Deliverable D25.6

Product development accomplishment

Antonio J. Bandera Rubio (UNIVERSITY OF MÁLAGA)

Version 1
Delivery date: **11.03.2019**

CONTENTS

Glossary of Terms

CGA: Comprehensive Geriatric Assessment

ECHORD++: European Clearing House for Open Robotics Development Plus Plus (E++ for short)

1 Introduction

This document describes the features of the final design of the whole CLARC framework, including a description of the environment in which it will run, its whole design and architecture, and its current major constraints.

1.1 Global overview of CLARC

CLARC is a complete framework for robotizing two specific tests that are typically part of a Comprehensive Geriatric Assessment (CGA) procedure: the Barthel test and the Get Up & Go test. CLARC consists of two major elements: **CLARA**, a social robot able to interact with the patients, and capture and analyze the obtained data; and the **CGAmed**, a local server able to store a database with all captured data and to provide the physicians with the tools for online monitoring and offline editing and supervision.

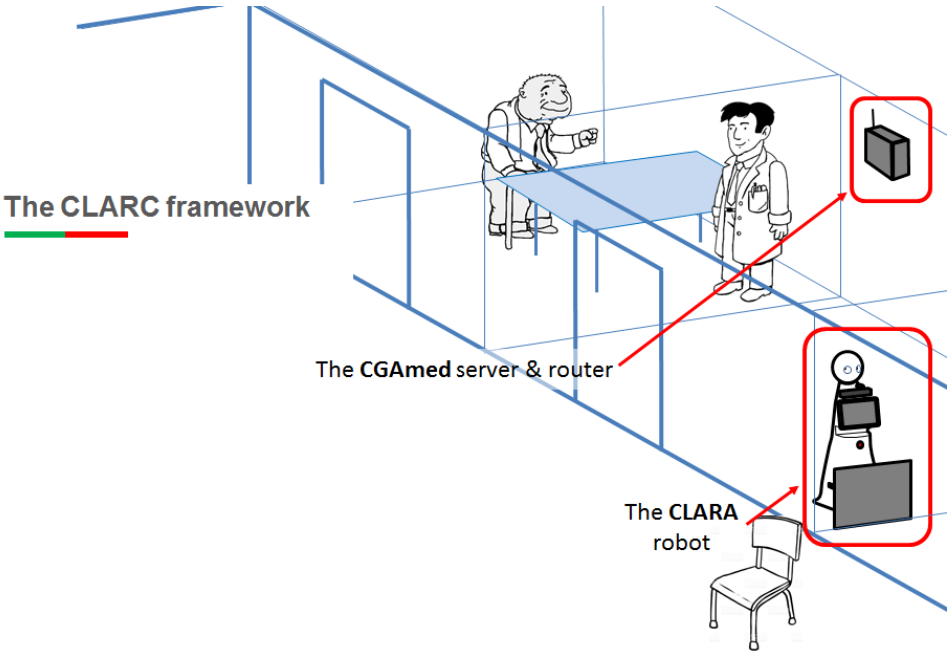

Figure 1: The CLARC framework

1.2 Main features

CLARC has been designed to be deployed without requiring any specific constraint. Thus, it currently provides all hardware items. According to several criteria, the CLARC solution can be summarized as

General

+ *Human-Robot Interaction.* The CLARA robot can drive and score the tests without human supervision. Thus, it can discharge the healthcare professional of performing the tests. Besides, the CLARC framework can also help the healthcare professional to schedule and organize the sessions, and to review and close the reports associated to the sessions. For doing this, it autonomously captures and stores the video/audio information on the CGAmed.

+ *End-User Involvement.* CLARC framework has been designed with the help of medical experts and elderly people.

System

+ *Mobility.* The CLARA robot can be easily moved by one person in the flat ground of an office-like environment. It is also possible to turn it on and move using a joystick or keyboard.
+ *Power supply.* The CLARA robot can dock in a charging station by itself. With the batteries charged, it runs for more than 8 hours.
+ *Language interface.* The CLARA robot is currently able to run test in Spanish, English and French languages. It can be programmed to work in other languages if they are available in the Microsoft Speech Platform SDK, being the major effort to generate new txt files with the sentences employed for running each test.
+ *Non-verbal interaction.* The CLARA robot is equipped with a touchscreen, which allows the user to answer questions using tactile interface. However, the position of the arm is forced to be in an uncomfortable position. The CLARC framework includes a remote control device to help elderlies answering the questions.
+ *Motion tracking.* CLARA robot uses a Kinect device for monitoring the presence of people in the surroundings. Thus, it is used for detecting for instance that the patient has get up from the chair during a Barthel test. But this device is also employed to provide the input of the Human Motion Tracking system, which can divide up the whole sequence of the Get up & Go test into actions, and provide a score for the test, according to measures chosen in close collaboration with physicians.

Evaluation and data management

+ *Patient-specific view.* Results are stored and can be evaluated for each patient. It is possible to change some test options according to the patient to match her preferences or adapt to her condition.
+ *Analysis of results.* CLARC provides an evaluation of the test but it also allows the clinician reviewing and modifying, if required, these results. It offers a report that can be easily copy-pasted by the clinician in any text box or document.
+ *Data protection.* The CLARC framework guarantees the privacy of sensible data. All these data are only processed locally. Besides, they can be accessed only by authorized personnel that needs to register in the system before handling them.

1.3 Hardware concepts

Standard hardware: The table below describes the standard hardware in an CLARC framework

Hardware	Explanation

CLARA robot	The robot is based on a differential driven platform by MetraLabs. Main components are listed in Section **The CLARA robot**.
Charging station	The robot has a charging station to be able to charge autonomously. The charging station is powered by standard main supply. In case of charging the power output is 400 W.
Remote Control	Portable device connected to the robot that allows the user to interact with the system using large buttons.
Router	CLARC works in a **local network**, in which all the components are connected to the wifi provided by this router.
CGAmed embedded PC	This PC stores all the information about users, sessions, etc.

Optional hardware: The table below describes the optional hardware in an CLARC framework

Hardware	Explanation
Remote Control (XL size)	Portable device connected to the robot that allows the user to interact with the system using large buttons and a small touchscreen.

2 Environment overview

This section provides an overview of the characteristics to be fulfilled by the environment where CLARC will be deployed.

2.1 The general environment

CLARC has been designed to be run in an office-like environment. A router is provided for creating the local network where the whole framework works, being necessary to check that the mobile element, the CLARA robot, is able to be connected in all rooms that it must visit. The CGAmed embedded PC and the charging station needs to be powered by standard main supplies (220/230 volts ac). For being able to move from one room to another one, all doors must be open. If CLARA finds a close door, it will stop. In the current version, it is not considered that CLARA must ask for help in these situations.

An example of deployment is schematized in Figure 2.

