Robots meet IVAs: A Mind-Body Interface For Migrating Artificial Intelligent Agents

Michael Kriegel[1], Ruth Aylett[1], Pedro Cuba[2], Marco Vala[2], and Ana Paiva[2]

[1] School Of Mathematical and Computer Sciences
Heriot-Watt University,
EH14 4AS, Edinburgh, Scotland, UK
{michael,ruth}@macs.hw.ac.uk
[2] INESC-ID, Av. Prof. Cavaco Silva,
2780-990 Porto Salvo, Tagus Park, Portugal
pedro.cuba@gaips.inesc-id.pt,{marco.vala,ana.paiva}@inesc-id.pt

Abstract. We describe CMION, an open source architecture for coordinating the various sensors and effectors of an artificial intelligent agent with its mind, i.e. the high level decision making processes. The architecture was designed to work for virtual graphical agents, including those on mobile devices, as well as robots. Its built-in migration feature allows a character to move between these differing embodiments, inhabiting them in turn. We emphasize the importance of modularity for an architecture supporting migration and highlight design decisions promoting modularity in CMION. An applied example of the architecture's use in a migration situation is given.

1 Introduction

Autonomous robots have been researched since the 1970s, but for a long time a functional task-oriented perspective dominated work. Much more recently [11] work has been carried out into social robotics in which the integration of robots into human social environments is the guiding focus. In contrast, work on intelligent graphical agents began much later [3] but interaction with human users has been researched almost from the start, with the development of Embodied Conversational Agents [4] or ECAs.

The LIREC project [3] (Living with Robots and intEractive Characters) is a project investigating long-term interaction that brings these two perspectives together. Its programme combines the development of an innovative technological infrastructure and scientifically-constructed user studies in an attempt to move beyond the novelty effect of both social robots and ECAs to social companions that can play an acceptable long-term role. Both social robots and ECAs are embodied, the former physically and the later virtually. Physical embodiment raises still unsolved engineering problems of power sources, mobility and localization that typically limit the ability of robots to accompany humans as they

[3] http://lirec.eu

move from one social environment to another - for example from home to work. Virtual embodiments are much more transportable but by their nature do not support physical task execution (fetch and carry for example).

For this reason, LIREC investigates migration, the ability of a synthetic companion to move from one embodiment to another. This of course raises a new set of research questions, of which the most important is: what exactly migrates? We define this as the companion's identity, by which we mean those features that persist and make it unique and recognizable from the user's perspective.

These features, themselves a research topic [20], may include common attributes of the different embodiments, for example similar facial appearance, but also common aspects of interactional behaviour, such as emotional expressiveness, and a memory of events and interactions that have taken place in multiple embodiments.

The question of what migrates also requires a technological answer. In this paper we consider the impact of migration on the architecture of such a companion. It is clear that a degree of architectural commonality across embodiment platforms is required if migration is to be a generic capability of LIREC companions. Companions may run on a number of different robots, on handheld devices, and fixed graphics installations. Creating a common architectural framework is in our view an innovative enterprise given that most researchers in robotics and ECAs have so far been involved in separate explorations of architectures.

2 An architecture for migrating companions

In this section we first explain our requirements for a social companion's architecture, then extend on those requirements for the specific case of migrating companions and then compare our requirements with related work.

2.1 Basic requirements

In the context of this paper the term architecture refers to a mind-body interface, i. e. a software framework that links the companion's *mind* (high level deliberative processes working with symbolic representations of the world) with it's *body* (a collection of sensors and actuators for the physical or virtual world).

The most important task of a mind-body architecture is thus the provision of a framework for the bidirectional mapping of information to different levels of abstraction, from raw sensory data to symbolic data and vice versa. For example images from a camera could be mapped by image processing and affect recognition algorithms into symbolic descriptions of what the companion is seeing, e.g. whether a user is present and what affective state the user is currently displaying. The mind processes this information, deliberates and acts upon it, for example it might decide to cheer the user up if they are sad. The architecture then needs to map the symbolic action of cheering up to actual actuator values.

One can identify three layers in such systems where layer 3 is the mind, layer 2 the mind-body interface and layer 1 the body consisting of sensors and

effectors. This concept of decomposing an embodied agent into three layers is very common and can be found in the majority of robotics control architectures [13]. Conceptually the same distinction of 3 layers can also be found in the more recent and ongoing efforts of the ECA community to create a common framework for multimodal generation named SAIBA [15]. In the SAIBA context the 3 layers are called Intent Planning, Behaviour Planning and Behaviour Realization.

