An Agent-Based Cognitive Robot Architecture

Conference Paper · January 2013

DOI: 10.1007/978-3-642-38700-5_4

2 authors:

Changyun Wei
Hohai University
19 PUBLICATIONS **91** CITATIONS
SEE PROFILE

Koen V. Hindriks
Vrije Universiteit Amsterdam
251 PUBLICATIONS **3,944** CITATIONS
SEE PROFILE

An Agent-Based Cognitive Robot Architecture

Changyun Wei and Koen V. Hindriks

Interactive Intelligence, Delft University of Technology, The Netherlands
{C.Wei,K.V.Hindriks}@tudelft.nl

Abstract. We propose a new *cognitive robot control architecture* in which the cognitive layer can be *programmed* by means of the agent programming language GOAL. The architecture exploits the support that agent-oriented programming offers for creating *cognitive robotic agents*, including symbolic knowledge representation, deliberation via modular, high-level action selection, and support for multiple, declarative goals. The benefits of the architecture are that it provides a flexible approach to develop cognitive robots and support for a clean and clear separation of concerns about symbolic reasoning and sub-symbolic processing. We discuss the design of our architecture and discuss the issue of translating sub-symbolic information and behaviour control into symbolic representations needed at the cognitive layer. An interactive navigation task is presented as a proof of concept.

1 Introduction

The main motivation for our work is the need for a flexible, generic, high-level control framework that facilitates the development of re-taskable robot systems and provides a feasible alternative to the usual task- and domain-dependent development of high-level robot control. We believe that agent-oriented programming offers such an approach as it supports the programming of cognitive agents. Using agent programs to create the cognitive layer in a robot control architecture is natural and provides several benefits. It becomes relatively easy to adapt the control *at the cognitive layer* itself to various domains by exchanging one agent program with another. This approach is flexible and, if functionality of other layers is generic and can be used in multiple task domains, facilitates reuse. An agent-based approach, moreover, provides support for autonomous, reactive, and proactive behaviours and also endows a robot with the required deliberation mechanism to decide what to do next [1]. Of course, generality may come at a trade-off and does not imply that a generic architecture will always perform better than a dedicated robot control architecture [2].

Designing and developing a *cognitive robot control architecture* poses several challenges. Robots are embedded systems that operate in physical, dynamic environments and need to be capable of operating in real-time. A range of perception and motor control activities need to be integrated into the architecture. This poses a particular challenge for a cognitive, symbolic architecture as "it can be particularly difficult to generate meaningful symbols for the symbolic components of cognitive architectures to reason about from (potentially noisy) sensor

data or to perform some low-level tasks such as control of motors" [3]. Ideally, moreover, such an architecture should provide support for the integration or exchange of new and different sensors and behaviours when needed. Given the complexity and the number of components needed in a robot control architecture, one also needs to consider how all the processing components in the system communicate and interact with each other [4].

In this paper, we propose an agent-based cognitive robot control architecture that combines low-level *sub-symbolic* control with high-level *symbolic* control into the robot control framework. We use the agent programming language GOAL [5, 6] to realize the cognitive layer, whereas low-level execution control and processing of sensor data are delegated to components in other layers in the proposed architecture. GOAL, among others, supports the goal-oriented behavior and the decomposition of complex behavior by means of modules that can focus their attention on relevant sub-goals. GOAL has already been successfully applied to control real-time, dynamic environments [7], and here we demonstrate that it also provides a feasible approach for controlling robots. In our approach, the cognitive layer is cleanly separated from the other layers by using the Environment Interface Standard (EIS; [8]).

The paper is structured as follows. Section 2 briefly discusses some related work. Section 3 presents and discusses the design of the cognitive robot architecture. Section 4 presents a proof of concept implementation. Section 5 concludes the paper and discusses future work.

2 Related Work

Cognitive robots are *autonomous* and *intelligent* robot systems that can perform tasks in real world environments without any external control, and are able to make decisions and select actions in dynamic environments [9]. Here we discuss some related work that either explicitly aims to develop a cognitive architecture or uses some kind of symbolic representation for controlling a robot.

