Motivation-Based Autonomous Behavior Control of Robotic Computer

Blagovest Vladimirov, Hyun Kim, and Namshik Park

Electronics and Telecommunications Research Institute,
138 Gajeongno, Yuseong-gu, 305-700 Daejeon, Korea
{vladimirov,hyunkim,nspark}@etri.re.kr

Abstract. Successful development of a robotic computer as a mediator in smart environments requires providing a certain level of behavior autonomy to the robot and a capability to adapt its behavior in long-term interaction with the users. We attempt to identify core autonomy-related functionalities and describe the design and implementation of an autonomous behavior control subsystem that provides them. The Motivation Module is essential for providing a balance between the robot's autonomy and our ability to influence its behavior development in a long term. We present the results of two test scenarios illustrating basic use of the newly provided functionality.

Keywords: autonomous behavior, motivation, robotic mediator.

1 Introduction

The volume of available digital information and the variety of related services in our everyday lives are increasing. Access to these services is not constrained to the traditional computers anymore, as witnessed by the proliferation of such appliances as smart phones, tablets, smart TVs, etc., gradually bringing us toward the realization of smart environments.

In our work on the Future Robotic Computer (FRC) project [1], we explore the possibility of using a robotic computer as a mediator in smart environments. By building a robotic computer we aim at weaving relevant digital information in a flexible manner into the objects that surround us to support multi-modal, context-aware interactions. As an extension to the FRC's Software Framework, which supports creating of applications on the platform, we developed an Autonomous Behavior control Subsystem (ABS). In this paper we describe the Motivation Module (MM) of ABS and its role for supporting behavior autonomy.

2 Behavior Autonomy for FRC

The concept of autonomy has been analyzed in various contexts. Froese et al. [2] considered different aspects of autonomy in biological and artificial systems in relation to research on artificial life. They made a distinction between constitutive

I. Noda et al. (Eds.): SIMPAR 2012, LNAI 7628, pp. 16–27, 2012.

and behavioral autonomy. The former, focuses on the internal organization and the system's capacity for self-production and self-sustaining of organizational identity in respect to its environment. The latter is concerned with external behavior and is related to the stability and flexibility of the system's interactions with the environment. Many factors that affect the robot's autonomy are summarized in the ALFUS project [3], which aimed to devise a comprehensive approach for evaluation of the autonomy level of unmanned systems. In addition to the independence from human intervention, it also takes into account such factors as mission complexity and environment difficulty when comparing the levels of autonomy observed in the performance of different systems.

The issue of balancing the autonomy of a system and our ability to control it from outside to perform desired tasks has been discussed in [4]. On one hand, exercising too much control, limits the system's autonomy and shifts toward us the burden of taking care of mundane details. On the other hand, too much autonomy makes it more difficult to obtain specific, useful behavior from the system. The author suggested guidelines for designing cognitive architecture that is autonomous and inherently trainable, drawing on essential ideas from the field of Developmental Robotics [5,6].

For our purposes, we are interested primarily in the role of robot's autonomy in supporting useful behaviors while relieving us from the need to specify all details in advance. In respect to the works mentioned above, this means that we focus on behavior autonomy with relative independence from human intervention, while still retaining the ability to guide the behavior adaptation process. Therefore, we consider the following aspects of behavior autonomy: autonomous behavior execution; autonomous behavior selection; autonomous adaptation of existing behaviors; and autonomous initiation of behaviors.

The autonomous behavior selection and execution form the core of behavior autonomy. The abilities to adapt and generate new behaviors further increase the behavior autonomy through modifying the set of available behaviors in response to environmental changes. Finally, the ability to initiate behaviors not only in direct response to external events (e.g., user's command) but also based on internal motivation allows for implementation of proactive services.

Thus, our aim with implementing ABS is to provide capabilities for selection and execution of appropriate behaviors without explicit request from the user and for adaptation of behaviors in the process of interaction with the users and the environment.