In three layer architectures typically control of the behaviour of the agent does not exclusively reside on the highest level. Instead the more common and flexible approach is to grant each layer some autonomy. In the case of a robot that means for example that time-critical control loops for processes like obstacle avoidance reside at the lowest level, with the involved perception data never having to travel up to the 2 higher layers. A process taken care of by the middle layer could be for example navigation (path-finding), while the top layer deals with high level task-driven behaviour.

2.2 Additional requirements for migrating companions

Sufficient level of abstraction The separation of mind and body discussed above fits neatly with the concept of migration, which requires a companion's mind to leave one embodiment and enter another one, a process undeniably made easier if mind and body are clearly separated. A mind-body interface for migration that is installed on a certain embodiment must therefore support the dynamic exchange of the mind (layer 3) component.

Once a mind enters a new body it needs to know how to deal with this new body's functions. The mind deals with abstract descriptions of behaviour while the mind-body interface needs to translate them into concrete embodied behaviour that suits the embodiment. For example the mind might simply want to move to a certain destination, but the mode of getting there could be very different depending on the embodiment (e.g. flying, swimming, driving, walking).

Modularity When designing a scenario involving a companion migrating across multiple embodiments it is of great importance for us to minimize the development effort, especially when some embodiments have things in common. This underlines the need for a highly modular mind-body interface that promotes reusability of components.

Flexible definition of identity We defined migration earlier as a transfer of the companion's identity to a different embodiment and described how this could be achieved by exchanging the companion's mind. While a majority of the identity resides within the mind (e.g. memory, personality, goals, etc) this is clearly not all there is to identity.

There are lower level features that form part of an agent's identity as well that would also be desirable to migrate if they can be used in the new embodiment and the architecture should allow for this. For example, the voice of a companion could be argued to form an important part of its identity. Unless the companion

migrates into an embodiment that does not support the use of voices (e.g. a dog robot that can only bark) it would be desirable to migrate the voice in one form or another.

Multiple platforms Finally, as mentioned in the introduction to this paper, the LIREC project uses a variety of different companion embodiments, from various mobile and static robots to ECAs and companions on mobile devices. An architecture for LIREC must therefore run on all of these platforms sufficiently well, which is of increased interest on mobiles with limited memory and computing power and on robots that require fast response times for tasks like navigating safely in the physical world.

2.3 Related Work

When examining existing architectures we found most of them not to be generic enough for our purposes and either geared towards robotics (e.g. [2], [12]) or behaviour realization for virtual agents (e.g. [5], [8], [21]). The same applies for the work of Cavazza and colleagues in the COMPANIONS project [6], which shares the exploration of long-term relationships with interactive companions with LIREC, but focuses much more on natural language interaction with ECAs and less on robots. While [17] presents an architecture that was tested with both ECAs and robotic heads, its focus is also on natural language interaction and issues relating to robot bodies that move in and manipulate the physical world are not addressed by it.

Some groups have explored the idea of migration before (e.g. [14], [18]) but the development of a reusable generic architecture seems not to have been the focus in these works. This is not the case of the agent chameleon project which has published details about their architecture for migrating agents [19]. However, rather than a mind-body interface, the agent chameleon architecture is a complete architecture including the deliberative layer. This made it not suitable as an architecture for the LIREC project which employs its own agent mind and implements memory and theory of mind mechanisms within it.

Not so much complete architectures, but nevertheless widely used and promoting our requirement of modularity, are middlewares such as YARP [10] and Player [7] which are both popular in the robotics field and Psyclone [1], which has been applied in the field of ECAs before. In order to not reinvent the wheel we have based our architecture as described in the next section on such a piece of middleware, the agent simulation framework ION [22] and derived from it the name CMION (Competency Management with ION).

3 Technical Overview

The CMION architecture designed for the migrating companions of the LIREC project was written in Java and uses only those features of the Java API that are also supported by the Android mobile OS, making it compatible besides PCs

with a wide range of mobile devices. Figure 1 gives a simplified overview of the components in CMION.