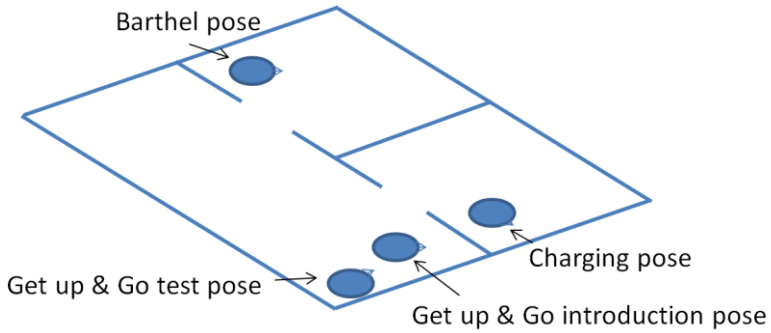

Figure 2: The goal poses required by the CLARC framework

The table below summarizes the relevant items on the figure.

Pose	Explanation
Charging pose	Pose just in front of the charging station. Once the robot reaches this pose, it can dock autonomously.
Barthel pose	Pose in front of the chair where the patient will sit down for a Barthel test.
Get up & Go introduction pose	Pose in front of the chair where the patient will sit down for allowing the robot to introduce her the Get up & Go test.
Get up & Go test pose	Pose from where the robot can correctly visualize and capture the Get up & Go test.

Constraints

The deployment of the CLARC framework must respect certain conditions. The table below describes the major constraints in the deployment of the CLARC framework.

Situation	Explanation
Light conditions	The motion capture is currently built over the Microsoft SDK and the Kinect sensor. This RGBD camera needs certain light conditions, which can be typically found in office-like environments. Low illumination values or the natural light coming from a large window can provoke that the sensor does not work.
Patient-robot distances for interaction	The Kinect sensor is employed by CLARA for detecting the presence of a person. In an face-to-face scenario, when the person is sitting down in front of the robot interacting through the touchscreen, it should be respected a patient-to-robot distance.
Patient-robot distance in	In the current version, CLARA is able to autonomously manage the Get up & Go test, but it requires that the chair where the patient will sit down is at 3 meters from the Kinect device. If the chair is more

the Get up & go test	distant than this value, CLARA could not correctly detect the sitting down action.
Safety distances for moving	CLARA uses CogniDrive from MetraLabs GmbH for safe navigation. If CLARA is close to an obstacle, it will not start to move. Moreover, it will require a safe path for moving from one pose to another one.

3 The CLARA robot

This section describes the hardware, software architecture and planning framework of the current version of the CLARA robot after Phase III.

3.1 Updating the CLARA robot (Phase III)

3.1.1 External aspect

After Phase II, all the devices needed for providing interaction or recording abilities are mounted over the external chassis of the CLARA robot. The final aspect can be seen at Figure 3. We can note the presence of an IP camera, the shotgun microphone, the Kinect sensor and a webcam.

Figure 3: External devices in CLARA (Phase III)

The major disadvantage of this scheme is the fragility of the coupling of these devices on the chassis. A minor hit can provoke an unexpected turn of the device, being the consequence that the microphone does not allow now to hear the patient or that the IP camera is not recording the session.

With the aim of providing a more robust coupling of these devices on the robot's structure, the chassis of CLARA was redesigning for Phase III. The objectives were:

+ To include the webcam and IP camera inside the head of the robot. The small monitor providing the 'face' of CLARA was removed.

+ To attach the microphone to the chassis, locating it over the Kinect sensor.

The new external aspect of CLARA is schematized at Figure 4.

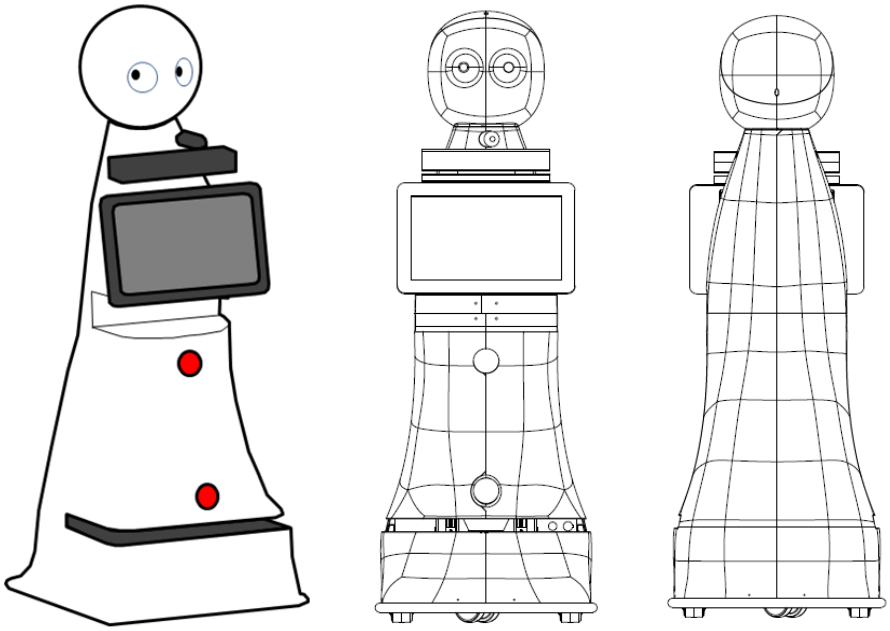

Figure 4: (Left) The new external aspect of CLARA (Phase III) and (right) external structure

The chassis is organized into six pieces that can be easily removed if needed. The disposition of the two cameras within the head and their fields of view are shown in Figure 5.

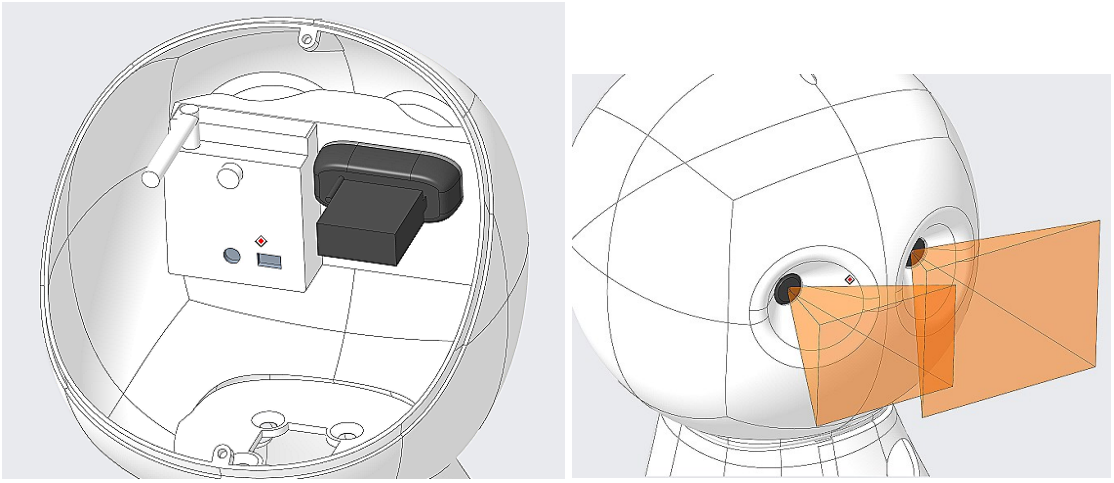

Figure 5: (Left) The bottom part of the head of CLARA (Phase III) showing the camera disposition and (right) the fields of view of both cameras

