Project No. 611391

DREAM
Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

Grant Agreement Type:	Collaborative Project
Grant Agreement Number:	611391

D6.3 Deliberative Subsystem

Due date: **1/4/2015**
Submission Date: **25/3/2015**

Start date of project: **01/04/2014** Duration: **54 months**

Organisation name of lead contractor for this deliverable: **Vrije Universiteit Brussel**

Responsible Person: **Bram Vanderborght** Revision: **2.2**

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including the Commission Service)	
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Executive Summary

Deliverable D6.3 defines the specification, design, implementation and validation of the Deliberative subsystem within the cognitive architecture in Work Package 6.

Specifically, this report presents the advances done in task T6.3 for the first year of the DREAM project. During this first year the cognitive architecture and the Deliberative subsystem have been designed. They are described within this report.

Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order).

Paul Baxter, Plymouth University
Tony Belpaeme, Plymouth University
Hoang-Long Cao, Vrije Universiteit Brussel
Albert de Beir, Vrije Universiteit Brussel
Pablo Gómez, Vrije Universiteit Brussel
Emmanuel Senft, Plymouth University
Greet Van de Perre, Vrije Universiteit Brussel
Bram Vanderborght, Vrije Universiteit Brussel

Revision History

Version 1.0 (P.G. 27-01-2015)
First draft.

Version 2.0 (P.G. 12-02-2015)
Some texts have been corrected.

Version 2.1 (P.B. 22-03-2015)
Addition of some appendices.

Version 2.2 (P.B. 25-03-2015)
Added information on preliminary implementation of deliberative subsystem and script manager components.

1 Overview of WP6 architecture

In DREAM we will move away from Wizard of Oz-controlled behaviour for the robot, which too often is the de facto mode of interaction in Robot Assisted Therapy [1]. Therefore work package WP6 aims to progress the theoretical and methodological understanding of how an embodied system can interact autonomously with young users in a learning task, specifically developed for atypically developing children. WP6 is concerned with the development of the robot behaviour subsystems to provide social robots with a behaviour underlying social interaction, which permits the robot to be used in RET in a supervised autonomous way. This involves both autonomous behaviour and behaviour created in supervised autonomy, whereby an operator requests certain interventions, which are then autonomously executed by the robot.

A general high level description of the robot control system is shown in figure 1 (also see annex 4.2). This basically describes how the autonomous controller is informed by three external sources: the child behaviour description, sensory information, current intervention script state, and input from a therapist (e.g. emergency stop, not shown in diagram). Combining these sources, the autonomous controller should trigger as an output the appropriate sequence of action primitives to be performed (as well as some feedback via the WoZ GUI), which then gets executed on the robot.

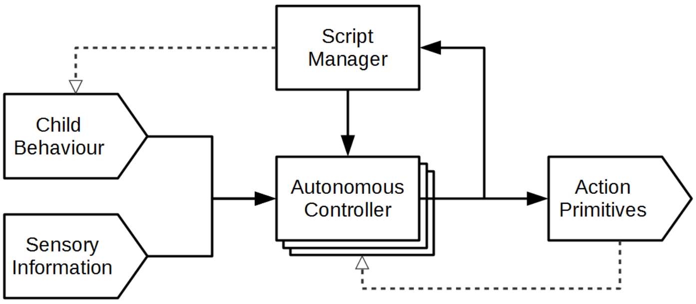

Figure 1: High level description of the robot control system. Child behaviour interpretation (WP5) and sensory information (WP4) provide the context for the autonomous action selection (as well as feedback from motor command execution), in combination with the particular intervention script being applied. The intervention script provides context for child behaviour interpretation.

