ProCRob Architecture for Personalized Social Robotics

Extended Abstract

Pouyan Ziafati*
SnT, University of Luxembourg
ziafati@luxai.eu

Francisco Lera
CSC, University of Luxembourg
francisco.lera@uni.lu

Andreia Pinto Costa
INSIDE, University of Luxembourg
andreia.pintocosta@uni.lu

Aida Nazarikhorram*
SnT, University of Luxembourg
nazari@luxai.eu

Leon Van Der Torre
CSC, University of Luxembourg
leon.vandertorre@uni.lu

ABSTRACT

For robot tutors, autonomy and personalizations are important factors in order to engage users as well as to personalize the content and interaction according to the needs of individuals. This paper presents the Programming Cognitive Robot (ProCRob) software architecture to target personalized social robotics in two complementary ways. ProCRob supports the development and personalization of social robot applications by teachers and therapists without computer programming background. It also supports the development of autonomous robots which can adapt according to the human-robot interaction context. ProCRob is based on our previous research on autonomous robotics and has been developed since 2015 by a multi-disciplinary team of researchers from the fields of AI, Robotics and Psychology as well as artists and designers at the University of Luxembourg. ProCRob is currently being used and further developed for therapy of children with autism, and for encouraging rehabilitation activities in patients with post-stroke. This paper presents a summary of ProCRob and its application in autism.

KEYWORDS

Social Robotics, Inclusive Robotics, Robot Software Architectures, Agent Programming Languages, Visual Programming Languages, Robot Tutors

ACM Reference format:
Pouyan Ziafati*, Francisco Lera, Andreia Pinto Costa, Aida Nazarikhorram, and Leon Van Der Torre. 2017. ProCRob Architecture for Personalized Social Robotics . In *Proceedings of R4L 2017, Viena, Austria, March 2017 (HRI'17),* 6 pages.
DOI: 10.475/123_4

*Pouyan Ziafati and Aida Nazarikhorram are founders of LuxAI S.A., a social robotic spinoff from the University of Luxembourg (www.luxai.eu). In order to avoid any conflict of interest, they do not participate in parts of this research related to scripting, conducting, testing and reporting robot applications and experiments for children with autism.

1 INTRODUCTION

There has been an increasing level of attention in educational robotics for the past 25 years [23], [15]. Most previous research use robots as educational tools [16], [4]. However in some recent research robots are proposed as tutors in classrooms [14], [10], [22].

The teaching process is defined by four elements [2]: context setting, class preparation, class delivery, and continuous improvement. Developing effective robot tutors requires the incorporation of the related know-how of professionals, perhaps most importantly the teachers, in the process. Such professionals often do not have robotics and computer programming background. Consequently, a crucial factor to advance in the field is to empower teaching professionals to develop, adapt, personalize and control robot tutor applications according to their desires in context setting, class preparation, class delivery and to improve the setting over time. However the application development for most robotic systems today is only accessible to computer programming experts. Such a system is a back-box in the eyes of teaching professionals.

In addition to user-friendliness for non computer experts, autonomy of robots is another important factor to build effective robot tutors. A robot tutor has to maintain students attention, concentration and enjoyment along the session. Autonomy and adaptation of the behaviors of a robot tutor during the interaction can be useful for maintaining the engagement and enhancing the motivation and performance of students [11, 29]. Such behaviors include verbal and non-verbal acts, for instance, addressing students with their first names and showing different facial emotions and body gestures [11].

This paper presents ProCRob, a multi-disciplinary work in progress to support the development of autonomous and personalized social robots and in particular robot tutors for children with special educational needs. The ProCRob software architecture is presented in Section 2. ProCRob offers RobAPL which builds upon the state of the art autonomous agent research to support the development of autonomous robots. RobAPL was designed in our previous research. This paper reports its ongoing implementation as well as a visual language which we have developed on top of RobAPL to support the development and personalization of social robot applications by non computer experts. After an overview of the ProCRob architecture in Section 2, Section 3 describes the application of ProCRob in an ongoing study on emotion regulation for children with autism spectrum disorder. In this study ProCRob is used by therapists without programming background to develop and run applications for the QT social robot. Finally, conclusion and future work is presented in section 4.