3 Autonomous Behavior Subsystem Architecture

Cognition is necessary for behavior autonomy because it provides adaptive mechanisms for action selection based not only on past and present events but also on possible future consequences of the selected actions [6]. In a survey of artificial cognitive systems, Vernon et al. [7] distinguish three general groups: Cognitivist, Emergent, and Hybrid, depending on the underlying approaches. Cognitivist approaches are based on symbolic information representation and processing systems, while Emergent approaches employ connectionist, dynamical, and enactive

systems. The Hybrid approaches attempt to combine the strengths of the other two groups so that we can retain the ability to supply the system with relatively advanced initial knowledge and rely on the system's capabilities for adaptation and self-development in the process of interaction with the environment for further tuning of the desired behavior.

In ABS, we follow the general ideas of Hybrid approaches [8], using a dual-level architecture with both connectionist modules and symbolic rules. The interactions among these two levels, including top-down learning and bottom-up rule extraction, support the desired ability to specify some initial behaviors and later to adapt or refine them through appropriate interaction with the robot.

Depending on their goals and underlying approaches, various cognitive architectures emphasize different cognitive functions: reasoning, planning, memory, learning, etc. KnowRob [9] has a rich knowledge model supporting robot activities in a realistic household environment. It can also learn/adapt the action models and use them to plan the robot's behavior. On the other hand, the iCub Cognitive Architecture [6] emphasizes on starting with a core set of fundamental capabilities and developing the desired behaviors and functionality in the long-term interaction with the environment.

Compared to KnowRob, the ABS favors the learning of appropriate behavior selection instead of planning. Since with ABS, we aim more at a practical implementation rather than at a comprehensive solution of autonomous behavior control, unlike the iCub Cognitive Architecture [6], we adopt the hybrid, dual-level behavior selection based on CLARION's approach [8] that allows us to start with more complex initial behaviors. Finally, while CLARION has been used to model a diverse set of cognitive capabilities trying to approximate data from human performance on standardized cognitive tasks, with ABS we concentrate on issues specific to the robot's autonomous behavior.

ABS includes the following modules: Behavior Selection Module (BSM), Task Selection Module (TSM), Motivation Module, User Model (UM), Working Memory (WM), and Controller. BSM is at the core of ABS and its main purpose is to select appropriate behaviors. The other subsystems help to improve this selection process in various ways. The WM maintains task-relevant context which includes relevant history of previous events. The UM learns user preferred services. The MM contributes to the robot's autonomy by modeling internal drives so that the robot's behavior can be modulated indirectly, based on internal state.

The ABS structure, the internal communication among its modules and the external communication with DAS are shown in Fig. 1. In one behavior selection step, initially, ABS receives a sensory event from DAS. The sensory event is augmented with information maintained in the WM to form the current input context. The current input context is sent to the UM and if there is a user preferred service in the current situation, it is added back to the input context. Based on the input context, MM updates the internal drives' and goals' activations and returns the new values. Next, TSM selects a task that is appropriate for the input context. Finally, BSM selects the appropriate behavior for execution. The action specified by the selected behavior is sent to DAS, while

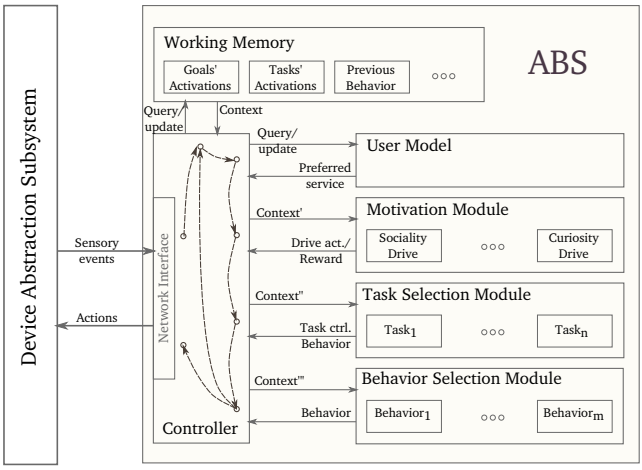

Fig. 1. ABS structure, internal communication among its modules and external communication with the Device Abstraction Subsystem

simultaneously the WM is updated with information that is deemed necessary for the system's functionality in the future.

Starting with the core functionality, below we provide detailed description of the ABS's modules.