The autonomous controller is composed of a number of subsystems, as described in the DoW: Reactive, Attention, Deliberative, Self-Monitor and Expression and Actuation. In the Reactive subsystem, sensory inputs are immediately acted upon with appropriate actuator outputs. The Attention subsystem determines the robot's focus of attention. In the Deliberative subsystem, the necessary interventions will be implemented in a general approach so it is not scenario-specific. The Self-Monitoring subsystem acts as an alarm system in two specifications. An internal one when the robot detects that it cannot act because of a technical limitation or an ethical issue. An external alarm is one where the therapist overrules the robot behaviour selection. Finally, the Expression and Actuation subsystem is responsible for generating believable human/animal-like smooth and natural motions and sounds that are platform independent. These subsystems interact, and must combine their suggested courses of actions to produce a coherent robot behaviour, in the context of constraints laid down by the therapist (for example, the script to be followed, types of behaviour not permissible for this particular child because of individual sensitivities, etc). As a result, we have formulated the following architecture describing how cognitive control informed by the therapy scripts is to be achieved (figure

2), see Annex 4.2 for further details.

A detailed description of the cognitive architecture will be provided in deliverable D6.1 at month 18. Next version of this document will be ready for month 24.

Within this report we describe the functionality of the Deliberative subsystem.

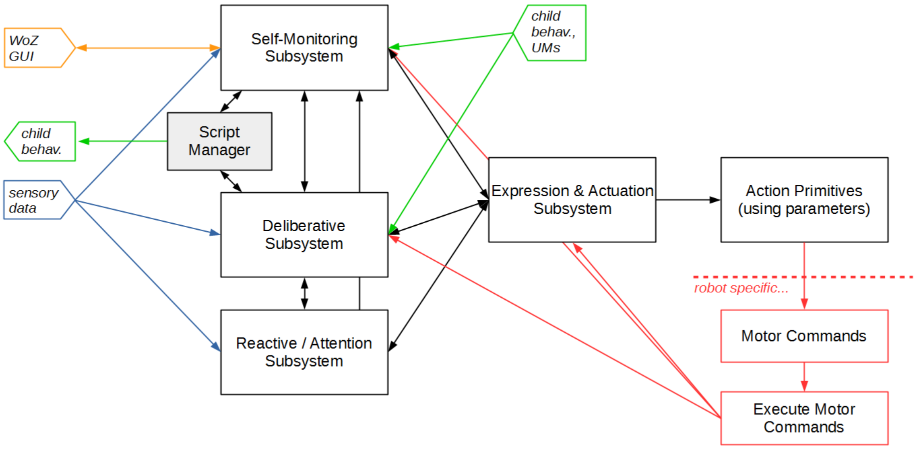

Figure 2: Description of the cognitive controller subsystems. The script manager is separate from, but tightly interacts with, the Deliberative subsystem to enable the robot control system to generate appropriate social/interaction behaviour even in the absence of an explicit interaction script. UMs: User Models.

2 The Deliberative subsystem

The main goal aimed by this subsystem is to make decisions on which behaviour has to be selected based on the requirements of the therapy; what the Attention subsystem is capturing from the surroundings; whether or not the child is motivated enough and how he or she is performing in each of the scenarios; and finally, the on-line feedback that the therapist could be providing through the WoZ GUI. Such behaviour will be sent to the Expression and Actuation subsystem.

2.1 Action Selection Mechanism

2.1.1 Context

The script provides a series of action that the robot should executes and that should be followed by a child specific reaction. However, the actual reaction from the young user can be different to the one expected. To be able to follow the script, the robot need to find a way, execute the appropriate behaviour, to obtain the desired action from the child. To do such a challenging task, a social Action Selection Mechanism is required. This social action selection mechanism has to fulfil multiple specifications: allowing the robot to detect a state where the execution of an action not planned in the script is required as well as finding the best action to select in order to obtain the expected action from the child and so being able continue the script. However, as we are working in the real world, the sensors are represented in a high-dimension and continuous space. This mean that designing by hand an efficient controller would probably not be feasible, there is no way to decide in advance what the best action to perform in each state is. Furthermore, as we are working with children, the environment is highly unpredictable and there is probably not a unique general solution for every child: the robot needs to be able to adapt to its different interaction partners. For these two reason, a classical static action selection mechanism is probably not enough, we have to provide the robot with the ability to learn or adapt a policy allowing it to select the best action in the unexpected states. The majority of the algorithms used for learning face a trade-off between exploitation and exploration, the agent needs first to explore its environment to be later able to select the best action to perform, and generally there is a part of randomness in the exploration. However, as we are in therapy scenario, every action needs to be adapted to the therapeutic goal, there is no place for randomness. We need to find an Action Selection Mechanism allowing the robot to become autonomous, to learn the best action to execute in a highly unpredictable, continuous and complex space without relying on random exploration.