2 PROCROB ARCHITECTURE

ProCRob is a software architecture to support the rapid prototyping and development of autonomous social robots. It also aims to make social robotics inclusive by allowing field experts such as teachers and therapists without computer programming background to develop and use social robot applications for their work.

The ProCRob architecture comprises three layers:

- Functional layer: is composed of a set of components implementing robot's action and perception capabilities and processing algorithms. These includes recording and playing robot body gestures, speech recognition, face recognition, object recognition and text to speech. This layer is implemented using ROS [21] and YARP [18], well-known robotic frameworks which provide a large number of open-source robot software components and support the implementation of communication mechanisms among different components.
- Robot Agent Programming Language (RobAPL): supports the development of autonomous robots at a high-level of abstraction in terms of a robot's goals, beliefs and plans. This layer implements the execution system of RobAPL, a language which we previously designed to this end [32].
- RobAPL interface: supports the development and personalization of social robot applications by providing a user-friendly visual programming language on top of RobAPL for teachers and therapists who do not have any computer programming background.

The following presents a brief overview of RobAPL and its visual interface.

2.1 Robot Agent Programming Language

A line of AI research has developed a family of Agent Programming Languages [3] to support the development of autonomous agents. Inspired by the Belief-Desire-Intention model of human practical reasoning [9], these languages include components such as beliefs, goals, plans, and plan generating rules. Each plan generating rule specifies a plan to reach a goal if executed in a specific belief state. The execution of an agent's program implemented by such a language is a cyclic process. Sensory information is processed, beliefs and goals of the agent are updated, applicable plan generating rules are applied and generated plans are executed. For example, a piece of sensory information can be, "the face of Aida was recognized". By receiving this information, agent can update its beliefs, "Aida is now standing in front of me". It can also have a rule to react to this event and execute a plan to greet her, " Hello Aida" and to start a dialogue system to interact with her.

While agent programming languages provide a suitable level of abstraction and programming support for implementing autonomous agents, they provide a limited support for sensory information processing and plan execution control, necessary capabilities to implement control mechanics of robots [33]. To provide a better support for sensory information processing, we developed the Retalis language for information engineering in autonomous robot software [34]. It is a logic-based language for processing, aggregating, correlating, storing, querying and reasoning on flows of robot's sensory data. Retalis supports the development of a social robot to process and reason about its sensory data in semi real-time to understand the situations of its environment, to record its knowledge of the environment in memory and to query such knowledge on-demand.

To provide a better support for plan execution control, we proposed the Robot Agent Programming Language (RobAPL) [32], being the main focus of this paper. RobAPL adapts and extends PLEXIL [30], an expressive and well-defined robotic plan execution language, for plan representation and execution in BDI-based agent programming languages. PLEXIL offers a simple structure for plan representation, a hierarchy of nodes with few syntactic constructs to monitor and govern their execution. However it is one of the most expressive plan execution languages unifying many of the existing ones. PLEXIL supports hierarchical task decomposition and controlling and monitoring the plan execution at different levels of plan hierarchy. It also supports conditional contingencies, loops, temporal constraints and floating contingencies (i.e. event driven task execution) in the task tree decomposition. PLEXIL has formal semantics with modular operational semantics, making it easy to modify and extend the language.

RobAPL extends PLEXIL by introducing basic execution nodes for querying and manipulating the agent's beliefs and goals in the BDI architecture and presenting an operational semantics for PLEXIL-like plan execution in BDI execution cycle. It also extends PLEXIL to support pausing, resuming and pre-empting plans, performing clean-up and wind-down activities when pausing, resuming, pre-empting or aborting plans, and coordinating the parallel execution of plans over shared resources. By adding an extended support for plan representation and execution to the BDI architecture, RobAPL facilitates and simplifies the development of autonomous robots by providing an architecture and programming constructs to specify a robot's behavior in a compact way at a high level of abstraction. Such support is necessary in order to build social robots which can present autonomy and goal-oriented behavior with a high level of variability in the interaction to increase the engagement of their users. An example is a BDI-based conversation management system where a relatively very small program has been shown which can represent a coherent, goal-oriented dialogue system with 2 million potential conversations [31].

