indicates how the sensor should be interpreted.

sensorId	name	sensorRule
1	Water Pipe Sink Hot	Moving Average
2	Water Pipe Sink Cold	Moving Average
3	Ceiling cupboard door left	Boolean
	Ceiling cupboard door	
4	middle	Boolean
5	Ceiling cupboard door right	Boolean
	Floor cupboard drawer	
6	middle	Boolean
7	Floor cupboard drawer right	Boolean
8	Floor cupboard door middle	Boolean
9	Floor cupboard door right	Boolean
10	Floor cupboard door left	Boolean
11	Water Pipe Washbasin Hot	Moving Average
12	Water Pipe Washbasin Cold	Moving Average
13	Bathroom door	Boolean
14	Toilet flush	Boolean
15	Sofa Seatplace 0	Boolean
16	Sofa Seatplace 1	Boolean
17	Sofa Seatplace 2	Boolean
18	Sofa Seatplace 3	Boolean
19	Sofa Seatplace 4	Boolean
20	Chair Seatplace 0	Boolean
21	Chair Seatplace 1	Boolean
22	Table Pressure 2	Boolean
23	Living room door	Boolean
24	Big cupboard drawer bottom	Boolean
25	Big cupboard drawer top	Boolean
26	Small cupboard door left	Boolean
27	Small cupboard door right	Boolean
28	Small cupboard drawer	Boolean

	bottom	
	Small cupboard drawer	
29	middle	Boolean
30	Small cupboard drawer top	Boolean
31	Desk drawer bottom	Boolean
32	Desk drawer middle	Boolean
33	Desk drawer top	Boolean
34	Desk door	Boolean
35	Office chair	Boolean
36	Bedroom door	Boolean
37	Bed contact	Boolean
38	Wardrobe door left	Boolean
39	Wardrobe door middle	Boolean
40	Wardrobe door right	Boolean
41	Exterior lights	Watts > 10
42	Upstairs lights	Watts > 10
43	Downstairs lights	Watts > 10
44	Cooker	Watts > 10
45	Garage	Watts > 5
46	Sockets	Watts > 5
47	Sockets exterior and garden	Watts > 5
48	Mains supply	Watts > 10
49	TV	Watts > 10
		(Watts > 10 && Watts < 50)    (Watts >
50	Fridge	105)
51	Kettle	Watts > 10
52	Computer Bedroom 1	Watts > 10
53	Table lamp	Watts > 10
54	Microwave	Watts > 10
55	Dishwasher	Watts > 10
56	Toaster	Watts > 10
57	Computer Dining Area	Watts > 5
58	Coffee Machine	Watts > 10
59	Doorbell	Watts > 1
100	Colour Camera	N/A
300	ZUYD Cup	Level
301	ZUYD Sofa Seat 1	Boolean
302	ZUYD Doorbell	Boolean
303	ZUYD Fridge	Boolean

304	ZUYD Sofa Seat 2	Boolean
305	ZUYD Small Sofa	Boolean
306	ZUYD Television	Watts > 10
307	ZUYD Cup 1 Switch	Boolean
500	trays	Predicate
501	trayStatus	Predicate
502	Medicine5PM-Status	Predicate
503	goto	Predicate
504	userNeedsDrink	Predicate
505	UserToldToDrink	Predicate
506	fridgeAlert	Predicate
507	dummyGUIpredicate	Predicate
508	robotLocation	Predicate
509	eyePosition	Predicate
510	UserDrinkStatus	Predicate
511	userShouldDrink	Predicate
512	AP_gotoDoor	Predicate
513	AP_gotoFridge	Predicate
514	AP_ignoreDoorbell	Predicate
515	doorBellReminder	Predicate
516	currentGUIexpression	Predicate
517	AP_gotoFridge	Predicate
518	AP_gotoChargingArea	Predicate
519	cupLocation	Predicate
600	test:lightswitch	Boolean
703	answerdoorbell	Predicate
719	checkFridge	Predicate
720	RemindMedicine	Predicate
901	Sitting_Living_room	Boolean
1001	Lock Siena GUI	Boolean

## 8.2 Typical Behaviours that have been tested with the UserGUI

The table below shows a list of possible behaviours that have been created using the userGUI described in the text (section 3.5.5). This set of behaviours was created in order to demonstrate how the user teaching facility can be used to support requirements for companionship, re-ablement and, medical and safety requirements.