2.1.2 State of the Art

Action selection has been explored by many research groups in the last decades, mainly in the case of survival robot, the robot is in a hostile environment and needs to find a way to stay alive, by keeping its battery above a certain level for example or multi agent systems [2] [3]. However, the case of social robot and especially robots enhanced therapy, only really simple autonomous robots have been used [4]. In classical robotics, many methods have emerged to tackle this action selection challenge. Ethology has been an important source of inspiration to design such controllers [5]. A well-documented approach is homeostasis control [3], in that case, the state of the robot is characterised by several variables representing for example energy, hunger and thirst and the robot needs to keep them between predefined boundaries. These variables are used to compute needs which generate drives. This drives are then arbitrated to select the appropriate action to fill the most important need. Many extensions of this model have been developed, for example using hormones [6] and adding an epigenetic adaptation [7]. In social robotics, this method has been used mainly to select an emotion to display, in her work, Breazel [8] used arousal and valence to represent the emotional state of the robot, and then select an appropriated emotion to display on the Kismet robot. However this method does not seem suitable for an adaptive control of action in HRI. Reinforcement learning is another possibility to allow a robot to learn a policy. Reinforcement learning is an algorithm for machine learning allowing an agent to learn a policy based on interaction with the environment. This method, based on trial and errors, uses a single variable describing the efficiency of the current behaviour, and the agent needs to find a way to maximise this value [9]. However this method is highly dependent on exploration which is generally done probabilistically by selecting a random action, and as explained earlier, we cannot use random exploration in our setup. In [10] Scheutz et al. propose method to select action based on selecting the action with the highest priority. They define priority as:

$$priority = urgency \cdot (affective\ value \cdot reward - cost) \tag{1}$$

With urgency a value symbolising how long the goal will be available, affective value, a number computing using the negative and positive affect of the action. However in this method, every different variables need to be defined at some point and this seems not suitable for our problem. Another approach is first model the environment, and then once the model is precise enough, use it to select the best action according to the description. A method which seems interesting in that direction is Partially Observable Markov Decision Process [11]. This assume that the environment is defined by a Markov Decision Process: there is a limited quantity of states, and every action taken from a state as a defined probability to lead to another state. In POMDP the only observe the model, but does not know the actual state, so it keeps a belief about the current state, and update it according to the information received from the environment. This method has been used successfully in HRI by Pineau et al. in [12]. In that case, a POMDP was used for the high level control for a robot designed to help elderly in nursing homes.

2.1.3 Proposed solutions

As shown previously, there is currently no algorithm in the literature allowing completing all the desiderata for our Action Selection Mechanism. We may use a learning algorithm, but in that case, we have to carefully guide the exploration to avoid random action. We need to find a way to provide some basic datapoints (link state-action) to the robot, but as a developer, we have no ideas of the best action to execute in an unexpected state, this knowledge belong to the therapist. We can take inspiration from the literature about Learning from Demonstration. The idea behind this method is to use a human or another robot show the robot how to perform the desired action. Generally, this technique is used to learn motor primitives, but few studies applied it to action selection [13]. In that case, the robot act autonomously as long as it is confident of its actions, but in an unexpected situation, it requires help from a supervisor to show it which action should be executed, and then learn from this to be able to reuse this knowledge to be more and more autonomous during the interaction. This method allows us to provide a supervised progressive autonomy to the robot. This method can be combined with other ones, it could be used to create or update a POMDP model, used in combination with an homeostatic model or to create new POSH

2.2 Planned Work

A central aspect of the cognitive controller is its ability to follow intervention scripts as defined by the clinicians for both diagnosis and therapy. These scripts describe the high-level desired behaviour of the robot[1], and the expected reactions and behaviours of the child, in a defined order.