RobAPL provides an extensive proposal for an agent-based language for robotic applications. While the language is still under implementation, we have developed a stable version of it, currently being used in social robot experiments outlined in the next section. This version includes support for representation of robot plans consisting of sequential and parallel actions of which the execution is governed by their orders as well as external events. An external event can be for instance the recognition of a face. This allows to represent a story application for our social robot described in the next section. A story is a sequence of plays where each play includes a text to be said by the robot, a gesture to be played by the robot and an animation to be shown by the robot. The execution of sequential plays of a story is synchronized such that the next play is executed only after and as soon as all three actions of the previous play were finished (i.e. text-to-speech, play gesture and show animation). The language also allows for instance to start a play conditioned on the occurrence of an event. For instance, we have developed a card game where the robot asks the user to choose and show the picture of a specific animal. Then depending on whether the robot sees the right picture or not, it execute different plays.

Our implementation of RobAPL is in Prolog, a well-known logic programming language. Prolog was chosen as it is the language of choice for knowledge representation and reasoning in most BDI-based agent programming languages. Prolog supports to encode knowledge using a set of rules. For instance we have defined a taxonomy of things which robot can recognize, classified into categories of human and animal. This provides robot with the knowledge of which picture is a human and which one is an animal. For instance, the start condition of a play can be, "if a picture of an animal was recognized". Then if the robot recognizes the picture of a cat, it can infer from its knowledge that a cat is an animal, a picture of an animal has been recognized and proceed with the execution of the play. In addition, our Prolog implementation has a small footprint which has proven to be great for rapid prototyping and development.

2.2 RobAPL Interface

In order to enable teachers and therapists to develop and personalize social robot applications, we have developed a visual programming interface on top of RobAPL. The interface is built using the Blockly visual programming language [8] and is offered to users as an Android application for tablets and smart phones. Blockly is similar to the well-known Scratch visual programming language developed by the MIT Media Lab. In Scratch, a computer program is implemented using a set of visual blocks. Scratch has a large community and has been proven user-friendly for computer non-experts to learn and practice programming.

Blockly is a language developed by Google for building Scratch like languages by supporting the creation of custom blocks. Empowered by Blockly, our Android app provides a set of blocks using which non-computer experts can build complex robot applications. Blockly-based programs are then translated into RobAPL source codes to be executed on the robot.

An example of a custom block is a play-block shown in Figure 1 which has three fields.

- Emotion-field: is a container in which a emotion-block is placed. Our Android app provides a library of pre-defined emotion-blocks, each representing an emotion such as sad or happy to be shown by the robot's face.
- Text-field: is filled with a text message to be told by the robot.
- Gesture-field: is a container in which a gesture-block is placed. Our Android app provides a library of pre-defined gesture-blocks, each representing a gesture such as wave-hand to be performed by the robot.

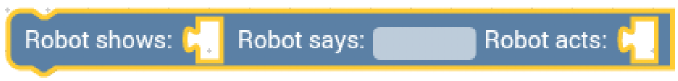

Figure 1: Play-block in the RobAPL Interface

There is also another type of play-block which includes an audio-field instead of a text-field. Audio-field is a container in which an audio-block is placed, representing an audio file to be played by the robot. The android app allows users to create custom emotion, gesture and audio blocks. For instance, new audio and video files can be imported from the Android device storage or DropBox and a simulated environment is provided to create new robot gestures. A story application is built using a sequence of the two types of play-blocks, example of which is shown in Figure 3. When this program is executed, the Andoird app presents the user with a control panel with 9 flags. When the user presses Flag_1, the story is played.