3.1 Behavior and Task Selection

The primary purpose of the BSM is to select an appropriate behavior in a given context. In addition, BSM supports learning of behavior selection rules and adaptation of rules specified in advance based on reward signal computed by MM. Thus, the BSM partially implements the required ABS functionality to provide capacity for behavior selection and behavior adaptation. After explaining the common information representation used in ABS, we will describe rules and behavior representations, and finally the behavior selection and adaptation mechanisms.

A sensory event consisting of a set of features with their values is represented by a chunk containing a set of dimensions. A dimension d is a named, ordered set of tuples (v_i^n, v_i^a) that represents a given feature, where v_i^n is the name and v_i^a is the activation of the i^{th} value. For example, a chunk with one dimension *recognized_object* and tuples *(("book", 0.8), ("magazine", 0.2), ("pie_box", 0.0))* could represent an object recognition event with the values' names denoting the possible objects and the values' activations showing the probabilities assigned by the recognition algorithm.

We represent explicit symbolic rules as a combination of a condition chunk and an output chunk. The condition chunk is used as a prototype against which the context is matched [10], while the output chunk represents an action with its parameters. The degree of activation A_R of rule R, in a given input context

represented by an input chunk I, is computed using a distance metric between the dimensions specified by the rule's condition chunk and the dimensions with the same names that are present in the input context description, as shown in (1).

$$A_R = \sum_{d \in D_R^C} W^C \Big(\sum_{v \in V_d^C} U_{dv}^C W_d^C \Big(1 - \| A_{dv}^C - A_{dv}^I \| \Big) \Big) \tag{1}$$

Here, D_R^C is the set of dimensions of the condition chunk of rule R; V_d is the set of values that belong to the current dimension d; A_{dv}^C and A_{dv}^I are the activations of the current value v of the current dimension d of the condition and input-context chunks correspondingly; the dimension weight W^C is 1 if we have a disjunction of dimensions in the condition chunk or $\frac{1}{|D_R^C|}$ if we have a conjunction; similarly, the value weights W_d^C are 1 for disjunction or $\frac{1}{|V_d|}$ for conjunction of values in the dimension d of the condition chunk; and finally, the parameters U_{dv}^C are set to 0 for the values v of dimension d that we want to ignore, and to 1 otherwise.

In ABS, we model robot's behaviors (external or internal) with internal control structures representing situated actions, where actions are basic primitives specified and implemented in DAS. Consequently, as shown in Fig. 2, behaviors are represented by an output chunk that specifies the action with its parameters and a dual-level structure specifying the conditions for executing that action. The top level of that structure consists of an initial rule and an optional collection of candidate rules extracted from the bottom level (currently, all rules share the behavior's output chunk). The bottom level consists of a multi-layer neural network trained with Q-Learning [11]. The structure of the neural network is defined at initialization time. It has one unit in the output layer to represent the estimated Q-Value. The neural network weights are updated according to (2).

$$\Delta w_t = \eta \Big[r_t + \gamma Q_{t+1} - Q_t \Big] \sum_{k=0}^{t} (\gamma\lambda)^{t-k} \nabla_w Q_k, \tag{2}$$

where η is the learning rate, γ is the future reward discount factor, λ is the eligibility traces decay parameter, r_t is the reward, and Q_t is the Q-value of the selected behavior's action in the given input context at time t.

For a given behavior and a given specific input context represented as an input chunk, we compute the rules' activations at the top level. We present the same input context as an input pattern to the bottom-level neural network and obtain the estimated Q-value from the output. Then, the behavior's activation is computed as a weighted, linear combination of the maximum rule activation from the top-level and the estimated Q-value. The behavior selection is performed by comparing the behaviors' activations and selecting the maximum activated behavior following an ϵ-Greedy policy.

Behavior adaptation functionality is provided by optional learning processes in the top level, through maintaining performance statistics for each rule, and in the bottom level, through Q-learning based on reward signals from the MM.