Functionalities of an embodiment are encapsulated inside constructs called competencies, which can be thought of as miniature programs that each run in a separate thread and that are directly linked to the sensors and actuators of the embodiment. Competencies can exchange information with each other by writing information onto the blackboard component, which is a hierarchical data storage container, which can contain properties (named data storage slots) and sub containers, which can again contain properties and sub containers. The world model component is another data storage container, whose content is organized in the same way, but in contrast to the blackboard it constitutes an external representation of the knowledge base that the agent mind operates on. Competencies can make changes to the world model which are then propagated to the agent mind and treated as perceptions.

If the agent mind decides to act, the competency manager component maps the action to a predefined competency execution plan consisting of a number of competencies that realize the requested action in the embodiment. The plan also contains specific running parameters for each competency involved and synchronization information regarding the running order and concurrency of the competencies. The structure and information content of a competency execution plan can be roughly compared to behaviours annotated in BML [15]. The competency execution system finally schedules the execution plan by invoking, starting and monitoring the competencies involved in the plan at the right time and with the parameters specified by the plan. Instead of being invoked through the execution system, certain competencies, especially those involved with sensing, can instead also be started immediately when the architecture is loaded and run continuously in the background.

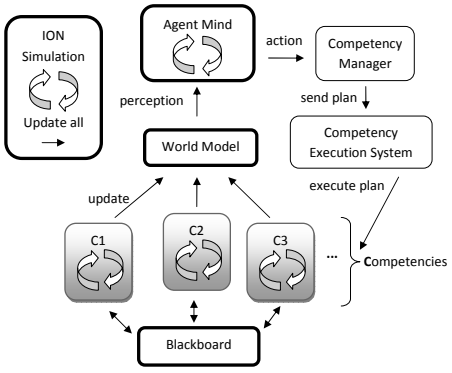

Fig. 1. CMION Architecture Overview

3.1 Communication

By basing the CMION architecture on the ION framework middleware [22] we increased performance, efficiency and modularity. The ION framework was originally designed as a multi-agent simulation environment for virtual worlds that allows to model and simulate the world state of the environment independently of the realization engine. Based on the observer design pattern it provides a request/event communication model that is synched with a simulation loop. In the case of CMION, every component, i.e. every box in Figure 1 (e.g. the world model, blackboard, agent mind, every competency, etc.) is an ION entity that can raise events, receive requests and register request or event handlers.

The difference between requests and events is subtle: while events are used to broadcast information, requests are directed towards a single entity and the scheduling entity has no guarantee that the receiving entity will actually carry out the request. For this reason it is a good practice for entities to inform others about the result of carrying out requests by using events. Moreover, scheduling entities should not assume success until the respective event is raised. While events are handled one by one, the simulation collects all requests that are scheduled with a certain entity within one update and the request handler processes them all together in a single call. This allows the mediation of conflicts when 2 or more contradicting requests are scheduled within the same update, e.g. when two competencies attempt to write different data onto the same property on the blackboard. All arrows in Figure 1 thus are either ION requests or events. This allows efficient and conflict-free communication between all the components, which is especially important for the competencies and the agent mind which run asynchronously in their own threads.

3.2 Modularity

Modularity is promoted in several ways by CMION. Through the type of inter-component communication provided by the ION framework it becomes very easy to add additional components to the architecture without making any changes to the existing code. Cases where this becomes useful include monitoring and debugging tools that can simply listen to certain events or schedule certain requests. CMION additionally supports this by dynamically loading components specified in an xml file at startup. This means additional components can be developed without recompilation of the architecture. The same also applies to competencies: they are separately compiled from the main architecture, specified in an xml file (the competency library) and dynamically loaded at startup. This allows sharing, adding and swapping competencies. The rules that map actions from the mind to competency execution plans are similarly defined via xml and loaded when the architecture is started.

Modularity is also achieved through object oriented design. Besides the abstract base classes for competencies and the agent mind, we also provide further specialised base classes that allow the remote connection of an agent mind or competency through a tcp socket or the middlewares YARP [10], Psyclone [1]

and SAMGAR [9]. Besides greater modularity in the case of robotic embodiments remote connecting a competency can also make sense in order to have the implementation in a language different from Java, possibly resulting in increased performance and reduced reaction time for time critical processes.