The decision was made to separate the script manager from the Deliberative subsystem itself (figure 3). This decision was taken for a number of reasons. Firstly, it enables the cognitive control of the robot to be independent of the precise application domain - with the intention that the developments made would be more generally applicable within the field of social robotics, although the script-based behaviours remain a central part of the behaviour generation of the system. Secondly, it ensures that it would be possible to change the scripts in the future to alter their relative difficulty, by for example including further steps in the intervention, changing the type of intervention, or creating different activities, due to a modular design[2]. As a consequence of this, the Deliberative subsystem is now primarily focussed on action selection considerations, making use of a range of algorithms and methodologies as will be explored in the coming years. Thirdly, this division of the script manager from the Deliberative subsystem enables the system to generate coherent behaviour even if there is not a script active at a given moment. This could be useful for periods between the explicit intervention sessions for example, where the robot would then still be able to respond appropriately to environmental stimuli, if so desired by the therapists. These are consistent with the aims expressed within the WP6 Description of Work.

The script manager itself separates the logic necessary to manage progression through the script (by taking into account the available sensory feedback after actions for example) from the script itself.

---

[1] These predefined robot behaviours differ from the the low-level motor control of the robot, as these may be mixed with other aspects of behaviour not specified explicitly in the high-level intervention script; e.g. the addition of attention to unexpected events in the environment.

[2] As noted above, these high-level scripts do not necessarily completely define the behaviour of the robot, and are distinct from any predefined robot motor control sequences that may be used, such as waving or nodding.

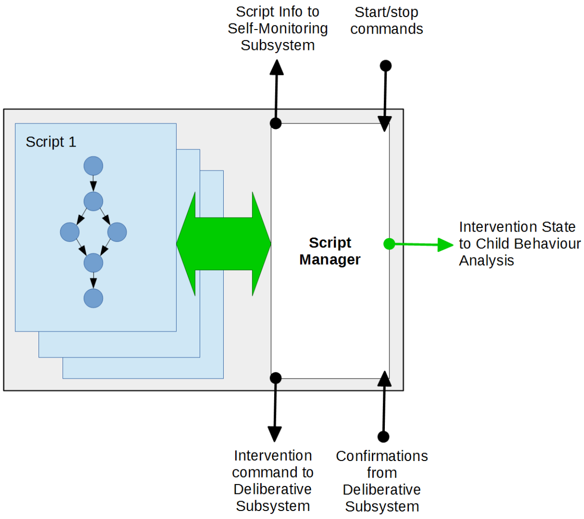

Figure 3: Overview of the script manager subsystem. The scripts are defined independently of the script manager, which is responsible for stepping through the script as appropriate and communicating with the other subsystems as required.

This makes it straightforward to add new scripts or modify existing scripts as required. This logic management could in the first instance be achieved using a Finite State Machine (FSM). Details of the preliminary script manager implementation may be found below in section 2.3.

One possibility for the scripts is that each step in the script be defined as a 3-tuple of the form: *[existing_state, proposed_action, consequent_state]*. In this context, *existing_state* could be defined by default to be the *consequent_state* of the previous step. The *proposed_action* defines what action should be taken by the robot, and be one of the actions (or unique identifier thereof) defined in D1.2. The *consequent_state* defines what robot state should be expected (in terms of sensed state) if the *proposed_action* were successfully completed. This may be used by the script manager to determine if and when it is appropriate to move onto the next script step. These 3-tuples may initially be held in a plain text file to facilitate examination and modification by the clinical staff as required. This can be changed later to ease the process (for example by providing a drag-and-drop script construction GUI).