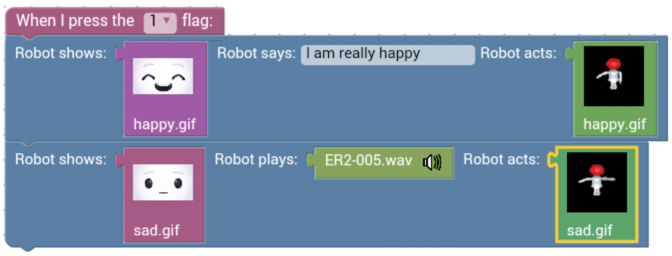

Figure 2: An example of a story in the RobAPL Interface

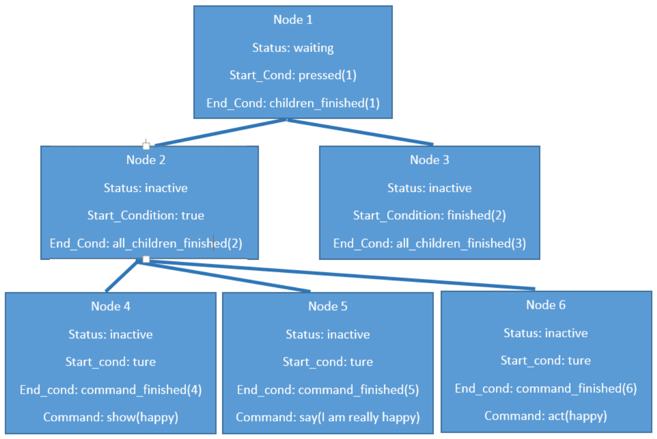

Figure 3: Corresponding RobAPL plan of Figure 2

Figure 3 shows the corresponding RobAPL plan for the story application presented in Figure 2. In this figure, Node 1 corresponds to the purple block in Figure 2. This node is in the waiting status in which the start condition of the node, pressed(1), is monitored. When Flag_1 is pressed by the user, this node transits to the executing state and then its children Node 2 and Node 3, the blue play-blocks, transit from the inactive state to the waiting state. As the start condition of Node 2 is true, it transits to the executing state. Consequently, its children (Node 4, 5 and 6) transit to the waiting state. As the start condition for all these three nodes are true, they transit to the executing state and their associated commands are executed in parallel. In this case, the robot shows the happy face, says " I am really happy" and performs the happy gesture. Each of these three nodes transit to the finished state when the execution of their associated command is finished. The end condition of Node 2 is when all of its three children transit to the finished state. Therefore when the three commands are finished, Node 2 transits to the finished state, making the start condition of Node 3 true and the robot starts playing the second play-block. Similar to Node 2, Node 3 has also three children, Nodes 7,8 and 9, which are not shown in the picture as they are similar to the children of Node 2 and are executed similarly. When these nodes are executed, the robot shows the sad face, plays the audio file ER2-005 and performs the sad gesture. After all these commands are finished, Node 3 transit to the finished state, making the end condition of Node 1 true. Therefore Node 1 transit to the finished state and the execution of the program is finished.

Various other custom blocks have been developed supporting the implementation of more advanced functionalities. For instance, the card game application described in the previous section is developed using the block presented in Figure 4. This block represents a choice among three branches of execution. When the robot's execution reaches this block, an external event determines what the robot will do. If the robot sees the picture of a cat, the robot says, "It is a cat." If what the robot sees is not a cat, the robot says, "This is not a cat." Finally, if the robot does not see anything for 5 seconds, it says, "I am waiting for an answer." This block also presents an example of how our Android app facilitates supervised autonomy of the robot. In this example, if the end user shows the picture of a cat to the robot, the robot should say, "it is a cat". It might be the case however that due to the lighting condition, the robot cannot recognize the picture of the cat. In this case, the first choice can be enforced by pressing Flag_1 on the Android app control panel by a user supervising the robot.