The work that is most similar in spirit to our own is that of [10] and [11]. In [10] the high-level language Golog is used for controlling a robot. Golog supports writing control programs in a high-level, logical language, and provides an interpreter that, given a logical axiomatization of a domain, will determine a plan. Golog, however, does not provide a BDI perspective on programming agents, and in [10] it does not discuss the robot control architecture itself. A teleo-reactive program, consisting of multiple prioritized condition-action rules, is used to control a robot in [11] the proposed robot architecture in is not discussed in any detail.

CRAM [1] is a software toolbox designed for controlling the Rosie robot platform developed at the Technische Universität München. It makes use of Prolog and includes a plan language that provides a construct for concurrent actions. The CRAM approach also aims at providing a flexible alternative to preprogrammed robot control programs. The main difference with our approach is that the reasoning and planning components are separated.

It has been argued that building robot systems for environments in which robots need to co-exist and cooperate with humans requires taking a *cognitive stance* [12] . According to [12], translating the key issues that such robot systems have to deal with requires a cognitive robot control architecture. Taking a cognitive stance towards the design and implementation of a robot system means that such a system needs to be designed to perform a range of *cognitive functions*. Various *cognitive architectures*, such as ACT-R [13] and SOAR [14], have been used to control robots. These architectures were not primarily aimed at addressing the robot control problem and in this sense are similar to agent programming languages, the technology that we advocate here for controlling robots. SOAR has been used to control the hexapod HexCrawler and a wheeled robot called the SuperDroid [3]. ADAPT (Adaptive Dynamics and Active Perception for Thought) is a cognitive architecture based on SOAR that is specifically designed for robotics [15]. The SS-RICS (Symbolic and Sub-symbolic Robotic Intelligent Control System) architecture for controlling robots is based on ACT-R; SS-RICS is intended to be a theory of robotic cognition based on human cognition [16, 2]. Unlike [17], we are not mainly concerned here with the long-term goal of developing robotic systems that have the full range of cognitive abilities of humans based on a cognitive architecture such as SOAR. Our work is oriented towards providing a more pragmatic solution for the robot control problem as discussed above. Although our work may contribute to the larger goal that [17] sets (as there are quite a few similarities between BDI-based agents and cognitive architectures such as ACT-R and SOAR), we do not address this question.

3 Cognitive Robot Control Architecture

This section introduces our cognitive robot control architecture. We discuss several issues including the processing and mapping of sensory data to a symbolic representation, the translation of high-level decisions into low-level motor control commands, and the interaction of the various architecture components.

3.1 Overall Design of the Architecture

Figure 1 shows the main components of our architecture, including a symbolic, cognitive layer (GOAL), a middle layer for controlling robot behavior (written in C++), and a hardware control layer (using URBI[1], a robotic programming language). The Environment Interface Standard (EIS) layer provides the technology we have used to manage the interaction between the symbolic and sub-symbolic layers. EIS provides a tool to deal with the issue that *sub-symbolic* sensory data is typically noisy, incomplete, quantitative measurements, whereas the symbolic layers need the *symbolic representation* that supports logical reasoning.

The main functions for controlling a robot are placed in the behavioural control layer. In addition to the functions such as (object) recognition, navigation,

[1] `http://www.urbiforge.org/`

localization, path planning and other common functions, this layer is also responsible for communicating with the higher-level symbolic components, including the interpretation of symbolic messages that represent actions and making the robot perform these actions in its physical environment.

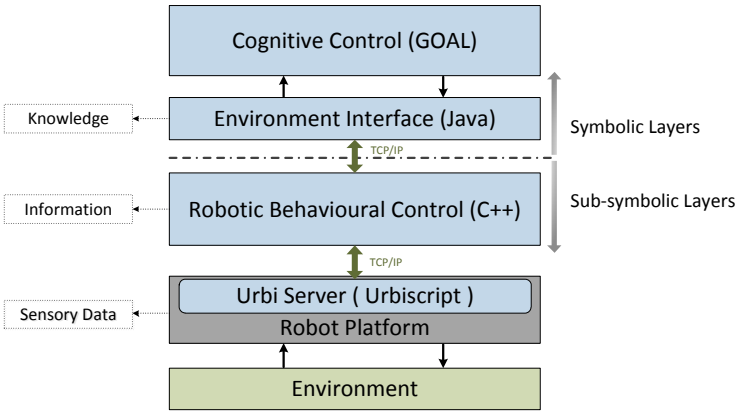

Fig. 1. The overall design of the architecture

The EIS layer acts as a bridge between the behavioural and cognitive control layers. Because these layers use different languages for representing sub-symbolic and symbolic information, respectively, we need an interface to translate between these representations. The cognitive control layer acts as a task manager for the robot and provides support for managing the robot's *mental state* which allows the robot to keep track of what is happening while executing a task.