The Deliberative subsystem is the primary locus of autonomous action selection in the cognitive controller (figure 2). This subsystem takes as input sensory data, child behaviour information, information on what step should be next executed from the therapy script, and higher-level direction from the Wizard/Self-Monitoring subsystem. It then proposes what action should be taken next by the robot (this proposal is sent to the Expression and Actuation subsystem). In a normal script execution context, the Deliberative subsystem is the primary driver of behaviour, which would typically propose the next script step. Details of the deliberative subsystem implementation may be found below in section 2.3.

There are however a number of circumstances in which this is not the most appropriate action to perform. For example, if the child is detected to have very low engagement with the task (as determined from the WP5 component/s, and/or information from WP4 sensory system saying the child is looking away for example), then it would be appropriate to attempt to re-engage the child with the robot/task prior to executing the next stage in the therapy script. In this case, the Deliberative subsystem can choose to depart from the behaviour defined in the script, and instead propose a different behaviour.

2.3 Preliminary Deliberative Subsystem Component

Based on the functional description of the cognitive controller system of the DREAM architecture (see section above, and "*Organisation of Cognitive Control and Robot Behaviour*" annex), preliminary implementations of the deliberative subsystem and script manager components have been formulated.

These first versions of the components are defined in terms of the input and output ports, following the guidelines established in the software engineering standards (WP3). These preliminary versions are directly informed by the development of the WP6 control architecture in Y1, where each subsystem was defined in terms of the interactions with other subsystems, and their functions as outlined in the DREAM DoW. Please refer to figure 1 to provide this context.

2.3.1 Script Manager

The functions of the script manager are described above: the main point is that the script manager is separated from the rest of the deliberative subsystem, which is instead focussed on autonomous action selection. The ports for this component are described in figure 4. As per the software engineering standards, each of the port names shown is prefixed by the system and subsystem names. For the script manager, this prefix is "*/cognitiveController/scriptManager/*".

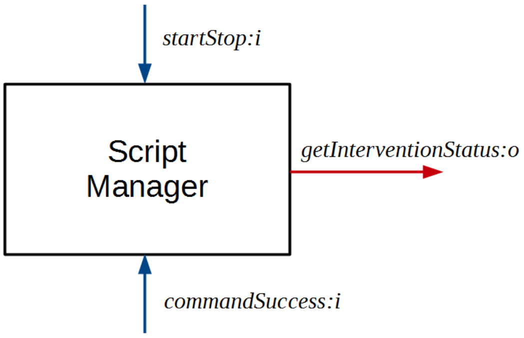

Figure 4: Script Manager component yarp ports: the prefix for the port names is listed in the main text. This component handles the script state and provides this information to other components.

The */getInterventionStatus:o* port provides the current state of the intervention, which can be accessed by any component that requires it (notably the WoZ interface, the deliberative subsystem, and the child behaviour recognition component). This port is of type *BufferedPort<VectorOf<Int> >*. The *startStop:i* port is used by the therapist GUI to indicate the start and end of the interaction. The *commandSuccess:i* port provides feedback that the last requested intervention command was successfully exeecuted (or not). In this preliminary specification, both of these ports are of type *BufferedPort<Bottle>*.

2.3.2 Deliberative Subsystem

As with the script manager component, the Deliberative subsystem port names are prepended with the subsystem name, in this case "*/cognitiveController/deliberativeSubsystem/*". An overview of the ports for this component are shown in figure 5.