3.1.2 Description of the hardware in the CLARA robot

The table below describes the standard hardware in the CLARA robot (Phase III)

Hardware	Explanation

The motors & gearboxes	
MetraLabs HG4 main control unit	Safety motor controller and power supply, battery charging
Battery 40 Ahrs	
Bumper	Stops the robot in case of collision
Safety LIDAR	Measures distances to walls for orientation, measures distances to obstacles to avoid collisions, reduces the velocity of the robot if it is close to a person
Embedded PC Shuttle DH170	Linux based PC that runs the CORTEX architecture and CogniDrive
Embedded PC Intel NUC NUC7i7DNHE	Windows based PC for person detection, human motion capture and speech recognition
Microsoft Kinect2	Sensor for motion detection
Network camera Edimax IC-3115W WiFi	IP camera for online supervision
Webcam Logitech C310 HD Logitech	Webcam for recording the session
Soundkarte USB 2.0 ROCCAT	Converts USB to Microphone
Display 13,3" with PCAP-Touchpanel	Touchscreen for tactile interaction
Shotgun Microphone	Directional microphone for speech capture
Speakers	

Technical updates

The table below describes the technical problems detected on the hardware and the provided solutions.

Hardware	Explanation
DC/DC converter	The DC/DC converter in charge of providing the 12V for the touchscreen and Kinect sensor was not able to correctly power both devices and unexpected turn off were detected. The converter was changed by a powerful unit.
Kinect v2 sensor	One of the Kinect v2 sensors had problems and was replaced by a new unit.

3.1.3 The internal PCs

The table below summarizes the main features of the embedded PCs within CLARA

PC	Explanation
Embedded PC Shuttle DH170	Linux based PC (Ubuntu 14.04) that runs the CORTEX architecture and CogniDrive
Embedded PC Intel NUC NUC7i7DNHE	Windows based PC for person detection, human motion capture and speech recognition

Technical updates

The table below describes the technical problems detected on the installed software and the provided solutions.

Software	Explanation
Ubuntu updating	The new embedded PCs within CLARA are differents of those in the prototype for Phase II. This difference caused some incompatibilities between the new hardware and the Ubuntu 14.04 operating system version needed by our software components. Specifically, the new graphic and sound cards were not supported. To solve it, the Ubuntu 14.04 kernel and its tools packages were updated to Ubuntu 16.04 ones. This implied to install new releases of some of the software tools.
Webcam recording	The webcam is employed for recording the session and providing the video pieces to be stored in the CGAmed. The required software modifications made that the previous video recording component of CLARA (based on VLC) was not able to record video and audio simultaneously. The CLARA component was modified to use FFmpeg based video and audio recording.

3.2 The software architecture CORTEX

3.2.1 Motivation

Abstract reasoning about concrete phenomena is intimately tied with the existence of an internal representation of this reality. From a robotic perspective, this implies the establishment and maintenance of a connection between what the robot reasons about and what it can sense. The symbol binding problem is a challenge, which is deeply connected to the more general question of meaning. It directly points at the problem of keeping a bond between a concrete object and a symbol. This problem has been approached from very different points of view by many researchers in recent decades. Among these proposals, some recent contributions point towards the use of a shared, unique internal representation. Contrary to purely symbolic representations, this shared memory should be fed with the symbolic and concrete tokens generated by all the software components in charge of solving the grounding problem. The richness of

this representation usually forces to consider those mechanisms in charge of making decisions on the basis only of what is relevant to an ongoing situation, without having explicitly to consider all that is not relevant. The problem becomes more complex with the inclusion on this representation of the actions, and their possible effects on both entities and the global evolution of an ongoing task. This complexity has been typically bounded by including on the framework a software module in charge of choosing the specific action to be performed. The importance of this module is significant, as it usually determines the subtasks to be addressed by the rest of modules on the architecture. In classical three-layer architectures, this role resembles the one of the Sequencer module. But the most relevant characteristic of this module (e.g. the goal manager on the original Cosy Architecture Schema or the Executive one on the RoboCog architecture) is that it sets a direct pipeline between the responses of the modules that determine the current goal, and the behavior to be addressed by the rest of modules to achieve that goal. This mechanism alleviates the representation of assuming this responsibility, being unnecessary to annotate these behaviors on it.

Significantly, this scheme implies that the modules will execute the required subtask as they receive a direct command. Hence, the internalized state of the world does not guide its behavior and it will be only used, as it is typical in the blackboard models, to share information about goals or partial results among the agents. When we started to design the software architecture of CLARA, we decided to change this scheme by removing this goal manager and forcing all the modules on the architecture to encode perceptions and actions using the same set of tokens. Briefly, as it occurs with perceptions, actions will also be thought of as changes to the inner world. It is in this new proposal where the shared representation truly becomes the core of the architecture, storing all data that is required for the software modules to perform their activities. This simplifies the architecture, as no further modules are required to take the responsibility of understanding the whole state of the robot and its context. Furthermore, fewer connections eases intercommunication and generalization. Although this scheme forces the task-dependent modules to use a more complex logic to infer their activities from the state (affordances), as they will not receive specific action commands, they have shown that can be more easily modified, added or removed without affecting the rest of the architecture.

3.2.2 The Inner World

The existence of a mechanism for representing the information related to the ongoing tasks is common to cognitive architectures, which use this *working memory* for driving attention, reasoning and learning. In our proposal, this memory is the *Deep State Representation* (DSR). The DSR is a multi-labeled directed graph which holds symbolic and geometric information within the same structure. Symbolic tokens are stated as logic attributes related by predicates that, within the graph, are stored in nodes and edges respectively. Geometric information is stored as predefined object types linked by 4 x 4 homogeneous matrices. Again, they are respectively stored as nodes and edges of the graph. Figure 6 shows a reduced view of the state associated to the execution of a test . The **person** and **robot** nodes are geometrical entities, both linked to

the **world** (a specific anchor providing the origin of coordinates) by a rigid transformation. But, at the same time that we can compute the geometrical relationship between both nodes (RT^{-1} x RT'), the **person** can be located (**is_with**) close to the **robot**. Furthermore, an agent can annotate that currently the **robot is_not speaking**.