3.2 System Architecture and Components

A robot control architecture provides an organizational structure for software components that control a robotic system [18]; such architectures are specialized because of the unique requirements that embedded systems such as robots impose on software. Here we discuss the detailed architecture presented in Figure 2 and also discuss some of the implementation details.

Robot Platform The architecture has been implemented on the humanoid Nao robot platform from Aldebaran Robotics[2]. We use the URBI [19] middleware that provides the *urbiscript* language for interfacing with the robot's hardware. We have chosen URBI instead of the Robot Operating System (ROS[3]) platform

[2] `http://www.aldebaran-robotics.com/`
[3] `http://www.ros.org/`

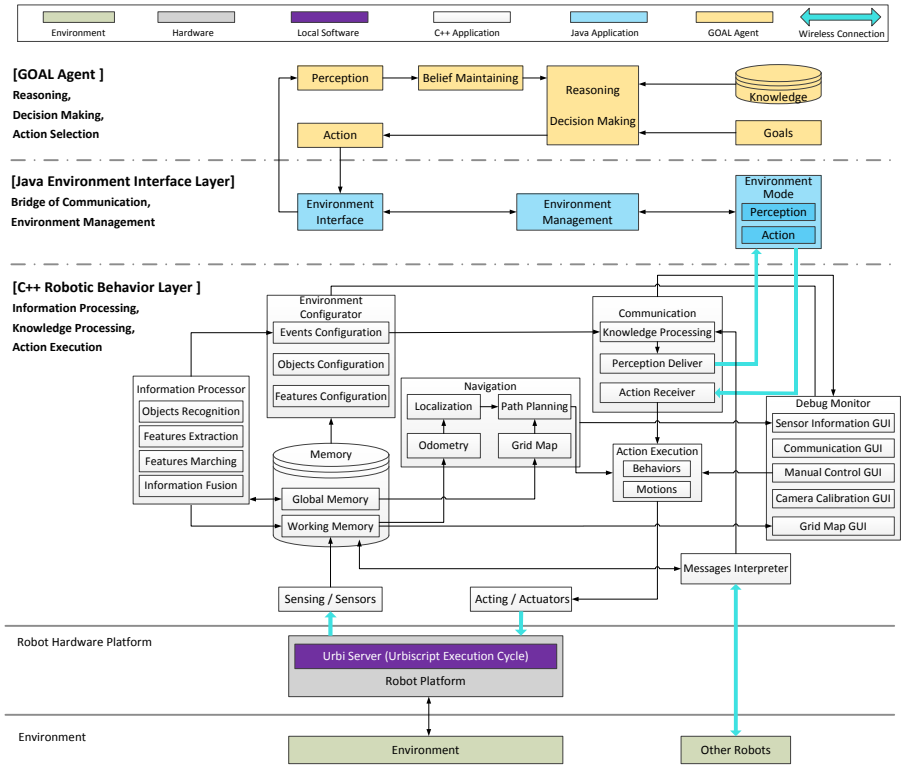

Fig. 2. Overview of the agent-based cognitive robot architecture

because it does not include the orchestration layers present in URBI that provide support for parallel, tag-based, and event-driven programming of behaviour scripts. When the architecture is executed, an urbiscript program, which initializes all API parameters of sensors and motors (e.g., the resolution and frame rate of camera images) is executed to configure the robot platform.

Behavioural Control The behavioural control layer is written in C++, connecting the robot hardware layer with higher layers via a TCP/IP connection. This layer is mainly responsible for information processing, knowledge processing, communication with the deliberative reasoning layer and external robots, and action and behaviour executions. All of these components can operate concurrently. The main functional modules involve:

– **Sensing**, for processing sensory data and receiving messages from other robots. Sensors that have been included are sonar, camera, microphone, an inertial sensor, and all sensors monitoring motors. Because the memory space required for camera images is significantly bigger than that for other sensors, the transmission of images is realized via a separate communication channel.