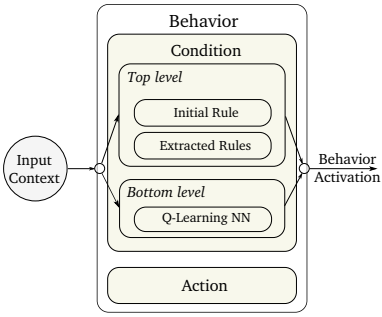

Fig. 2. Dual-level Behavior representation in ABS

In addition, top-down learning can be performed by using only the top-level rules to compute the behavior's activation and at the next step using the obtained reward to train the bottom-level network. In the current implementation, the following simple approach is used for bottom-up rule extraction. When the reward is above a pre-specified threshold, the input context used to compute the behavior's activation is transformed into a new candidate rule condition. This new rule is added to the collection of candidate rules if no similar rule exists already. In the consequent interaction, the rules' performance statistics are used to remove rules that fail to meet a pre-specified performance threshold.

The purpose of the TSM is to select an appropriate task for the current context. In ABS, a task is related to a set of behaviors that are used in combination to achieve some desired result. The TSM supplements BSM in implementing the required ABS functionality to provide behavior selection and adaptation capabilities.

The TSM has the same structure as the BSM. It consists of a set of *task-control behaviors*. For each task there are task-control behaviors with fixed actions for starting, suspending, resuming, and stopping the task, which modify the corresponding task-activation state maintained in the WM. This task-activation state is used in the behaviors' conditions in BSM to distinguish, when necessary, the behaviors that belong to the currently active task from the rest of the behaviors. Thus, the task selection functionality is provided by appropriate selection of task-control behaviors.

3.2 Motivation Module

The MM plays a role in the implementation of autonomous behavior initiation and in behavior adaptation functionality. The MM provides internal state that can be used in the behavior conditions to trigger or to inhibit behaviors. It also guides the behavior adaptation by computing reward signals used in the learning process.

This is achieved by implementing internal drives' models and a mechanism for computing reward signals from drives' activations and activation changes.

The activation state of the internal drives and goals is included into the input context used in TSM and BSM to compute the behaviors' activations. The reward signals are used in TSM and BSM to modify the network weights and the rules performance statistics of the relevant behaviors' conditions.

Currently, in ABS we have implemented two main internal drives: a *Sociality drive* and a *Curiosity drive*. The aim with implementing a Sociality drive is to balance behavior autonomy and external behavior control. It gives the users a mechanism for influencing the behavior adaptation process effectively, while providing for a certain level of autonomy. On the other hand, the Curiosity drive makes exploration-based behavior adaptation possible even without user intervention.

The model of the Sociality drive is shown in Fig. 3. The Sociality drive is configured with a collection of conditions that specify the effect of certain sensory events on the drive's activation level. When an input context is presented to the MM, the Sociality drive's activation d^s is computed as a linear combination of its conditions' activations a_i and a time component $b(t)$ as follows: $d^s = w_t b(t) + \sum_{i=1}^{n} w_i a_i$. The time component is included to allow increasing of the Sociality drive's activation in the absence of social interaction. A reward signal is computed from the Sociality drive's activation change, interpreting a decrease in the activation as a positive reward.

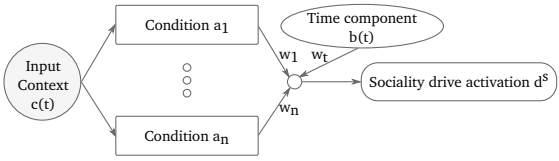

Fig. 3. Sociality drive model in MM

The model of the Curiosity drive is shown in Fig. 4. Due to the chosen implementation model of the Curiosity drive, in the process of exploration the robot develops a predictive model for the consequences of performing a given behavior. This model can be used to support behavior planning in further development of the ABS.