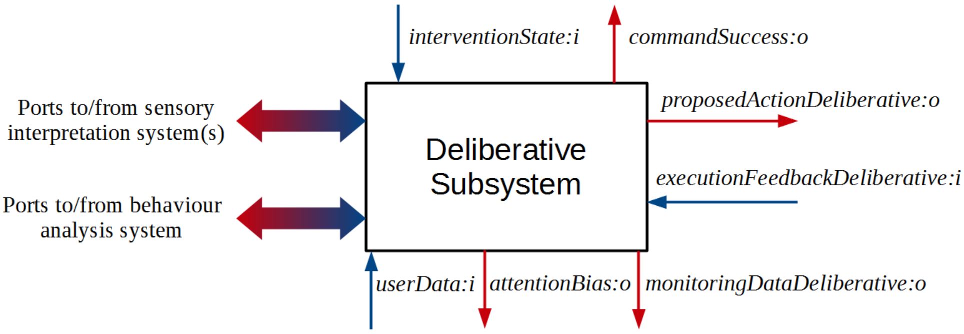

Figure 5: Deliberative subsystem component yarp ports: the prefix for the port names is listed in the main text. This component is responsible for the autonomous action selection in cases of deviation from the script, etc. Ports to/from the sensory systems and child behaviour analysis system are not shown for clarity.

For the sake of clarity, the ports to/from the child behaviour classification component (3 ports) and the sensory interpretation component (25 ports) are not shown in figure 5: full details of these may be found in D3.1. The "*getInterventionStatus:i*" is of type *BufferedPort<VectorOf<Int> >*. In the preliminary version of the Deliberative subsystem, the remaining six ports are of type *BufferedPort<Bottle>*.

Further to the definition of this preliminary version of the yarp component to fit into the DREAM architecture according to the established software engineering framework (see D3.1), a range of work has been conducted on the theoretical basis of autonomous action selection mechanisms for social robots. In addition to the technical principles described above (section 2.1), this development work has also included the principled examination of what sort of functionality should be undertaken by this component, and what the limitations of this should be with respect to the supervisory oversight provided by the ever-present therapist (as defined by the *supervised autonomy* objective of DREAM). Our preliminary work in this regards is summarised in annex 4.1.

3 Script Following

A primary objective of the first year of work is the evaluation of manually controlled ('wizarded' or tele-operated) versions of the diagnosis/intervention scripts defined in deliverable D1.1, as relevant to T2.1. This evaluation provides guidance for the further development of the autonomous interpretation and behaviour systems for the DREAM architecture.

In Y1, WP6 has provided substantial support to provide the systems necessary for these evaluations. The robot behaviour capabilities to enable execution of each of the basic versions of the scripts has been implemented: imitation task, joint attention task, and the turn-taking task. These systems are currently deployed and in use at partner UBB.

Two methods were used to provide this functionality. For the imitation and joint attention tasks, behaviours were constructed in the Aldebaran-produced Choregraphe suite, such that a therapist could manually control the robot behaviours for each of the stages of the task. Details of this system can be found in annex 4.4 of this deliverable. For the turn-taking task, since the Sandtray device is used, a standalone system using the software engineering standards defined in WP3 were used. Details of this system can be found in annex 4.3 of this deliverable.

This work provides the basis for further developments within WP6 in Y2 and beyond. The development of the behaviours for each of the intervention tasks can be reused in the autonomous versions of these tasks, along with further behaviours as required. Furthermore, the establishment of preliminary versions of the various components using the software engineering framework, and the development of the WP6 cognitive control architecture, will facilitate the implementation of the autonomous versions of these components, and their subsequent integration with the rest of the system.

References

[1] Serge Thill, Cristina A Pop, Tony Belpaeme, Tom Ziemke, and Bram Vanderborght. Robot-assisted therapy for autism spectrum disorders with (partially) autonomous control: Challenges and outlook. *Paladyn*, 3(4):209–217, 2012.

[2] Toby Tyrrell. Computational mechanisms for action selection. 1993.

[3] R C Arkin. Homeostatic Control for a Mobile Robot - Dynamic Replanning in Hazardous Environments. *Journal of Robotic Systems*, 9(2):197–214, 1992.

[4] D Feil-Seifer and Maja Matarić. B3IA: A control architecture for autonomous robot-assisted behavior intervention for children with Autism Spectrum Disorders. *RO-MAN 2008. The 17th IEEE Internal Symposium on Robot and Human Interactive Communication*, pages 328–333, 2008.