- **Memory**, for maintaining the *global memory* and *working memory* for the behavioural layer. In global memory, a map of the environment and properties of objects or features extracted from sensor data are stored. We have used a 2D grid map for representing the environment. This map also keeps track which of the grid cells are occupied and which are available for path planning. The working memory stores temporary sensor data (e.g., images, sonar values) that are used for updating the global memory.
- **Information Processor**, for image processing. This component provides support for object recognition, feature extraction and matching, as well as for information fusion. Information fusion is used to generate more reliable information from the data received from different sensors. We have used the OpenCV [20] library to implement algorithms and methods for processing images captured by the camera. Algorithms such as Harris-SIFT [21], RANSAC [22], and SVM [23] for object recognition and scene classification have been integrated in this module.
- **Environment Configurator**, for interpreting and classifying events that occur in the environment. For example, when the output of the left sonar exceeds a certain value, this module may send a corresponding event message that has been pre-configured. This is useful in case such readings have special meaning in a domain. In such cases, an event message may be used to indicate what is happening and which objects and features such as human faces and pictures have been detected.
- **Navigation** includes support for *Localization*, *Odometry*, *Path Planning* and *Mapping* components, which aid robots in locating themselves and in planning an optimal path from the robot's current position to destination places in a dynamic, real-time environment. Once a robot receives a navigation message from the cognitive layer with an instruction to walk towards the destination in the 2D grid-map space, the *Path Planning* component calculates an optimal path. After each walking step, *Odometry* will try to keep track of the robot's current position, providing the real-time coordinates of the robot for planning the next walking step. Due to the joint backlash and foot slippage, it is impossible to accurately estimate the robot's position just by means of odometry. For this reason, the robot has been equipped with the capability to actively re-localize or correct its position by means of predefined landmarks. Additionally, navigation requires back-and-forth transformation between coordinate systems. The odometry sensors, for example, provide the robot's coordinates in a World Coordinate System (WCS) that needs to be mapped to the robot's position in 2D Grid-map Coordinate System (GCS) for path planning. However, when executing walking steps, the position in GCS has to be transformed into the Local Coordinate System (LCS) that the robot uses.
- **Communication**, for delivering perception messages to the EIS component and receiving action messages from the EIS. The communication component is constructed based on a Server/Client structure and uses the TCP/IP protocol. The behaviour layer in a robot starts a unique server to which

a cognitive layer can connect as a client (thus facilitating the swapping of control from one robot to another).
- **Debugger Monitor** provides several GUIs that are useful for debugging robot programs, enabling developers to visualize sensory data and allowing them to set specific function parameters. This component also includes a *Wizard of Oz interface* to conduct human-robot interaction experiments.
- **Action Execution**, for instructing a robot to perform concrete behaviours and actions. The actions include motion movements such as walking and turning around while the behaviours include predefined body gestures such as sitting down, standing up and raising the arms of humanoid robots.

Environment Interface The environment interface layer built using EIS [8] provides support for interfacing the behaviour layer with the cognitive layer. The core components in this layer include an *Environment Model* that establishes the connection between cognitive robotic agents with external environments, an *Environment Management* component that initializes and manages the interface, and an *Environment Interface* component that provides the actual bridge between a cognitive agent and the behavioural layer.

Deliberative Reasoning and Decision Making The cognitive control layer provides support for reasoning and decision making. In our architecture, we employ the GOAL agent programming language for programming the high-level cognitive agent that controls the robot. GOAL is a language for programming logic-based, cognitive agents that use symbolic representations for their beliefs and goals from which they derive their choice of action. Due to space limitations, we do not describe GOAL agent programs in any detail here but refer the interested reader for more information to [5, 6].

3.3 Information & Control Flow in the Architecture

A key issue in any robot control architecture concerns the flow of information and control. Each component in such an architecture needs to have access to the relevant information in order to function effectively. In line with the general architecture of Figure 1 and layered architectures in general, different types of data are associated with each of the different layers, where EIS is a mediating layer between the behavioural and cognitive layer. At the lowest level all "raw" sensor data is processed yielding information that is used in the behavioural layer. The information present in the behavioural layer is abstracted and translated into knowledge by means of EIS that is used in the cognitive layer. Actions in turn are decided upon in the cognitive layer. These actions are translated by EIS into behaviours that can be executed at the behavioural layer. Finally, these behaviours are translated into motor control commands at the lowest layer.