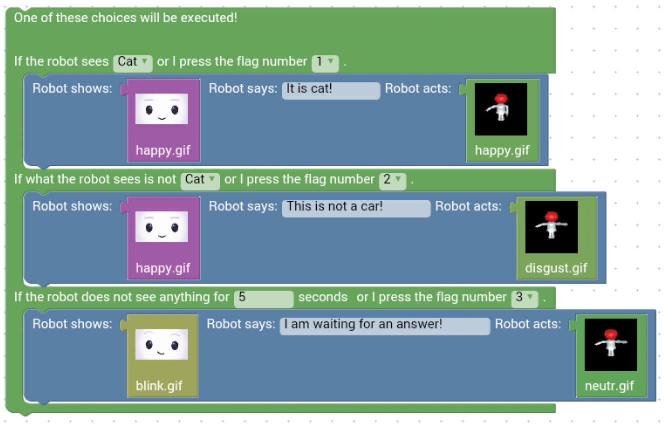
One of these choices will be executed!
If the robot sees Cat or I press the flag number 1 .
Robot shows: happy.gif Robot says: It is cat! Robot acts: happy.gif
If what the robot sees is not Cat or I press the flag number 2 .
Robot shows: happy.gif Robot says: This is not a car! Robot acts: disgust.gif
If the robot does not see anything for 5 seconds or I press the flag number 3 .
Robot shows: blink.gif Robot says: I am waiting for an answer! Robot acts: neutr.gif

Figure 4: An example of a choice block

Our android app has been proven user-friendly in practice both for developing applications and running the robot. Currently medical doctors and Psychologists of our team, without any computer programming background, are independently in charge of programming applications for therapy of children with autism, described in the next section. They are also able to run and control the execution of robot's programs without support from the engineers.

The RobAPL language has also proven developer-friendly in our practice so far in terms of rapid prototyping and development of custom blocks. This provides a great advantage for our multidisciplinary work to quickly address the needs of therapists in application development using the visual programming interface to meet the deadlines. For instance, in the script of an application made by a therapists, a choice block with nine possible branches of execution was required. Thanks to the developer friendliness of RobAPL and its interface, the new block was developed and integrated in the language within a few hours and the team met the next day deadline of an experiment.

3 PROCROB APPLICATIONS

This section presents a work in progress to use the ProCRob architecture in an experiment for therapy of children with autism. First, we will briefly describe the robot platform on which we run ProCRob.

3.1 Robot Hardware

For our research and development purposes we use prototypes of QT, a commercial social robot from LuxAI[1]. QT is a humanoid robot with an expressive social appearance. It has a screen as its face, allowing the presentation of facial emotions using animated characters. Figure 5 presents pictures of some of the QT's animated characters. The robot has 14 degrees of freedom to present upper-body gestures. Eight degrees of freedom are motor-controlled, two in each shoulder, one in each arm plus pitch and yaw movements of the head. The other four, one in each wrist and one in each hand, are manually configured.

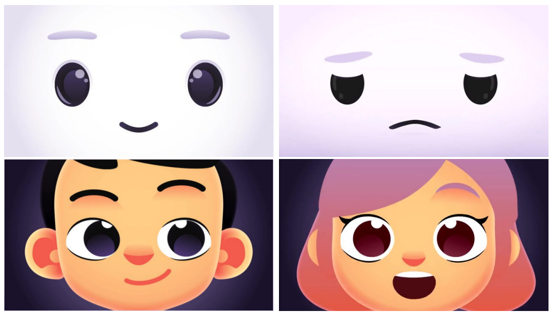

Figure 5: QT's animated characters and emotions

Figure 6 presents QT embodiment, it has a close-range 3D camera (20 to 150 cm) mounted on its forehead and is provided with a microphone array. QT is powered with an Intel NUC processor and Ubuntu 16.04 Lts, supporting native compilation of programs in Ubuntu, and is provided with a native ROS interface. Communication with QT is established by wifi.