[5] Bruce Blumberg. Action-selection in hamsterdam: Lessons from ethology. *Proceedings of the third international conference . . .* , 1994.

[6] O Avila-Garcia and L Cañamero. Using hormonal feedback to modulate action selection in a competitive scenario. *From animals to animats*, 2004.

[7] John Lones, L Cañamero, and Matthew Lewis. Epigenetic adaptation in action selection environments with temporal dynamics. *Advances in Artificial Life, ECAL 2013*, pages 505–512, September 2013.

[8] Cynthia L Breazeal. *Designing sociable robots*. MIT press, 2004.

[9] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, pages 237–285, 1996.

[10] Matthias Scheutz and Paul Schermerhorn. Affective goal and task selection for social robots. *Handbook of research on synthetic emotions and sociable robotics: new applications in affective computing and artificial intelligence*, page 74, 2009.

[11] Anthony R Cassandra. A survey of pomdp applications. In *Working Notes of AAAI 1998 Fall Symposium on Planning with Partially Observable Markov Decision Processes*, pages 17–24. Citeseer, 1998.

[12] Joelle Pineau, Michael Montemerlo, Martha Pollack, Nicholas Roy, and Sebastian Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42(3):271–281, 2003.

[13] Sonia Chernova and Manuela Veloso. Confidence-based policy learning from demonstration using gaussian mixture models. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 233. ACM, 2007.

4 Annexes

4.1 Senft, E. et al. (2015), When is it better to give up? Towards autonomous action selection for robot assisted ASD therapy

Bibliography - Senft, E., Baxter, P., Kennedy, J., Belpaeme, T (2015), "When is it better to give up? Towards autonomous action selection for robot assisted ASD therapy", HRI'15 Extended Abstracts, doi: 10.1145/2701973.2702715

Abstract - Robot Assisted Therapy (RAT) for children with ASD has found promising applications. In this paper, we outline an autonomous action selection mechanism to extend current RAT approaches. This will include the ability to revert control of the therapeutic intervention to the supervising therapist. We suggest that in order to maintain the goals of therapy, sometimes it is better if the robot gives up.

Relation to WP This work directly contributes to Task T6.3.

4.2 Baxter, P. et al. (2015), Technical Report: Organisation of Cognitive Control and Robot Behaviour

Abstract - The purpose of this technical report is to summarise the motivations and constraints underlying the cognitive control structures, and to outline an organisation of these subsystems. This is a proposal only; this document is intended to be a working one, to be updated as required during development. This version of the report is based primarily on the discussions that took place in Brussels (23/01/15).

Relation to WP This work directly contributes to Task T6.3.

4.3 Baxter, P. et al. (2015), Technical Report: Sandtray Wizard-of-Oz System for Turn-taking Intervention

Abstract - In this technical report we describe the software organisation of the Sandtray system created for the turn-taking diagnosis/intervention interactions. This system is based on the organisation defined by the WP3 software engineering standards, although at the moment does not fit into the rest of the DREAM system:this was to facilitate ease of setup and launch for the end-user (i.e. minimal installation, and no compilation required). TheWoZ system provides a GUI from which the therapist can control the robot behaviour in the turn-taking task, and logs of the interaction are automatically stored for retrospective analysis.

Relation to WP This work provides the basis of work in Task T6.3, and is relevant to T2.1.

4.4 Esteban, P.G. et al. (2015), Technical Report: Manual for the use of Choregraphe boxes in Wizard of Oz experiments

Abstract - In this technical report we describe a manual to help UBB team in the development of the Wizard of Oz experiments within Work Package 2. Both PLYM and VUB have collaborated to develop the corresponding modules in Choregraphe. This manual aims at being a reference point to ease the habituation of the therapists to the software.

Relation to WP This work provides the basis of work in Task T6.3, and is relevant to T2.1.