Bottom-up Data Processing and Top-down Action Execution The processing of data starts at the bottom in the architecture and follows a strict bottom-up processing scheme whereas for action execution the order is reversed and a strict

top-down scheme is followed. At each of the layers different types of representations are used: urbiscript at the lowest, C++ data structures in the behavioural layer, and Prolog representations are used in the logic-based cognitive layer. Data processing starts by capturing values from various sensors. This sensory data then is analysed to extract useful features and to classify the data using basic knowledge of the environment. Fusion of different data sources is used to increase reliability. Finally, by using more high-level, semantic knowledge about the environment, symbolic representations are obtained by combining, abstracting and translating the information present at the behavioural layer by means of the EIS layer. The EIS layer provides symbolic "percepts" as input to the cognitive layer. Different aspects of the environment are handled by different techniques. For example, in order to recognize human faces, camera images need to be used and processed. First, several image frames need to be captured by the lowest layer for further processing in the behavioural layer which runs face detection or recognition algorithms. Using information obtained about the exact regions where a face is located in an image, further processing is needed to match these regions with a database of faces. Upon recognition, finally, a face can be associated with a qualitative description, i.e. a `name`, which is sent to the cognitive layer. The cognitive layer then uses this information and decides e.g. to say "Hello, `name`.".

The cognitive layer is the center of control and decides on the actions that the robot will perform. Actions are translated into one or more possibly complex behaviours at the behavioural layer. Of course, such behaviours, and thus the action that triggered these, take time and it is important to monitor the progress of the behaviours that are used to execute a high-level action. The cognitive layer delegates most of the detailed control to the behavioural layer but needs to remain informed to be able to interrupt or correct things that go wrong at this layer. One clear example concerns navigation. At the cognitive layer a decision to perform a high-level action `goto(roomA)` may be made which then is passed on to a navigation module at the behavioural layer. The latter layer figures out details such as the exact destination position of `roomA` on the 2D grid map and then plans an optimal path to the destination. Due to the unreliability of walking (e.g. foot slippage), the optimal path needs to be continuously re-evaluated and re-planned in real-time. This happens in the behavioural layer. However, at the same time, the cognitive layer needs to monitor whether the planning at the behavioural layer yields expected progress and may interrupt by changing target locations.

Synchronization of Perception and Action Part of the job of the EIS intermediate layer is to manage the "cognitive" load that needs to be processed in the cognitive layer. One issue is that the *Environment Configurator* in the behavioural layer typically will produce a large number of (more or less) identical percepts. For example, assuming an camera image frame rate of 30 fps, if a robot stares at an object for only one second, this component will generate 30 identical percepts. In order to prevent that a stream of redundant percepts is generated the EIS layer acts as a filter. However, this only works if in the behavioural layer objects

can reliably be differentiated and some assumptions about the environment can be made.

Another issue that needs to be handled by the EIS layer concerns the actions that are sent by the cognitive layer to the behavioural layer. Because behaviours have duration and the cognitive layer runs in parallel with the behavioural layer, the cognitive layer may send the same action multiple times to the behavioural layer or may send another action while an earlier action has not yet been completed. Because only some actions can be run in parallel, a decision may have to be made about which action has priority. For example, a robot can walk and talk at the same time but this is not the case for walking and sitting down. In the latter case, the behavioural layer needs to be able to gracefully terminate one behaviour in favour of another, or, in exceptional cases, ignore an action command from the cognitive layer.

3.4 Other Distinctive Properties of Our Architecture

In this section, we briefly discuss some distinctive properties of our architecture.

Decoupled Reactive and Deliberative Control Similar to most hierarchical architectures, we also distinguish reactive and deliberative control in our architecture. As usual, reactive control resides in the behavioural layer that is responsible for sub-symbolic information processing and behaviour execution, whereas deliberative control resides in the cognitive layer that is responsible for symbolic reasoning and decision making. In fact, these layers have been rigidly separated in our architecture by means of the EIS layer. In other words, in our design these layers have been completely decoupled. The main benefit of such a design consist of a clean separation of concerns. The price that needs to be paid for this strict separation is duplication of information and an additional effort of developers to design a well-defined interface between the two layers.