[1]www.luxai.eu

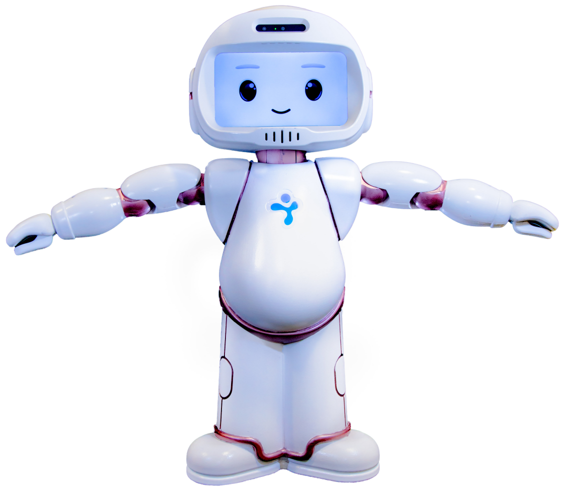

Figure 6: QR Robot

3.2 Therapy of Children with Autism

Autism Spectrum Disorder (ASD) is a neurodevelopmental disorder characterized by deficits in social communication, social interaction, and by restricted and repetitive patterns of behaviors and interests [1]. Many problems in ASD have been linked to difficulties in emotional reactivity and emotion regulation [26], [17]. Moreover, emotional awareness has been found to be linked to some emotional disturbances in individuals with ASD [26], [5]. Furthermore, children's capacity to produce and recognize facial and vocal expressions of emotions is a fundamental pre-requisite for effective emotional ability [12]. Therefore, improving emotional ability in children with ASD can be of paramount relevance for their development.

Robots are promising tools for children with social interaction difficulties such as children with autism spectrum disorder (ASD). It has been shown that children with ASD prefer interactions with robots over humans [25], [6], [19]. Robots provide novel sensor stimuli which can stimulate childrenfis interest and attention. Compared to humans, robots are more predictable, less confusing, less complex, and less distressing for children with ASD[28]. Furthermore, some studies have shown that outcomes can be better for robot-based therapies than for human-based therapies [25], [27], [24]. Due to these characteristics, socially assistive robots could be ideal tutors for emotional trainings with children with ASD.

Therapists could also potentially benefit from trainings using socially assistive robots with children with ASD. A training which is administered by a robot can reduce the therapistfis burden of memorizing the contents in a standardized way and of administrating the same training to several children while following strict protocols. There has been some research on using robots to increase children's emotional expression [7], [13], [20]. However to the best of our knowledge no studies have so far examined the effectiveness of using robots to train emotional abilities in children with ASD.

We are currently conducting an experiment with QT robot to improve emotional ability in five domains of emotional ability in children with ASD: facial and vocal production of emotions, facial and vocal recognition of emotions, emotional reactivity, emotional awareness, and emotion regulation. The experiment is at pilot stage, details of which are described in another paper, “socially assistive robots for teaching emotional abilities to children with autism spectrum disorder”, presented in the “Growing-Up Hand in Hand with Robots” workshop of HRI 2017.

In this experiment, the QT robot acts as a tutor and fully administers the training. All instructions and interactions with the child are provided solely by QT. However, interactions are controlled by a therapist using the robAPL android app. Additionally, for some exercises, printed images are placed in front of the child by the researcher. Each session starts with a short introduction where objectives and concepts are explained to the child in an age appropriate language. Then QT proposes several games where different aspects of emotional ability are trained. Each session finishes with a summary of what was taught during the session and instructions for home practice are given.

The QT robot in this experiment is fully programmed and controlled using the RobAPL visual interface by non-computer experts. The visual interface is used to develop custom social robot applications according to the specific aim of each training session, in total six training sessions with the robot for each child. Furthermore, sessions are customized for each child. For instance, QT is greeting each child by his/her name. Training sessions are conducted in a Wizard of Oz setup due to the complexity of the child-robot interaction where in one of the games for instance QT has to recognize the child's facial expressions and provide appropriate feedback. In order to have a smooth interaction and avoid potential mistakes from computer vision algorithms, the robot's feedback is controlled by the therapist. The visual interface provides a good support for this by presenting a control panel to the therapist using which he/she can control the robot's course of execution at runtime.