4 Migration

Now that we have given an overview of CMION we can proceed in describing how the process of migration works. For migration to be possible several instances of CMION running on several embodiments are needed. Each instance will likely contain a slightly different set of competencies and library of competency execution plans that reflect the abilities of its embodiment. Initially when each CMION instance is started up, its embodiment can either be inhabited or uninhabited.

Inhabited embodiments are considered "'alive"' while uninhabited embodiments are initially empty vessels waiting to be inhabited through a migration. How the agent mind should be initialized for inhabited and uninhabited embodiments respectively depends on the concrete implementation of the agent mind. However it is advisable to design the agent mind so that data and processes are separated. This is the case with our LIREC agent mind, so that when initializing an inhabited embodiment the agent mind data needs to be specified (planning data, personality, previous memory, etc.) whereas it can be left empty when initializing an uninhabited embodiment.

4.1 Outgoing Migration

Migration is implemented as an actuating competency and thus can be initiated by the agent mind like other competencies. How the mind decides to migrate is dependent on the concrete mind implementation and world modelling. It is relatively straightforward to integrate migration into the decision making process of an agent mind. The approach we take in companions developed for LIREC, where we use an agent mind based on a continuous planner, is to model migration as a planning operator that changes certain properties related to embodiment and location. The agent mind will then automatically include migration actions into its plans whenever the current task requires it. Conceptually performing a migration is not supposed to alter an agent's emotional state or personality unless one would conceive a scenario where migrating for some reason is a painful or joyful experience for the agent. Either way for the implementation of migration in CMION this is of no concern as this issue is again dependent on the implementation and configuration of the agent mind used.
When the migration competency in CMION is executed it first looks up the network address of the target embodiment. A network connection is established and if the target embodiment is already inhabited, the migration will obviously fail. Otherwise, the migration competency will raise an ION event stating that a migration is about to occur. It will then wait for one ION simulation update,

a time frame during which any component (competencies, agent mind, etc.) in its event handler to the migration event can append data in XML format to migrate to the migration competency. Whether components provide this handler and what data they append depends entirely on their developer and what part of the state of the component is related to the agent's identity.

For companions developed in LIREC we typically append data describing the entire state of the agent mind and the partial state of a few but not most competencies. Since the ION event handlers are processed within the ION simulation thread there is no need to worry that the data might be appended too late and miss the migration. However, there are cases where one might want to gather the data asynchronously, for example because it is expected to take a while and the ION simulation should not stall during this time. For events like this the component can request a migration wait and resolve it later once it has finished gathering data. The migration competency keeps track of all waits requested and will only proceed once all of them have been resolved. It will then package all appended data into one XML document and transfer it to the target embodiment. If the transfer was acknowledged by the receiver as successful the sending embodiment is marked as uninhabited. An agent mind should (as the LIREC agent mind does) handle a successful outgoing migration event and go into a sleeping state.

4.2 Incoming Migration

On the incoming embodiment an event is broadcast that a migration has occurred. Every component that has registered a handler for this event will be able to query whether the received migration XML document contains a section marked as belonging to this component and, if this is the case, receive and parse the data. The embodiment is then marked as inhabited and the previously sleeping agent mind is woken up. It now contains the personality, memory, knowledge base, action repertoire, goals, etc. that previously resided in the source embodiment. This could however present a problem as parts of the knowledge base are a symbolic description of the environment in which the agent is embodied and might be wrong in the new embodiment. For this reason, immediately after loading the migrated state an agent mind should query the world model in the new embodiment, which contains a symbolic description of the new environment and replace the section of its knowledge base containing embodiment specific knowledge with the world model contents.

5 Example scenario

We now provide a description of a real migration scenario to illustrate the functionality of the CMION architecture. The example is taken from the Spirit Of the Building scenario (see Figure 2), one of the actual companion scenarios developed in LIREC. In this scenario a group of researchers share their office with a robotic companion that assists the team in certain daily tasks. When one day

they expect a visitor but are uncertain of his time of arrival they ask their companion to migrate to the screen installed at the building entrance and to wait there for the visitor and greet him. This scenario could continue with the agent migrating to the visitor's phone to guiding him to his destination and then finally migrating back into the robot embodiment. We will only focus on the first migration here.