Robot Behaviour	Achieving
Wake me up in the afternoon if I sleep for longer than 1 hour	<i>Re-ablement - avoid sleeping too much during the day</i>
Remind me to ring my daughter if I have not rung her for 2 days	<i>Re-ablement - avoid social isolation</i>
Come and tell me to take 2 aspirin every 2 days, at 4 hourly intervals starting at 10am for 1 month	<i>Assistance – medical - medicine</i>
Let me know that the oven is on (after 1 hour) and keep reminding me (very 1/2 hour) until I have turned it off	<i>Assistance - safety</i>
Come and tell me if someone has rung the doorbell	<i>Assistance – medical - hearing impairment</i>
Whenever I am in the kitchen, come to the kitchen and wait outside with tray raised	<i>Assistance - physical</i>
Remind me to always call my friend Albert on Thursday at 2pm	<i>Re-ablement - avoid social isolation</i>
If the bath taps have been running for more than 5 minutes, tell me and keep reminding me every minute until the taps are turned off	<i>Assistance - safety</i>
If the bath taps have been running for more than 20 minutes text my daughter and repeat every 10m until the taps are turned off	<i>Assistance – safety – raised priority</i>
Occasionally suggest a game of chess together in the early evening	<i>Robot as Companion</i>
Remind me every day from 3 days before that it's my daughter's birthday on the 12 June	<i>Assistance – medical - memory</i>
Join me at the table if I am sitting there and wait with your tray	<i>Assistance - physical</i>



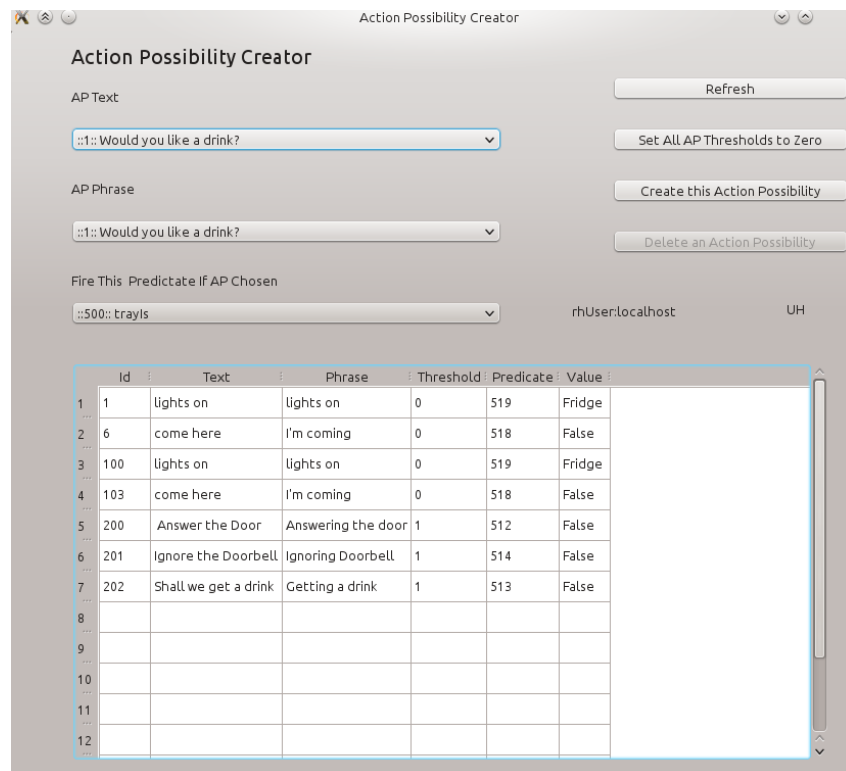
<b>On Mondays, Wednesdays and Fridays come and remind me that the dinner lady is coming at 12:30pm</b>	<i>Assistance – medical - memory</i>
<b>If the weather is good in the afternoon suggest I take a walk in the garden</b>	<i>Re-ablement - physical</i>

### 8.3 Ancillary programs associated with Behavioural setup and Execution

The following is a list of programs that were also developed to support the CMA during the creation of the experimental scenarios and are listed here for completeness.

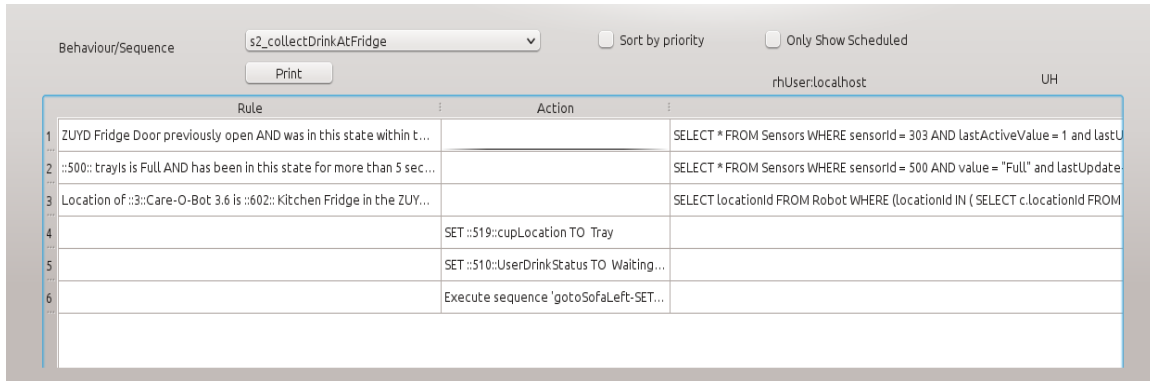
#### 8.3.1 Action Possibility Creator

This program allows action possibilities to be associated with appropriate language local messages and to fire abstract predicate sensors for use in behavioural generation.