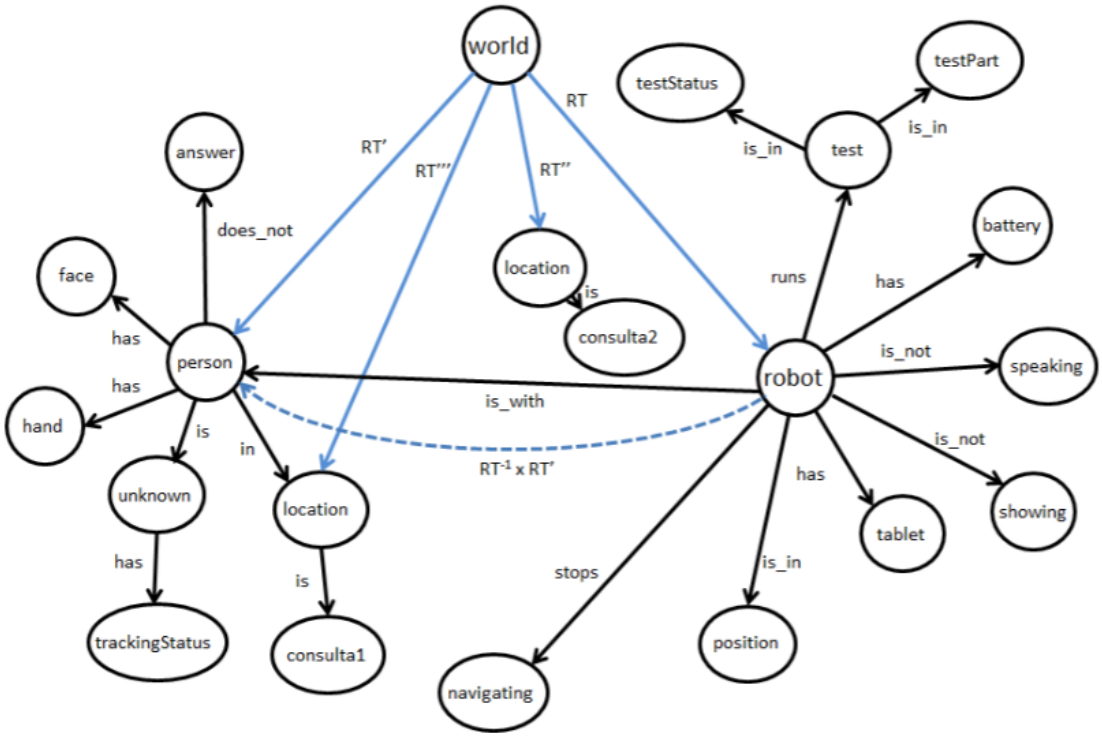

*Figure 6: Unified representation as a multi-labeled directed graph. For instance, edges labeled as **is_with** or **is_not** denote logic predicates between nodes. On the other hand, edges starting at **world** and ending at **person** and **robot** are geometric and they encode a rigid transformation (RT' and RT respectively) between them. Geometric transformations can be chained or inverted to compute changes in coordinate systems (see text).*

The complexity of the domain-dependent modules typically implies that they will be internally organized as networks of software components (compoNets). Within each compoNet, the connection with the DSR is achieved through a specific component, the so-called agent. These agents are also present in previous architectures. In fact, they are needed in any blackboard-based scheme. However, its degree of complexity has significantly changed when we move from these schemes to the proposed one. With the removing of the goal managers, our agents need to search for those changes on the DSR that launch the specific problem-solving skills (actions) of the compoNets they represent (e.g. detect the pose of a person, or start to speech a sentence). This mapping between subgraphs on the DSR and actions constitutes the knowledge on each agent, which is intimately linked to our definition of affordances. The internal data flow of these agents is briefly outlined at Algorithm 1.

```
while (1) do
    subscribe to DSR updates;
    search_for_changes { output: action };
    process { action };
    if DSR_changes then
        update DSR;
    end
end
```

Algorithm 1: Procedure of an agent within our proposal

The **search_for_changes** skill depends on each agent and the behaviors that the compoNet can solve. Within the algorithm, it is stated that this function returns the *action* to perform. This is the most significant difference between other blackboard-based approaches and our proposal: in the first case the action is imposed by an external module, but in the second one, the action is determined by the agent.

3.2.3 Overview of the archictecture

Figure 7 shows an overview of the whole architecture in charge of performing the CGA tests within the CLARC project. Surrounding the inner world provided by the DSR there are eight agents: PELEA, Speech, Panel, Tablet, CogniDrive, Recorder, HMC and Person.

+ The PELEA agent is in charge of providing the deliberative skills to the architecture. It is an instantiation of the Planning, Learning and Execution Architecture (PELEA), which maintains its own internal memory and the software modules for monitoring the course of action. It interacts with the rest of agents using the same procedure (i.e. changing the Inner World).
+ The Recorder agent manages a IP camera, which provides a stream of video for online supervision and also records the session for offline visualization.
+ The channels for patient--robot interaction are provided by the Speech, Panel and Tablet agents. The Speech agent is the responsible of understanding the answers of the patient or guardian and of translating the text into speech, generating the voice of CLARA. It is internally connected to the WinSpeechComp, a module in charge of generating the voice from text using the Text-To-Speech (TTS) software provided by Microsoft Speech Platform SDK. This software is also used for voice recognition, with the help of specific grammars that are loaded for each question, in order to maximize recognition rates. The transcription system of this ASR recently reached the 5.1 percent error rate, the same rate measured for humans (human parity word error rate). Furthermore, the use of the Microsoft SDK eases us to implement a multi-language interface, which is currently able to interact with the patient in French, English or Spanish. The verbal channel is enhanced by using a touchscreen on the torso of the robot. The Panel agent manages the tactile interaction and the information shown in this touchscreen, whose design has been carefully designed for dealing with elderly people. The position on the robot of the touchscreen is shown in Figure 4. The intense use of this quasi-vertical touchscreen forces the patient to adopt an uncomfortable position, not only because of keeping the arm extended, but also because the robot cannot be so close to the patient to avoid that s/he is continually approaching/moving away of the screen to touch it. For avoiding this, we add a third element to the interface: the Remote Control device described in Section 4.