Decoupling of these layers facilitates the more or less independent programming of high-level, cognitive agents that control a robot as well as of lower-level behavioural functions. An agent programmer, for example, does not need to spend time handling object recognition. Similarly, a behaviour programmer who codes in C++ does not have to consider decision making nor even master the agent programming language used in the cognitive layer.

Multi-Goal Maintenance Often when robots try to accomplish a complex task, they will have to fulfil multiple goals [28]. Robots, however, cannot perform all of these goals at the same time either because goals may conflict with each other, or achieving a goal presupposes the achievement of others. As a simple example, when a robot needs to execute a command `"Pick up object B in place A"`, it actually has to fulfil two sub-goals in order to accomplish the goal; the robot needs to figure out that it first has to perform the action `goto(place(A))` and thereafter perform `pickup(object(B))`.

The cognitive layer realized by an agent programming language such as GOAL is able to handle multiple, possibly conflicting goals. GOAL [5] has a special operator to express that a goal needs to be achieved: `a-goal(`φ`)` as well as that (part of) a goal has been achieved `goal-a(`φ`)`. For instance, `goto(place(A))` can be expressed as an achievement goal as follows:

$$\texttt{a-goal(goto(place(A)))} \stackrel{df}{=} \texttt{goal(goto(place(A))), not(bel(in(place(A)))}$$

Goal achievement can be expressed by:

$$\texttt{goal-a(goto(place(A))} \stackrel{df}{=} \texttt{goal(goto(place(A)), bel(in(place(A))}$$

The handling of multiple goals has been demonstrated already successfully in real-time gaming environments [7].

Multi-Modal Communication Our architecture endows robots with multiple channels of communication with humans or other robots.

- **Semantic Representation for Human Robot Interaction** Since one of the objectives of developing intelligent robots is to offer better lives to humans, robots should be able to operate and help humans in daily activities. When interacting with humans, robots are required to incorporate communication and collaboration abilities. Human-robot interaction imposes special requirements on robot systems, and such interaction should be natural and intuitive for a human user. In particular, human-oriented interaction needs to take human abilities for effectively interacting into account. Symbolic knowledge representations could be accessible for human interpretation through precept-symbol identification by the designer [29]. Generally speaking, robot control systems need to adapt to humans by facilitating interaction that humans can understand. Our architecture supports knowledge processing and can produce semantic messages to facilitate interaction at the knowledge level with humans.
- **Shared Perceptions and Beliefs for Multi-Robot Interaction** From the perspective of multi-robot systems, robot systems are seldom standalone systems; however, robots should share their perceptions in order to interact with each other. For an individual robot, the perception messages will be transmitted from the sub-symbolic layers to the symbolic layers via an inner-communication channel, where the connection is constructed based on Server/Client structure in the TCP/IP protocol. The sub-symbolic layer in each robot starts a server that is unique for that robot, whereas the symbolic layer itself can have several clients, each of which can connect to either the server of its own or other robots. In this way, the robots cannot only receive the information from its own perceptions but also exchange their perceptions with other robots.

Our architecture provides the communication mechanisms for both inner-communication and external-communication, enabling robots to send and receive perception messages, and exchange mental beliefs present in the cognitive layer.

4 Experiment

To evaluate the feasibility of the proposed architecture, we have carried out a simple navigation task using a humanoid robot Nao (See Figure 3) built by Aldebaran Robotics. In this task, the Nao robot acts as a message deliverer which is supposed to enter the destination room and deliver a message to the people in the room.