As shown in Fig. 4, the Curiosity drive model is based on the ability to predict relevant subset of the next input context. Currently, we use a neural network, trained on-line to learn the mapping from current context and currently selected behavior to the input context at the next step. The prediction performance is used to compute two components of the curiosity. The first component d_s^c is related to the 'surprise' or the unexpectedness of the obtained context and is computed from the prediction error as shown in (3)

$$d_s^c(t) = g d_s^c(t-1) + e(t), \tag{3}$$

where $e(t)$ is the normalized prediction error at the current time step and $0 \leq g < 1.0$ is a coefficient controlling the rate of decrease. The second component d_k^c

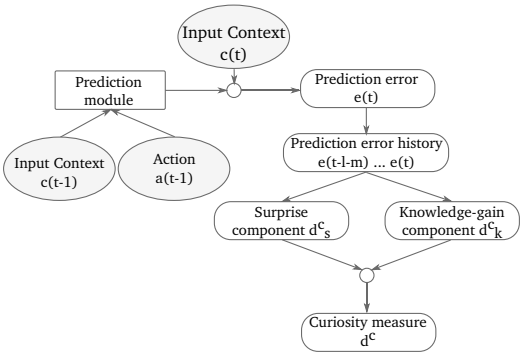

Fig. 4. Curiosity drive model in MM

is related to the knowledge gain and is computed as a decrease of the prediction error according to (4)

$$d_k^c(t) = \frac{1}{n}\sum_{i=0}^{n-1} e(t-i) - \frac{1}{m}\sum_{j=0}^{m-1} e(t-l-j), \tag{4}$$

where l is the span between the reference time points used in comparing the error, while n and m are smoothing parameters. Separate error history traces are kept for each behavior, thus the knowledge gain reflects the prediction performance for the specific behavior that was selected for execution.

The Curiosity drive activation is set to the curiosity measure d^c, which is a linear combination of the two components d_s^c and d_k^c. The reward signal from the Curiosity drive is proportional to the current drive activation.

The MM uses the computed drives' activations in predefined, linear combination dependencies to set the goals' activations. Also, a common reward signal $r = c_s r^s + c_c r^c$ is computed, where r^s and r^c are the reward signals from the Sociality drive and the Curiosity drive correspondingly, and c_s and c_c are coefficients balancing the contribution into the common reward signal.

3.3 User Model, Working Memory and Controller

The UM learns users' preferences from interaction. When a user requests some service (internally represented by a task), the UM associates the requested service with the perceived current context using probability-based associative memory. With time, the salient associations are used to extract explicit rules. A detailed description of the UM is given in [12].

The WM uses a collection of chunks to maintain relevant information necessary for setting drives' and goals' activations, suggesting services, and selecting tasks and behaviors.

The purpose of the Controller is to coordinate the interaction among the ABS modules and the external interaction with DAS. Through the interaction

with DAS, it provides the capabilities for collecting sensory information and for behavior execution.

4 System Implementation

The current implementation of FRC includes an Agent Unit shown in Fig. 5 and a server. The main components of the Agent Unit are two projector/camera pairs with five degrees of freedom, 3-channel microphone array, a stereo speaker set, and an embedded PC with wireless networking.

Fig. 5. The FRC's Agent Unit

DAS is a part of the FRC's Software Framework implementation ICARS (integrated control architecture for robotic mediator in smart environments) is described in [13]. ICARS consists of three layers that provide: a flexible communication/device model; an adaptive service model for the integrated robot control architecture; and a behavior-based high-level collaboration model. The ABS is implemented in C++ and the Controller communicates with ICARS over a TCP connection.

5 Test Scenarios

We present two test scenarios showing the role of the Curiosity drive in behavior initiation and inhibition.

5.1 Curiosity Driven Behavior Task 1

In this task, the Agent Unit observes the room around itself trying to detect user presence. The prediction module of the Curiosity drive has user presence from the previous time step and the last two performed behaviors as input and current user presence as target. If there is a user and if the Curiosity drive's activation is above certain level, then the Agent Unit approaches the user. In this scenario we have one task and eleven behaviors shown in Fig. 6 a). The Sociality drive and the UM are not used. The top plot in Fig. 6 b) shows the Curiosity drive's activation. The predictor starts from a random state and the initial hundred steps it learns that there is nobody in the room. The arrival