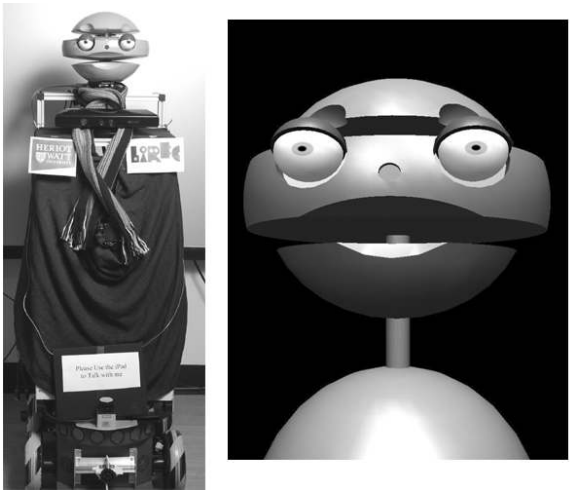

Fig. 2. The robot and virtual character embodiment of the Spirit Of the Building companion showcase, that the example scenario is derived from

5.1 Technical Realization

Figure 3 shows the state of the 2 embodiments in the example before the task to greet the guest has been activated. The robot embodiment (A) is inhabited and has an active agent mind containing data, while the ECA (B) is uninhabited with an empty inactive mind. Both embodiments have a different set of competencies but share certain of them. Similarly both embodiments have some differences in their execution plans. While the robotic embodiment A can move in physical space, B can't so the way they realize a speech act differs. In both cases the plan specifies the execution of the language generation competency that translates a speech act symbol into an actual utterance followed by a parallel execution of speech synthesis and animation competency to achieve lip synch[4]. In embodiment A however, additionally the robot first needs to move within an

[4] The speech synthesis and animation competencies exchange synchronization data via the black board.

acceptable communication distance using the Navigation competency, which can be executed in parallel with the language generation.

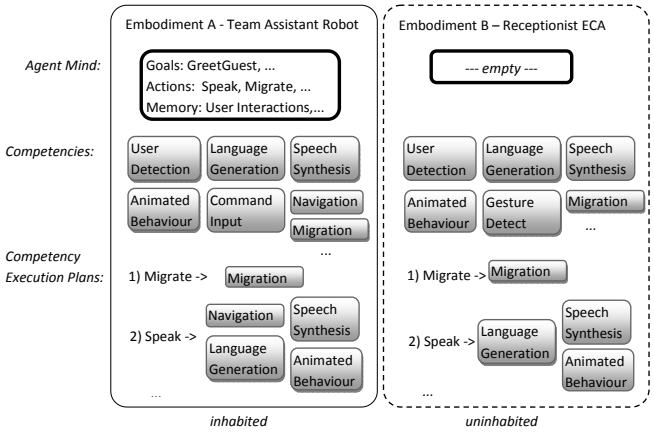

Fig. 3. Initial state of the embodiments

The steps that initiate the migration in the robot embodiment would now involve the following:

- The command input competency detects a user's command to greet a guest and passing it to the agent mind via the world model.
- This activates the GreetUser goal in the agent mind.
- The planner in the agent mind knows that a pre-condition for waiting for the user is being embodied in the reception ECA and it also knows that a migration action can change the embodiment.
- The agent mind thus sends a migration action with the reception ECA as a target for execution.
- This is mapped by the competency manager to the competency execution plan 1, which passed on to the competency execution system invokes the migration competency.

The migration competency as described before raises an event that a migration is about to occur. The agent mind and some of the competencies append data as shown in Figure 4.

After the migration, A is uninhabited and B is inhabited. The agent mind's plan to greet the user resumes now on B with the agent waiting. The User Detection competency notifies the agent mind via the world model, when a user has approached the screen. The agent will then ask the user whether they are the expected guest (Speak action). Via the Gesture Detect competency a yes/no answer to this question is obtained. If the answer is yes the scenario then might proceed as outlined earlier.

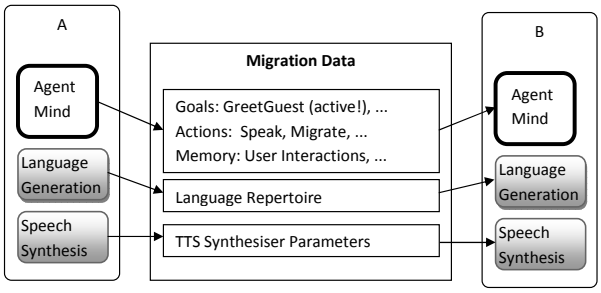

Fig. 4. The data that is being migrated

6 Conclusion

In this paper we have described CMION, a novel architecture that provides a mind-body interface for embodied (robotic or virtual) agents with specific support for migration between different embodiments.