While not used in this experiment, we have also developed various demo applications to showcase the robot's autonomy, for instance, the card game described in Section 2.1. Our aim is to develop autonomous social robots of which the educational content is provided and customized by field-experts, teachers and therapists, but can autonomously interact with end users and adapts their behavior according to the interaction context. Such a robot for instance would be able to present a goal oriented behavior to teach a specific skill to the child while analyzing the the child's level of attention and take appropriate courses of action to keep the child engaged in the interaction.

4 CONCLUSION AND FUTURE WORK

This paper presents a multi-disciplinary work in-progress on the ProCRob software architecture to support the development of autonomous and personalized social robots, in particular, robot tutors for children with special educational needs. We report on recent implementation of the architecture including a visual programming interface which enables teachers and therapists to develop and use advanced social robot applications for their work.

ProCRob is currently used to program and control the QT social robot by therapists in an on-going study for emotion regulation in children with autism spectrum disorder. An overview of the study is presented and advantages of using ProCRob in this study is briefly highlighted.

Future work includes extending ProCRob to support the implementation of goal-oriented proactive social robot behaviors. Also in addition to the on-going study presented in this paper, ProCRob is currently being used to develop applications for the QT robot to encourage post-stroke rehabilitation activities.

ACKNOWLEDGMENTS

This research is funded by Luxembourg National Research Fund (FNR) Proof-of-Concept program.

REFERENCES

[1] American Psychiatric Association. 2000. Diagnostic and statistical manual of mental disorders. *text rev.)* (2000).

[2] Ellen R Auster and Krista K Wylie. 2006. Creating active learning in the classroom: A systematic approach. *Journal of Management Education* 30, 2 (2006), 333–353.

[3] Rafael H Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni. 2009. *Multi-Agent Programming:: Languages, Tools and Applications*. Springer Science & Business Media.

[4] Chih-Wei Chang, Jih-Hsien Lee, Po-Yao Chao, Chin-Yeh Wang, Gwo-Dong Chen, and others. 2010. Exploring the possibility of using humanoid robots as instructional tools for teaching a second language in primary school. *Educational Technology & Society* 13, 2 (2010), 13–24.

[5] Richard Cook, Rebecca Brewer, Punit Shah, and Geoffrey Bird. 2013. Alexithymia, not autism, predicts poor recognition of emotional facial expressions. *Psychological Science* 24, 5 (2013), 723–732.

[6] Kerstin Dautenhahn and Iain Werry. 2004. Towards interactive robots in autism therapy: Background, motivation and challenges. *Pragmatics & Cognition* 12, 1 (2004), 1–35.

[7] Audrey Duquette, François Michaud, and Henri Mercier. 2008. Exploring the use of a mobile robot as an imitation agent with children with low-functioning autism. *Autonomous Robots* 24, 2 (2008), 147–157.

[8] Neil Fraser. 2014. Google blockly-a visual programming editor. *URL: http://code. google. com/p/blockly. Accessed Aug* (2014).

[9] Michael Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Michael Wooldridge. 1998. The belief-desire-intention model of agency. In *International Workshop on Agent Theories, Architectures, and Languages*. Springer, 1–10.

[10] Goren Gordon, Samuel Spaulding, Jacqueline Kory Westlund, Jin Joo Lee, Luke Plummer, Marayna Martinez, Madhurima Das, and Cynthia Breazeal. 2016. Affective personalization of a social robot tutor for children's second language skills. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 3951–3957.

[11] Joan Gorham. 1988. The relationship between verbal teacher immediacy behaviors and student learning. *Communication education* 37, 1 (1988), 40–53.

[12] Madeline B Harms, Alex Martin, and Gregory L Wallace. 2010. Facial emotion recognition in autism spectrum disorders: a review of behavioral and neuroimaging studies. *Neuropsychology review* 20, 3 (2010), 290–322.

[13] Hideki Kozima, Cocoro Nakagawa, and Yuriko Yasuda. 2007. Children–robot interaction: a pilot study in autism therapy. *Progress in Brain Research* 164 (2007), 385–400.

[14] Daniel Leyzberg, Samuel Spaulding, and Brian Scassellati. 2014. Personalizing robot tutors to individuals' learning differences. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*. ACM, 423–430.