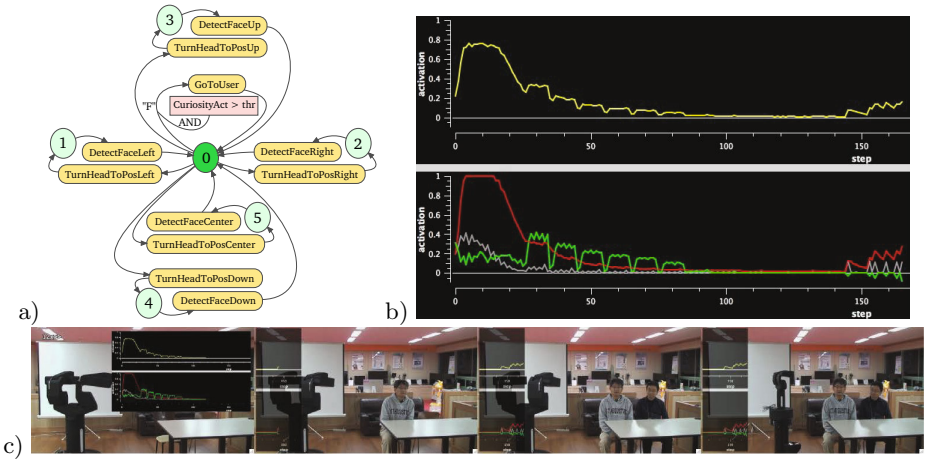

Fig. 6. Simplified task diagram a) and results for the curiosity test scenario b) and c). The rounded rectangles represent behaviors. The abbreviation "F" stands for face-detected sensory event. In b) the top plot shows the Curiosity drive activation while the bottom plot shows the 'surprise' (red) and knowledge-gain (green) components and the prediction error (gray). The snapshots in c) correspond approximately to the following time steps in b): 118, 145, 165, and 166 (not on the plot).

of the first user violates the expectation of *no-user-presence* and leads to an increase of the curiosity but it is still below the threshold. The increase of the curiosity from the arrival of the second user is sufficient to trigger the approaching behavior *GoToUser*.

5.2 Curiosity Driven Behavior Task 2

In this task, the user puts objects in front of the Agent Unit, which reacts by saying something related to the objects. Fig. 7 a) shows a simplified diagram of the five behaviors and their conditions. The Curiosity drive is related to predicting the appearance of unknown objects in result of *WatchTableTop* behavior. The plot in Fig. 7 c) shows the Curiosity drive's activation while known *Pie box*, initially unknown *Time magazine*, known *Wimpy Kid book* and already seen *Time magazine* are presented. Fig. 7 b) includes annotated snapshots from the Agent Unit's camera related to the events in c).

As can be seen from diagram a), the User Agent will ask a question if an unknown object is presented and the Curiosity drive's activation is above a specified threshold. Fig. 7 d) is a plot of the Curiosity drive's activation from a simulated run where unknown objects are presented in a sequence to illustrate a behavior inhibition effect. After a while, the predictor learns to expect unknown objects, the curiosity drive's activation falls below the threshold (0.1) and the Agent Unit stops asking questions.

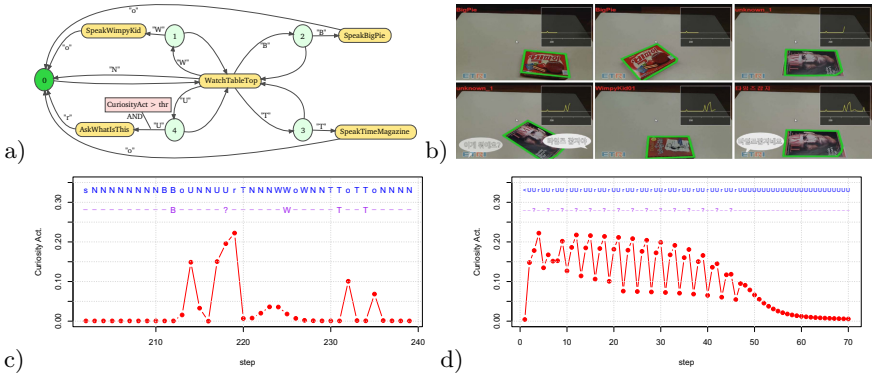

Fig. 7. Simplified task diagram a) and results for the curiosity test scenario b), c) and d). The rounded rectangles represent behaviors with the labels on incoming arcs showing the conditions for execution and the labels on the outgoing arcs show the resulting sensory event. The abbreviations stand for "N" no object, "W" 'Wimpy Kid' book, "T" Time magazine, "B" 'Big Pie' box, "U" unknown object, "o" task finished, "r" response from the user. In c) and d) the top-row labels show sensory input and the bottom-row ones show the behavior, where "-" is WatchTableTop, "?" is AskWhatIsThis.