While we motivated migration as a generic method of overcoming the different limitations of physical and virtual embodiments, the scenario discussed above makes it clear that migration is not the only solution. In this scenario, the receptionist could be implemented as a different entity from the team assistant with the two agents communicating rather than one agent migrating. Alternatively, the same agent could have multiple embodiments in several places at once.

The multiple agents solution is not at some implementation level so different from our style of migration. In both cases, there is a transmission of a task, of knowledge, and interaction preferences. However this ducks the issue of agent identity and its realisation. We see maintaining the identity of an agent for the user as a basis for a long-term companion relationship, and argue that this requires more than presenting a similar face and voice (subject to the constraints of the embodiment). Identity involves consistent patterns of behaviour and affect over time, interpretable as personality, and a long-term shared frame of reference that is implemented in our level 3 agent mind as a human-like memory model [16]. The parametrisation of level 3 already referred to allows the dynamic affective state of the agent to be migrated. Work is also being carried out in migrating the agent episodic memory, which given its use of spreading activation across a network of episodes is far from a flat database from which items can be taken and reinserted without altering others in the process. In the future we are planning some long term user evaluation of our companion systems to gain further insight into the users' perception of a companion's identity and its retention after the migration process.

There are both implementational and interactional arguments against an agent with multiple embodiments. Maintaining the consistency of a memory of the type just mentioned would be very challenging. Interactionally, imagine

a second research group in the same building with their own companion agent with a distinct personality adapted to that group's preferences. What if on other days this second companion might want to inhabit the reception screen. In order for this to work the screen would need to be vacated if it is not used. One could again counter this argument with an alternative design in which both research groups are actually dealing with the same companion, an omnipresent AI that can interact with both groups simultaneously and in those interactions behave differently for both groups adapting to their preferences. This is reminiscent of many AIs of buildings or spaceships, etc. encountered in Science Fiction but we think realistically this approach has limits in terms of the size such an omnipresent AI could grow to. More than anything else we use migration because it is transparent and easily understood by users and helps them to think of the companion as a social entity rather than just a computer program.

The CMION architecture has been and is being successfully applied in the construction of several migrating social companions in the LIREC project. It is a basis for much longer-term experiments than have typically been carried out with embodied agents. Through this direct link to a wide variety of applications we could establish an iterative development loop where feedback from companion development feeds directly into improvements of CMION.
There are many directions into which this work can be taken further. One could for example add authorization mechanisms that verify whether a companion is allowed to migrate into a certain embodiment. For companions that often move between a number of embodiments one could add a caching mechanism to reduce the amount of data required to transmit. We already commented on the parallels between CMION and the SAIBA framework, but making the architecture fully SAIBA compliant could be a further worthwhile endeavour. By releasing CMION as open source we provide a basis for further research into migrating companions and hope that others will find in it a modular and useful framework to advance the state of migrating companions. The code and documentation for CMION can be found at *http://trac.lirec.org/wiki/SoftwareReleases*.

Acknowledgements

This work was partially supported by the European Commission (EC) and is currently funded by the EU FP7 ICT-215554 project LIREC (Living with Robots and Interactive Companions). The authors are solely responsible for the content of this publication. It does not represent the opinion of the EC, and the EC is not responsible for any use that might be made of data appearing therein.

References

[1] Mindmakers - project psyclone. http://www.mindmakers.org/projects/Psyclone
[2] Albus, J.S., Lumia, R., Fiala, J., Wavering, A.: NASA/NBS standard reference model for telerobot control system architecture (Nasrem). Technical Report 1235, Natl. Inst. Standards and Technology, Gaithersburg, Md. (1989)

[3] Aylett, R., Luck, M.: Applying artificial intelligence to virtual reality: Intelligent virtual environments. Applied Artificial Intelligence 14, 3—32 (2000)

[4] Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhjálmsson, H., Yan, H.: Embodiment in conversational interfaces: Rea. Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit p. 520–527 (1999), ACM ID: 303150