Fig. 3. Humanoid robot Nao

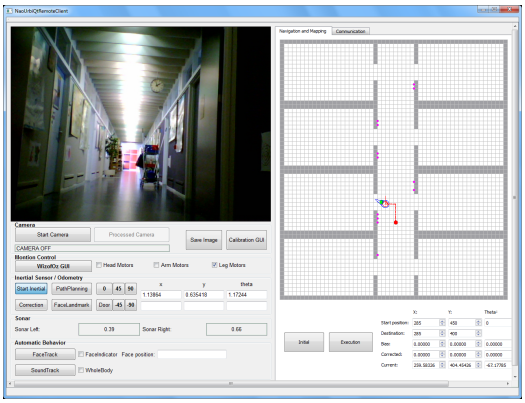

Fig. 4. The main GUI of the robotic behavior layer

The experiment is carried out in a domestic corridor environment. A predefined map has been built for the robot to localize itself. The robot begins walking from the starting place, and the goal of the robot is to enter the destination room, to show an arm-raising gesture, and eventually to deliver a coffee-break message. The behavioural layer processes the lower level functions; for example, Localization is used to estimate the robot's position, and real-time path planning is used to steer the walking of the robot. Meanwhile, it will communicate with and provide perceptual knowledge to the high-level cognitive layer. Figure 4 shows the main GUI of the behavioural layer, and Figure 5 shows the GUI for the cognitive layer implemented in GOAL.

The navigation task begins with a message `navigation` broadcasted from the behavioural layer to the cognitive layer, which indicates that both the low-level control and high-level control are ready to start. Then, the first action command `goto(roomA)` is generated. The Navigation module in the behavioural layer takes charge and steers the walking from the starting place to the destination room. When the robot arrives at the door of the destination room, a percept `at(roomA)` is sent to the cognitive layer that will generate the next action `enterRoom(roomA)`. After the robot enters the destination room, another percept `in(roomA)` is sent. Thereafter, when the robot has the belief of being in the destination room, it will perform the actions: showing `gesture(raiseArm)`

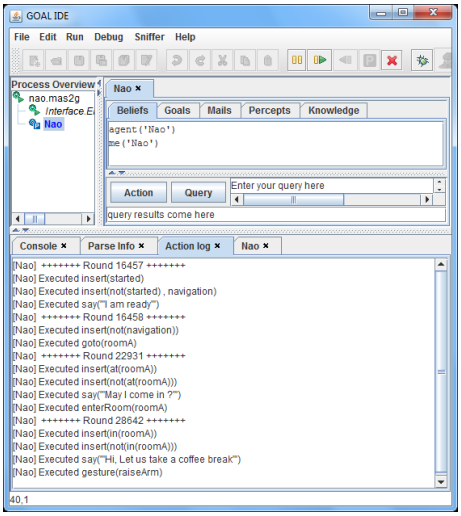

Fig. 5. The deliberative reasoning agent in Goal

and saying `"Hi, Let's take a coffee break!"`. In Figure 5, the action logs have been printed out to illustrate the decision making related to the task.

Although this navigation task is simple, it shows that uncertain, quantitative sensory data is mapped into symbolic perceptual knowledge and allows the robot to select their actions while performing tasks in a real world environment.

5 Conclusion and Future Work

The navigation task that the robot has performed in the above section demonstrates the feasibility of using a cognitive layer to control physical robots by means of agent-oriented programming. It also demonstrates that the clean separation of sub-symbolic and symbolic layers via the EIS component is feasible. Although our general architecture is similar to the layered approaches that have been used in many robot projects [24–27], we believe that the use of EIS provides a more principled approach to manage the interaction between sub-symbolic and symbolic processors. Of course, it is important that the cognitive layer (agent program) needs adequate perceptions to make rational decisions given its specific environment. Our architecture provides sufficient support from various components dealing with Perception, Knowledge Processing, and Communication to ensure this.

Future work will concentrate on applying our proposed architecture for multi-robot teamwork in the Block World for Teams environment [30], in which each robot can exchange their perceptions and share their mental states in the cognitive layer to coordinate their actions. To this end, in the low-level layers, we will integrate more image processing functions to endow robots with more perception inputs. Besides, we will also develop the adaptive ability so that robots can

learn or gain knowledge from experience. We believe this learning mechanism will facilitate robots to adjust to or understand environments more efficiently and reliably in the future.