6 Conclusions

The development of ABS aims at augmenting the FRC's Software Framework by providing behavior control with autonomous behavior selection, initiation and adaptation functionality. An important role in this process is played by the MM which, in a certain sense, shifts the balance of control over the robot's behavior toward the robot itself. In future work on the FRC, we will develop more application-oriented scenarios with long-term, user interactions and curiosity-based behavior adaptation. A promising direction of development is to use FRC to help the user to modify its own attitudes and habits as envisioned by Fogg [14]. One example could be to devise internal drives that guide the Agent Unit's behaviors to provide adaptive, timely, context-dependent opportunities and cues to aid the user in acquiring healthier lifestyle habits.

Acknowledgment. This research has been supported by Korean Ministry of Knowledge Economy and Korea Research Council for Industrial Science & Technology [Grant No. 2010-ZC1140, Development of Future Robotic Computer].

References

1. Kim, H., Suh, Y.-H., Lee, K., Vladimirov, B.: Introduction to system architecture for a robotic computer. In: Int. Conf. Ubiquitous Robots and Ambient Intelligence (URAI), pp. 607–611 (November 2011)

2. Froese, T., Virgo, N., Izquierdo, E.: Autonomy: a review and a reappraisal. In: Proc. of 9th European Conference on Artificial Life, Berlin, Germany (2007)
3. Huang, H., Pavek, K., Novak, B., Albus, J., Messin, E.: A framework for autonomy levels for unmanned systems (ALFUS). In: Proc. of AUVSI's Unmanned Systems North America, Baltimore, Maryland (2005)
4. Vernon, D.: Reconciling autonomy with utility: A roadmap and architecture for cognitive development. In: Proc. of Int. Conf. on Biologically-Inspired Cognitive Architectures, pp. 412–418. IOS Press (2011)
5. Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M., Yoshida, C.: Cognitive developmental robotics: A survey. IEEE Trans. Autonomous Mental Development 1(1), 12–34 (2009)
6. Vernon, D., von Hofsten, C., Fadiga, L.: A Roadmap for Cognitive Development in Humanoid Robots. Cognitive Systems Monographs (COSMOS), vol. 11. Springer (2011)
7. Vernon, D., Metta, G., Sandini, G.: A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. IEEE Trans. Evolutionary Computation 11(2), 151–180 (2007)
8. Sun, R.: The importance of cognitive architectures: An analysis based on CLARION. J. Experimental and Theoretical Artificial Intelligence 19(2), 159–193 (2007)
9. Tenorth, M., Beetz, M.: KnowRob – Knowledge Processing for Autonomous Personal Robots. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 4261–4266 (2009)
10. Duch, W., Setiono, R., Zurada, J.: Computational intelligence methods for rule-based data understanding. Proc. IEEE 92, 771–805 (2004)
11. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, University of Cambridge, Cambridge, England (1994)
12. Koo, S.-Y., Park, K., Kwon, D.-S.: A dual-layer user model based cognitive system for user-adaptive service robots. In: 20th IEEE International Symposium on Robot and Human Interactive Communication (Ro-Man 2011), pp. 59–64 (2011)
13. Suh, Y.-H., Lee, K.-W., Lee, M., Kin, H., Cho, E.-S.: ICARS: Integrated Control Architecture for the Robotic mediator in Smart environments A Software Framework for the Robotic Mediator collaborating with Smart Environments. In: 9th IEEE International Conference on Embedded Software and Systems (ICESS 2012), pp. 25–27 (2012)
14. Fogg, B.J.: Persuasive Technology: Using Computers to Change What We Think and Do. Morgan Kaufmann, San Francisco (2003)