[5] Cassell, J., Vilhjálmsson, H.H., Bickmore, T.: BEAT: the behavior expression animation toolkit. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. p. 477–486. SIGGRAPH '01 (2001), ACM ID: 383315

[6] Cavazza, M., de la Camara, R.S., Turunen, M.: How was your day?: a companion ECA. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1. p. 1629–1630. AAMAS '10, International Foundation for Autonomous Agents and Multiagent Systems, Toronto, Canada (2010), ACM ID: 1838515

[7] Collett, T.H.J., Macdonald, B.A.: Player 2.0: Toward a practical robot programming framework. In: Proc. of the Australasian Conference on Robotics and Automation (ACRA) (2005)

[8] De Carolis, B., Pelachaud, C., Poggi, I., de Rosis, F.: Behavior planning for a reflexive agent. In: Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2. p. 1059–1064 (2001), ACM ID: 1642236

[9] Du Casse, K., Koay, K.L., Ho, W.C., Dautenhahn, K.: Reducing the cost of robotics software: SAMGAR, a generic modular robotic software communication architecture. In: Int. Conf. on Advanced Robotics 2009, ICAR 2009. pp. 1–6. IEEE (2009)

[10] Fitzpatrick, P., Metta, G., Natale, L.: Towards long-lived robot genes. Robot. Auton. Syst. 56(1), 29–45 (2008)

[11] Fong, T., Nourbakhsh, I., Dautenhahn, K.: A survey of socially interactive robots: Concepts, design, and applications. Robotics and Autonomous Systems 42(3-4), 142–166 (2002)

[12] Fritsch, J., Kleinehagenbrock, M., Haasch, A., Wrede, S., Sagerer, G.: A flexible infrastructure for the development of a robot companion with extensible HRI-Capabilities. In: Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. pp. 3408–3414 (2005)

[13] Gat, E.: On Three-Layer architectures. Artificial Intelligence and Mobile Robots (1998)

[14] Imai, M., Ono, T., Etani, T.: Agent migration: communications between a human and robot. In: IEEE International Conference on System, Man, and Cybernetics (SMC'99). vol. 4, pp. 1044–1048 vol.4 (1999)

[15] Kopp, S., Krenn, B., Marsella, S., Marshall, A.N., Pelachaud, C., Pirker, H., Thórisson, K.R., Vilhjalmsson, H.: Towards a common framework for multimodal generation: The behavior markup language. In: Intelligent Virtual Agents (IVA) (2006)

[16] Lim, M.Y., Aylett, R., Ho, W.C., Enz, S., Vargas, P.: A Socially-Aware memory for companion agents. Proceedings of the 9th International Conference on Intelligent Virtual Agents p. 20–26 (2009), ACM ID: 1612564

[17] Magnenat-Thalmann, N., Kasap, Z., Moussa, M.B.: Communicating with a virtual human or a skin-based robot head. In: ACM SIGGRAPH ASIA 2008 courses. p. 55:1–55:7. SIGGRAPH Asia '08, ACM, Singapore (2008), ACM ID: 1508099

[18] Ogawa, K., Ono, T.: Ubiquitous cognition: mobile environment achieved by migratable agent. In: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services. p. 337–338 (2005), ACM ID: 1085854

[19] O'Hare, G.M.P., Duffy, B.R., Schön, B., Martin, A.N., Bradley, J.F.: Agent chameleons: Virtual agents real intelligence. In: Proceedings of the fourth international working conference on Intelligent Virtual Agents (IVA 2003). pp. 218—225 (2003)

[20] Syrdal, D.S., Koay, K.L., Walters, M.L., Dautenhahn, K.: The boy-robot should bark!–Children's impressions of agent migration into diverse embodiments. In: Proceedings: New Frontiers of Human-Robot Interaction, a symposium at AISB (2009)

[21] Thiebaux, M., Marsella, S., Marshall, A.N., Kallmann, M.: Smartbody: Behavior realization for embodied conversational agents. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1. p. 151–158 (2008)

[22] Vala, M., Raimundo, G., Sequeira, P., Cuba, P., Prada, R., Martinho, C., Paiva, A.: ION framework — a simulation environment for worlds with virtual agents. In: Proceedings of the 9th International Conference on Intelligent Virtual Agents. pp. 418–424. Springer-Verlag, Amsterdam, The Netherlands (2009)