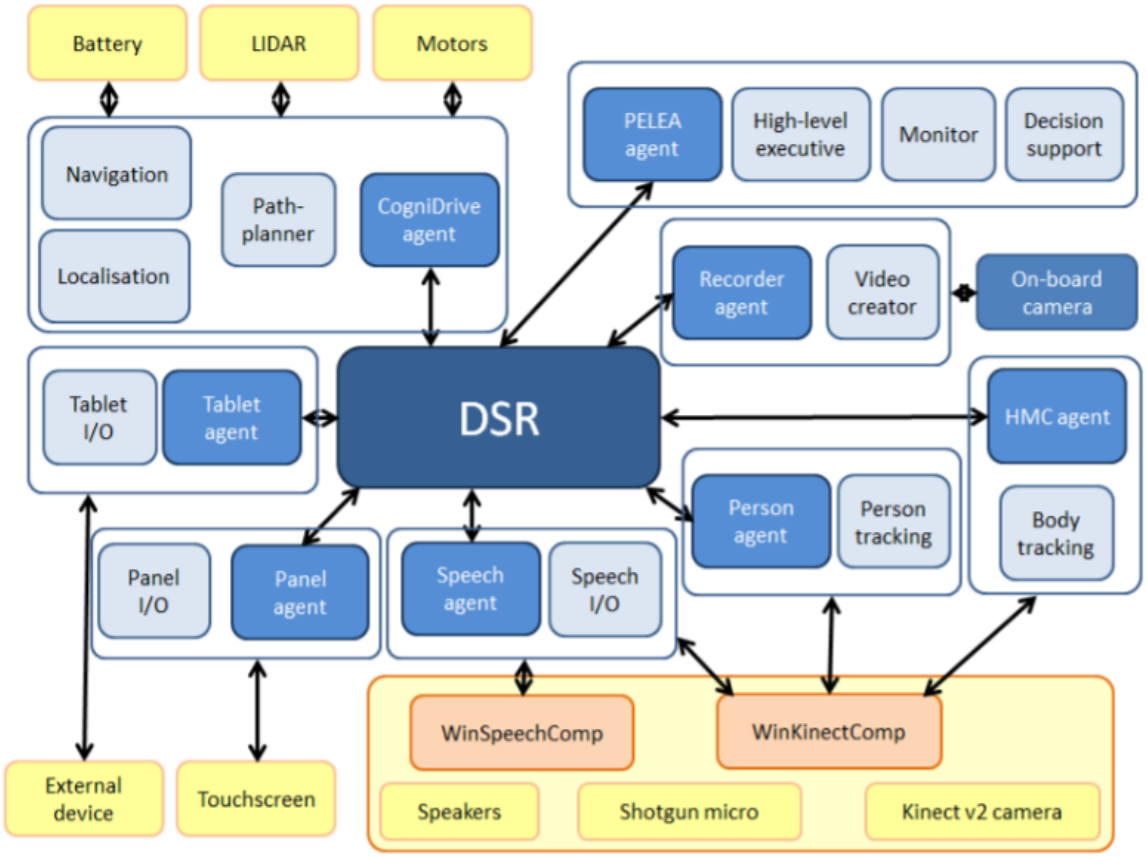

Figure 7: Overview of the CORTEX architecture endowed in CLARC

+ The CLARA robot is built over the MetraLabs SCITOS G3 platform. This base uses a LIDAR sensor for localization, navigation and obstacle avoidance, functionalities that are provided by the CogniDrive software running over the MIRA middleware. The agent CogniDrive implements a bridge for connecting these software modules to our cognitive architecture.
+ Finally, the Person and HMC agents are in charge of detecting, tracking and capturing the motion of the people sharing the environment with the robot. The Person agent is the responsible of detecting and tracking the upper body of the interviewed person, meanwhile the HMC agent captures the motion of the full body of the patient. For solving these tasks, both agents are connected to the WinKinectComp module. This module is in charge of capturing the preprocessed data provided by a Kinect sensor v2 (i.e. joints and face of the person).

3.3 Using Automated Planning to Control the Social Robot

In this section we describe how we have used Automated Planning to control the robot performing CGA. Summarizing, we use a sense-plan-act approach, where the robot receives information about the environment, creates a plan with it and executes an action. The environment is sensed again to compare the results of the action to the expected ones, and either the next action is executed or a new plan is created in case of discrepancy.

We start first describing the planning paradigm chosen, Classical Planning, and the rationale followed to select it. Next the PELEA planning-executing architecture used is introduced, and finally, some insights about how we have modeled CGAs in PDDL are provided.

3.3.1 Selecting the Planning Paradigm

Two main planning approaches have been used in the literature for the high-level control of robots: Timeline-based and Action-based.

Timeline-based planning has been extensively used in robotics. However, the definition of domains can be almost as difficult as dealing with pure Finite State Machines and requires expert knowledge. In addittion there is a low number of planners supporting this planning approach, and a variety of definition languages.

In Action-based planning, which is the one used in this project, the task is usually described as a tuple P={F,A,I,G}. F is a finite set of facts, expressed using predicate logic, that can be true or false. Each state is defined by the set $s \subset F$ of facts that are true. A are the actions that the robot can perform, $I \subset F$ is the initial state or, in other words, the current state perceived by the robot, and $G \subset F$ describes the goal state(s); the desired outcome of the human-robot social interaction. Search is typically used to find the sequence of actions going from I to F. Action-based planning has the advantage of being easier to model, having standard languages like PDDL, where the domain and problem models are defined. Once the model is created several different domain-independent planners can be used to find a plan.

Different approaches exist also inside Action-based planning; in this work, classical planning is followed due to several reasons. Actions performed by the robot are sequential, and we do not need action overlapping (due to the sequential nature of social interaction), we found it more useful to rely on the Monitoring module to deal both with duration and preconditions checking. For each action a maximum duration is set; if action does not end in the expected time it is marked as failed and we replan. Besides, each condition is tagged to specify whether it must hold at the start, end or during the whole action. Monitoring continuously checks them and if one precondition no longer holds, replanning is triggered. Using this schema we simplify the model, increasing the scalability of the planner and decreasing its response times.

Other approaches, as temporal, probabilistic or hierarchical planning are discarded mainly to modelling difficulties and algorithmic complexity of current solvers, which makes its use in social robotics unpractical.

3.3.2 The Planning architecture: PELEA

The Automatic Planning and Monitoring PELEA module, shown in CORTEX architecture (Figure 7), is in charge of commanding the robot to perform the tests. Figure 8 shows a deeper description of this architecture.

At the beginning, the Executive has the domain and the problem with the initial internal state of the robot (e.g., test name, number of questions). Then, it completes this information using the information recovered from the Inner World (e.g., the patient is far away from the robot, the patient is seated). This external information is recovered by the PELEA agent (step 1 in the Figure), and sent to the Executive module (2), which

in turn redirects them to the LowToHigh module (3). This module transforms the received external information into high-level predicates which joint to the internal predicates, creating a high-level state, which is sent back to Executive (4).