References

1. Beetz, M., Mosenlechner, L., Tenorth, M. In: CRAM - A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments. IEEE (2010) 1012–1017
2. Kelley, T.D.: Developing a psychologically inspired cognitive architecture for robotic control : The symbolic and subsymbolic robotic intelligence control system. Advanced Robotic **3** (2006) 219–222
3. Hanford, S.D., Janrathitikarn, O., Long, L.N.: Control of mobile robots using the soar cognitive architecture. Journal of Aerospace Computing Information and Communication **5** (2009) 1–47
4. Hawes, N., Sloman, A., Wyatt, J., Zillich, M., Jacobsson, H., Kruijff, G., Brenner, M., Berginc, G., Skocaj, D. In: Towards an Integrated Robot with Multiple Cognitive Functions. Volume 22. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2007) 1548–1553
5. Hindriks, K.: Programming rational agents in goal. In: Multi-Agent Programming: Languages, Tools and Applications. Springer US (2009) 119–157
6. `http://ii.tudelft.nl/trac/goal` (2012)
7. Hindriks, K.V., van Riemsdijk, M.B., Behrens, T.M., Korstanje, R., Kraaijenbrink, N., Pasman, W., de Rijk, L.: Unreal goal agents. In: AGS 2010. (2010)
8. Behrens, T., Hindriks, K., Dix, J.: Towards an environment interface standard for agent platforms. Annals of Mathematics and Artificial Intelligence (2010) 1–35
9. Shanahan, M., Witkowski, M.: High-level robot control through logic. Event London (2000) 104–121
10. Soutchanski, M. In: High-level Robot Programming and Program Execution. (2003)
11. Coffey, S., Clark, K. In: A Hybrid, Teleo-Reactive Architecture for Robot Control. (2006)
12. Burghart, C., Mikut, R., Stiefelhagen, R., Asfour, T., Holzapfel, H., Steinhaus, P., Dillmann, R.: A cognitive architecture for a humanoid robot: a first approach. Architecture (2005) 357–362
13. Anderson, J.R., Lebiere, C.: The atomic components of thought. Volume 3. Erlbaum (1998)
14. Laird, J.E., Newell, A., Rosenbloom, P.S.: Soar: An architecture for general intelligence. Artificial Intelligence **33** (1987) 1–64
15. Benjamin, P., Lyons, D., Lonsdale, D.: Designing a robot cognitive architecture with concurrency and active perception. In: Proceedings of the AAAI Fall Symposium on the Intersection of Cognitive Science and Robotics. (2004)
16. Avery, E., Kelley, T., Davani, D.: Using cognitive architectures to improve robot control : Integrating production systems , semantic networks , and sub-symbolic processing. System **77** (1990)
17. Laird, J.E.: Toward cognitive robotics. Proceedings of SPIE **7332** (2009) 73320Z–73320Z–11
18. Bekey, G.A.: Autonomous Robots: From Biological Inspiration to Implementation and Control. The MIT Press (2005)

19. Baillie, J.C.: Urbi: Towards a universal robotic low-level programming language. 2005 IEEERSJ International Conference on Intelligent Robots and Systems (2005) 820–825
20. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
21. Azad, P., Asfour, T., Dillmann, R.: Combining Harris interest points and the SIFT descriptor for fast scale-invariant object recognition. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE (2009) 4275–4280
22. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24** (1981) 381–395
23. Mika, S., Schaefer, C., Laskov, P., Tax, D., Mller, K.R. In: Support vector machines. Volume 1. Springer (2004) 1–33
24. Qureshi, F., Terzopoulos, D., Gillett, R.: The cognitive controller: A hybrid, deliberative/reactive control architecture for autonomous robots. Innovations in Applied Artificial Intelligence 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert System IEAAIE 2004 **3029** (2004) 1102–1111
25. Arkin, R.C.: Integrating behavioral, perceptual, and world knowledge in reactive navigation. Robotics and Autonomous Systems **6** (1990) 105–122
26. Connell, J.H. In: SSS: a hybrid architecture applied to robot navigation. Volume 3. IEEE Comput. Soc. Press (1992) 2719–2724
27. Gat, E. In: Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots. Citeseer (1992) 809–815
28. Brooks, R.: A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation **2** (1986) 14–23
29. Vernon, D., Metta, G., Sandini, G.: A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. Evolutionary Computation, IEEE Transactions on **11** (2007) 151 –180
30. Johnson, M., Jonker, C.M., Riemsdijk, B.V., Feltovich, P.J., Bradshaw, J.M.: Joint activity testbed: Blocks world for teams (bw4t). Engineering Societies in the Agents World X **442** (2009) 433–442