[15] Fred Garth Martin. 1994. *Circuits to control: Learning engineering by designing LEGO robots*. Ph.D. Dissertation. Citeseer.

[16] David Keating' Martyn Cooper and Kerstin Dautenhahnfi William Harwin. 1999. Robots in the classroom-tools for accessible education. *Assistive technology on the threshold of the new millennium* 6 (1999), 448.

[17] Carla A Mazefsky and Susan W White. 2014. Emotion regulation: Concepts & practice in autism spectrum disorder. *Child and adolescent psychiatric clinics of North America* 23, 1 (2014), 15–24.

[18] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. 2006. Yarp: Yet another robot platform. *International Journal of Advanced Robotic Systems* 3, 1 (2006), 8.

[19] Andrea C Pierno, Morena Mari, Dean Lusher, and Umberto Castiello. 2008. Robotic movement elicits visuomotor priming in children with autism. *Neuropsychologia* 46, 2 (2008), 448–454.

[20] Giovanni Pioggia, Roberta Igliozzi, Maria Luisa Sica, Marcello Ferro, Filippo Muratori, Arti Ahluwalia, and Danilo De Rossi. 2008. Exploring emotional and imitational android-based interactions in autistic spectrum disorders. *Journal of CyberTherapy & Rehabilitation* 1, 1 (2008), 49–61.

[21] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, Vol. 3. Kobe, 5.

[22] Aditi Ramachandran and Brian Scassellati. 2015. Developing adaptive social robot tutors for children. In *2015 AAAI Fall Symposium Series*.

[23] Mitchel Resnick. 1993. Behavior construction kits. *Commun. ACM* 36, 7 (1993), 64–71.

[24] Daniel J Ricks and Mark B Colton. 2010. Trends and considerations in robot-assisted autism therapy. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 4354–4359.

[25] Ben Robins, Kerstin Dautenhahn, and Janek Dubowski. 2006. Does appearance matter in the interaction of children with autism with a humanoid robot? *Interaction studies* 7, 3 (2006), 509–542.

[26] Andrea C Samson, Oswald Huber, and James J Gross. 2012. Emotion regulation in Asperger's syndrome and high-functioning autism. *Emotion* 12, 4 (2012), 659.

[27] Brian Scassellati. 2007. How social robots will help us to diagnose, treat, and understand autism. In *Robotics research*. Springer, 552–563.

[28] Brian Scassellati, Henny Admoni, and Maja Matarić. 2012. Robots for use in autism research. *Annual review of biomedical engineering* 14 (2012), 275–294.

[29] David J Shernoff, Mihaly Csikszentmihalyi, Barbara Shneider, and Elisa Steele Shernoff. 2003. Student engagement in high school classrooms from the perspective of flow theory. *School Psychology Quarterly* 18, 2 (2003), 158.

[30] Vandi Verma, Tara Estlin, Ari Jónsson, Corina Pasareanu, Reid Simmons, and Kam Tso. 2005. Plan execution interchange language (PLEXIL) for executable plans and command sequences. In *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

[31] Wilson Wong, Lawrence Cavedon, John Thangarajah, and Lin Padgham. 2012. Flexible conversation management using a bdi agent approach. In *International Conference on Intelligent Virtual Agents*. Springer, 464–470.

[32] Pouyan Ziafati. 2014. Plexil-Like Plan Execution Control in Agent Programming. (2014). http://www.aaai.org/ocs/index.php/WS/AAAIW14/paper/view/8810

[33] Pouyan Ziafati, Mehdi Dastani, John-Jules Meyer, and Leendert van der Torre. 2012. Agent programming languages requirements for programming autonomous robots. In *International Workshop on Programming Multi-Agent Systems*. Springer Berlin Heidelberg, 35–53.

[34] Pouyan Ziafati, Mehdi Dastani, John-Jules Meyer, Leendert van der Torre, and Holger Voos. 2015. Retalis Language for Information Engineering in Autonomous Robot Software. *IfCoLog Journal of Logics and their Applications* 2, 2 (2015), 85.