This complete high-level state is sent to Monitoring (5) to check if it is compatible with the expected state of the world. If it is the first plan or if the previous plan is not valid anymore, Monitoring retrieves a plan from Decision Support, which encapsulates a high-level planner (6,7). This plan is stored by Monitoring, which returns the next action to Executive (8). If, in contrast, the actual state of the world is compatible with the expected one, then Monitoring just returns the next action of the previously planned plan (skips steps 6 and 7). Then, Executive transforms this high-level action into a set of low-level actions with the help of HighToLow (9, 10), and sends it to the Agent (11), which inserts it into the Inner World (12). Other compoNets scanning the Inner World will notice the changes introduced by PELEA Agent and act consequently. Then, PELEA Agent checks the Inner World for relevant changes until it notices the execution is finished (whether it has finished correctly, or it has been interrupted). After that, it retrieves again the external information needed to complete the expected state (1) and the cycle starts again. In this scheme, the Executive has full control of the execution, timing the maximum duration of an action, pauses, etc. to control the pace of the social interaction.

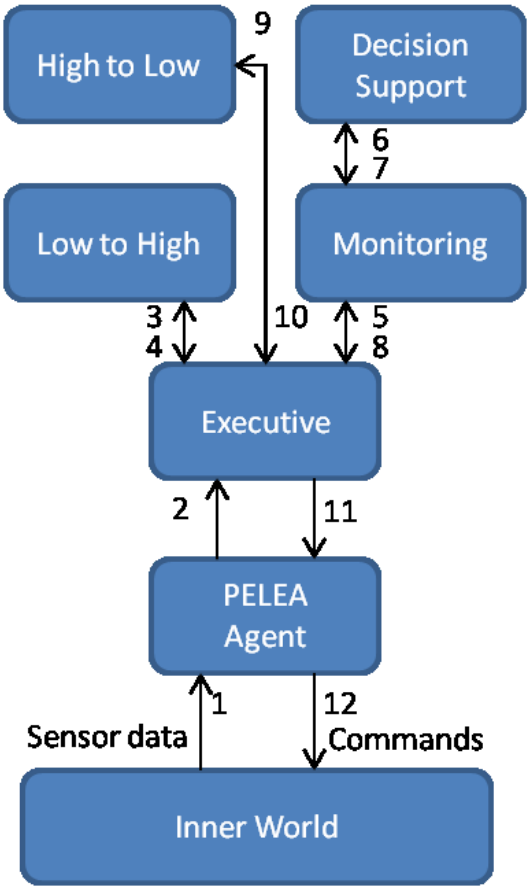

Figure 8: The PELEA framework

3.3.3 Integration of PELEA into CORTEX

Finally, by way of example, Figure 9 shows how PELEA communicates with other compoNets through the Inner World for the execution of the high-level action SAY. In the figure, Executive receives the high-level action, SAY label, from Monitoring, where *label* represents what the robot has to say. In this point, it is important to be aware of the fact that the planning system uses a high-level of abstraction while the robot works at low level. For this reason, a conversion is needed to translate high-level actions into low-level commands, and this conversion is carried out by the HighToLow module. In this example, the high-level action SAY is decomposed into two low-level actions or commands interpretable by the robot: *say label* and *show-screen*, so the different interaction modes are controlled in the low level. The first command is used to make the robot speak, and the second to show on a screen what it is saying.

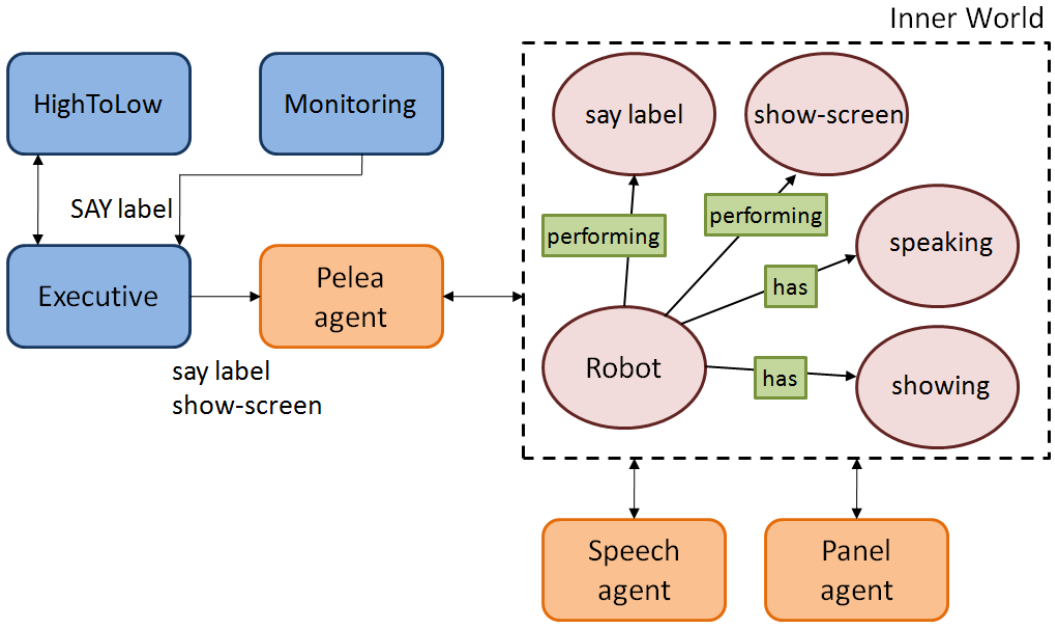

Figure 9: Connecting PELEA with the rest of the CORTEX architecture

Then, PELEA Agent receives these low-level actions and writes them in the Inner World. In the example used in the Figure, PELEA agent adds two nodes in this graph, one per action: say label and show-screen. The Speech agent and the Panel agent detect these new nodes, and they change the arcs between **robot** and **speaking** and **robot** and **showing** from has to is. Then, the robot says the label, and it also shown on its screen what it is saying. Each set of low-level actions are executed in parallel. During the execution of the action, PELEA constantly monitors the graph for relevant changes, so interruptions or replanning requirements can be detected.

3.3.4 Modelling CGAs with Planning Domain Definition Language

There are several challenges identified while using classical planning to model interactions in Social Robotics. On the one hand, those related with the modelling of the world and the use case, as state modelling or nominal behaviour specification. On the

other hand, those related with the continuous world execution, as the definition of corrective actions or replanning, continuous monitoring, interrupting actions, etc.