### 8.3.2 Behaviour Viewer

*Allows behaviour pre-conditions, actions and post-conditions to be easily viewed and printed. This is useful for behaviour checking and setup.*

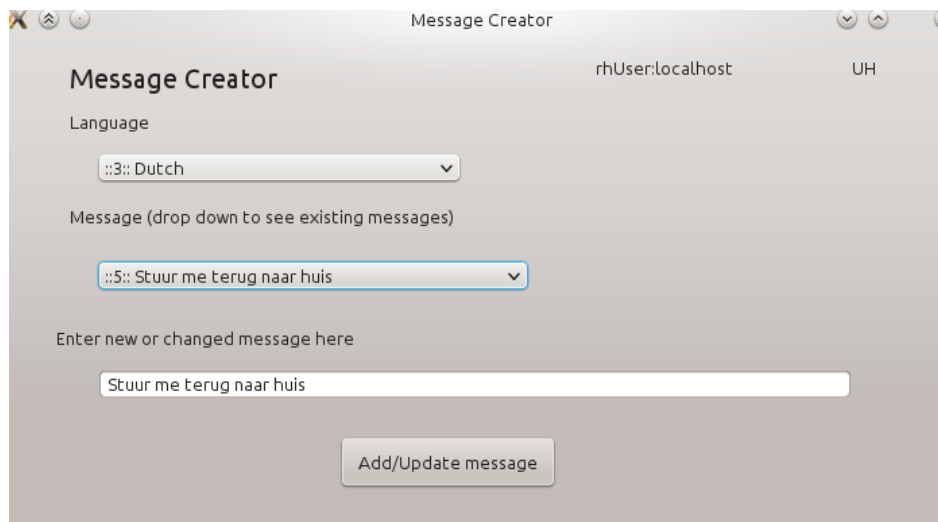


The screenshot shows the Behaviour Viewer interface. At the top, there is a dropdown menu for 'Behaviour/Sequence' set to 's2\_collectDrinkAtFridge'. To its right are two checkboxes: 'Sort by priority' (unchecked) and 'Only Show Scheduled' (checked). Below these is a 'Print' button. The main area is a table with three columns: 'Rule', 'Action', and 'Post-Condition'. The table contains six rows of data.

	Rule	Action	Post-Condition
1	ZUYD Fridge Door previously open AND was in this state within t...		SELECT * FROM Sensors WHERE sensorid = 303 AND lastActiveValue = 1 and lastU
2	::500:: trays is Full AND has been in this state for more than 5 sec...		SELECT * FROM Sensors WHERE sensorid = 500 AND value = "Full" and lastUpdate
3	Location of ::3::Care-O-Bot 3.6 is ::602:: Kitchen Fridge in the ZUY...		SELECT locationId FROM Robot WHERE (locationId IN ( SELECT c.locationId FROM
4		SET ::19::cupLocation TO Tray	
5		SET ::10::UserDrinkStatus TO Waiting...	
6		Execute sequence 'gotoSofaLeft-SET...	

### 8.3.3 Message Creator

*Allows multi-language messages to be setup and subsequently used in behaviours and for action possibilities.*



The screenshot shows the Message Creator interface. At the top, there is a title bar 'Message Creator' and a user indicator 'rhUser:localhost UH'. Below the title bar, there is a 'Language' dropdown menu set to '::3:: Dutch'. Below that is a 'Message (drop down to see existing messages)' dropdown menu set to '::5:: Stuur me terug naar huis'. Below that is a text input field with the same message 'Stuur me terug naar huis'. At the bottom, there is an 'Add/Update message' button.

### 8.3.4 House Simulators

A number of house simulators were developed for pre-testing of scenarios. The example below is for the robot house at ZUYD.

The screenshot shows the 'ZUYD Apartment Sensor Simulator' interface. At the top, there is a 'Refresh' button and the text 'Running at: UH'. Below this, there are three main sections: 'User Location', 'Robot Location', and 'Sensors'. The 'User Location' section has a dropdown menu showing '::606:: Table Right Unload in the ZUYD Apartment'. The 'Robot Location' section has a dropdown menu showing '::603:: Door Left Side in the ZUYD Apartment'. The 'Sensors' section contains several checkboxes: 'Sofa 1 Occupied' (checked), 'Sofa 2 Occupied' (unchecked), 'Small Sofa Occupied' (unchecked), 'Fridge Door Open/Closed (Closed if ticked)' (checked), 'Cup Empty/Full (Full if ticked)' (checked), and 'Tray Empty/Full (Full if ticked)' (unchecked). There is also a 'Doorbell' button.

### 8.3.5 Session Control

This program was developed to ease experimental setup for scenario evaluation.

The screenshot shows the 'Session Control' interface. At the top right, it says 'UH'. The interface has several fields: 'Location' with a dropdown menu showing 'ZUYD Apartment', 'User' with a dropdown menu showing 'ZUYD User English::51::', 'Time Offset' in minutes (0) and 'Time Offset' in hours (0), and two time displays: 'Actual Time' and 'Experimental Time', both showing '12:29:31'.