Domain actions are defined in terms of their preconditions and effects using PDDL (Planning Domain Definition Language) with numeric fluents (functions) and conditional effects. Both, preconditions and effects refer to the predicates and functions defined in the conceptual knowledge model. We do not include in this report specific details, but the nominal flow defined for the human-robot interaction. Specifically, Figure 10 shows a diagram with the actions of a generic nominal execution flow that represents the general use-case of a CGA, which has been later modelled using PDDL. This diagram contains bifurcations depending on the specific definition of the interaction domain. The nominal execution flow of the robot is divided into three phases: configuration, execution of components and finish interaction. The execution of components phase has three steps for every component: perform communicative act, process answer and finish component. The first two steps can be repeated depending on the communicative acts of the component.

The configuration phase contains the action *configure_interaction*. The configuration is performed at the low level, so that the high level just controls it has been done. The configuration consists of the selection of the interaction language, the verbal tense, etc.

Then, the nominal execution flow continues as follows:

1. The robot performs the first communicative act of the current component, that may involve or not to execute previously a behaviour, as showing a video. Involved domain actions are *say* and *execute-behaviour&say*}.
2. The robot keeps doing Step_1 until all communicative acts of the component have been performed or an answer is needed (i.e. the communicative act requires feedback). In that case, it executes the action *receive_answer*.
3. Then, the robot waits for the answer, giving the user a maximum number of attempts to receive it. Each attempt involves to repeat the last communicative act to ask for the answer again. Some atomic acts may require to validate the answer plausibility (for instance, that it is included in the set of valid answers for a closed answers question) or to validate its accuracy (for instance, that the answer makes sense for an open answers question). If the answer is not acceptable, the communicative act is repeated to its maximum number of attempts.
4. If the maximum number of attempts is reached, the flow continues to the following communicative act which defined as a different way of achieving the answer. If no such way exists the nominal flow is interrupted.
5. If the answer is achieved, it could be the case that it is the answer for a component that requires an answer confirmation. In this case, the nominal flow will contain two actions: *offer_confirmation* and *receive_confirmation*. When the confirmation is not received the nominal execution flow is interrupted (replanning is required).
6. The final action for the interaction of every question is *end-component-interaction*, which allows either going to the next question or to finish the test.

In addition to the actions described in the nominal execution flow, the domain contains the control action *prepare-component-farewell*. It is not associated to any low level action, but modifies the control to advance to the first communicative act of the component farewell. This action can appear in the nominal control flow for components which communicative acts define alternative ways of getting the answer, when the answer is received before the last of those alternative ways.

The domain contains corrective actions to deal with the following situations that may arise when the robot is interacting with the patient:

+ Change of answer. It is produced when an answer should be confirmed by the patient and he/she does not confirm it.
+ Answer failed event. It is produced when a communicative act for obtaining the answer has been repeated its maximum number of times and the answer (or a plausible or correct one) has not been received.
+ Component failed event. It is produced when the maximum number of failed answers for a component has been reached.
+ Maximum number of failed components event. It is produced when the maximum number of failed components has been reached.
+ Detected event. It is produced when an event has been detected during the execution of a component (question). It forces the low level to execute the behaviour defined to manage the corresponding detected event.
+ The interaction requires to call the supervisor at the end. It occurs if there has been a situation during the interaction that requires the supervisor to be called and the end.

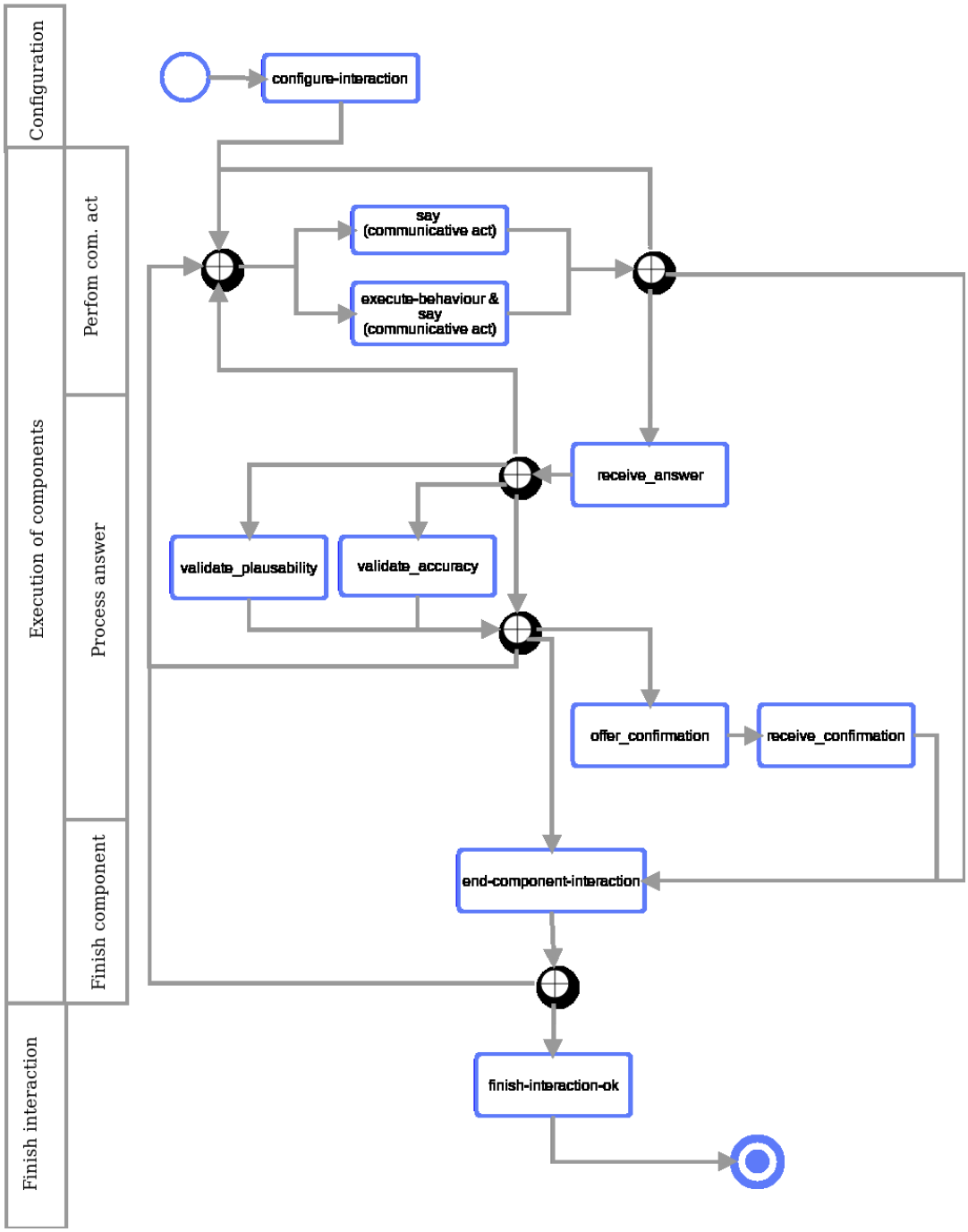

Figure 10: Diagram showing the actions of a generic nominal execution flow that represents the general use-case of a CGA (see text for details)

4 The Remote Control

This section describes the structure and features of the Remote Control device.

4.1 Updating the Remote Control device

As Figure 11 shows, the Remote Control device employed on the tests at Phase II had a relatively large size. It was built around a Tablet device, and it incorporates large buttons to ease the interaction with the elderly.

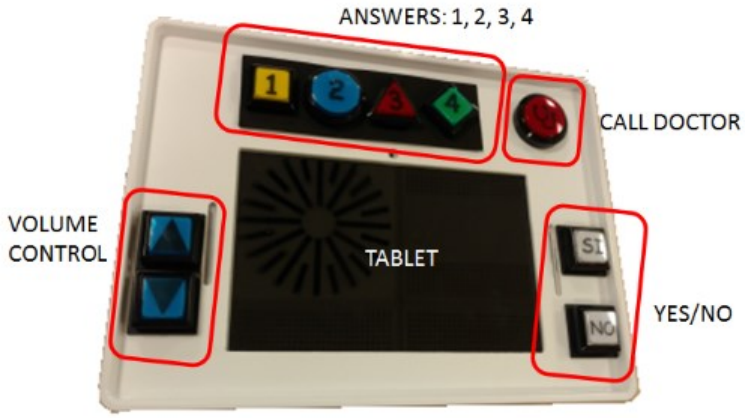

Figure 11: The Remote Control device (Phase II)

The device has been updated during Phase III. The core of the new Remote Control is a Raspberri Pi Zero and the tablet device has been removed. The final aspect can be seen at Figure 12. It now incorporates a charging port, which allows external battery charging. The buttons can be lighted, and this is used for marking those available questions in the Barthel test.

The external chassis was updated during Phase III. The current one eases the user to take the device and employ it during the test.

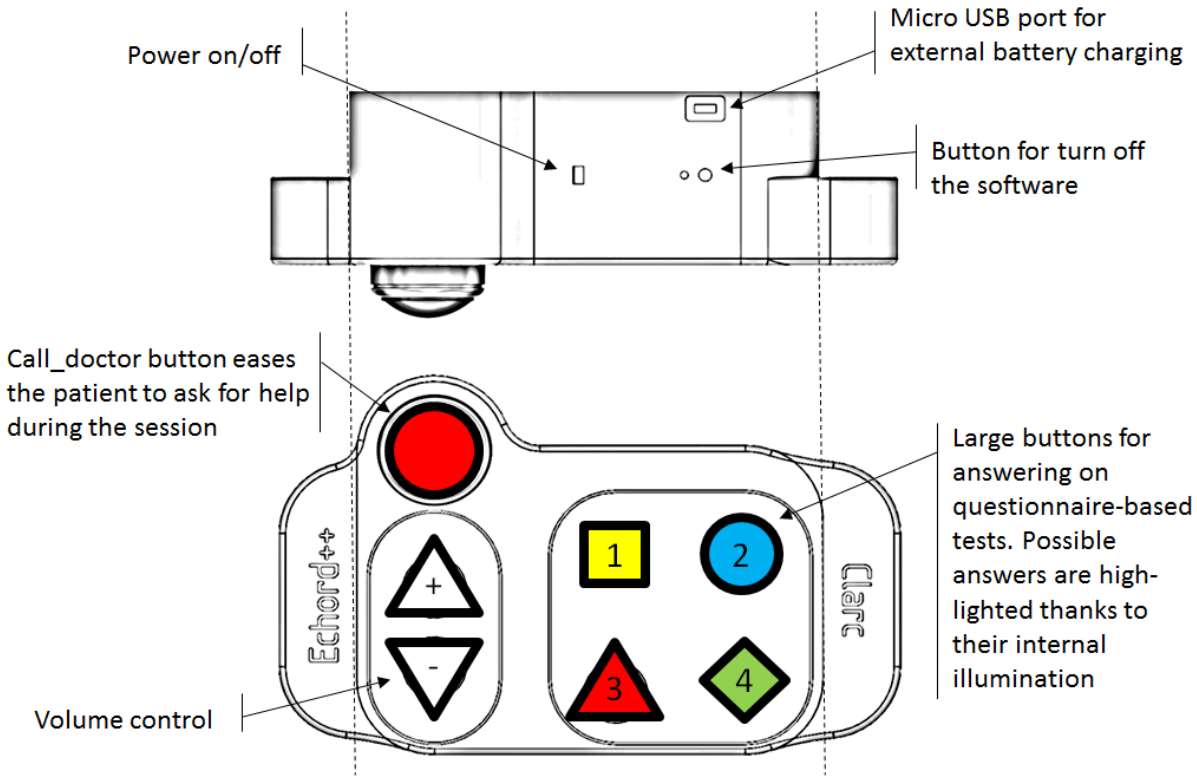

Figure 12: The Remote Control device (Phase III)

5 The CGAmed

This section introduces the webs in the CGAmed station, that are described in the tables below:

Web	Explanation
Administration 192.168.0.70	The administration web is used to configure • The positions where the robot is going to perform the tests • The list of patients • The IP address of the camera for online supervision mounted on CLARA robot (Section 1.1.1 - The CLARA robot)
CGAmed 192.168.0.70/cgamed	The CGAMed is used to: • Add new patients. • Add new sessions. • Start/Stop a session. • Pause/Resume a session. • Move the robot to a position (from a list of predefined ones). • See and compare the results of the tests.

5.1 Updates on Phase III

Modification	Explanation
Barthel results visualization	To allow clinicians to get and idea of the Barthel results at a glance, answered questions with the maximum score are shown in green, unanswered questions are shown in red and the rest ones are shown in gray.
Get up & Go results visualization	Following the recommendations of the clinicians, the visualization of the graphics has been replaced with the visualization of four measures related with the patient gait: the time takes by the patient to perform the test, the number of steps, the length of the steps and the velocity.
Medical report	For each test, a medical report is automatically generated. It can be copied and pasted.
Comparison of several tests	In the previous version of the CGAmed only the results of two tests can be simultaneously visualized to compare them. In the new version, more than two tests can be visualized at the same time.
New patients addition using CGAMed web	New patients can be added to the scheduler using the CGAMed web interface.

6 Constraints

Module	Constraint
CGAmed	The IP Address is currently the same in all CGAmed stations. This will provoke conflicts when several robots work in the same environment.
CGAmed (Adm web)	The current version of the Administration web only runs in Spanish.
CLARA robot	CLARA uses a Kinect v2 sensor for person monitoring. This device is deprecated and alternatives are under evaluation. The most probable replacement for this device will be the Azure Kinect[i], that will be available in June, 2019, and its the natural evolution of Kinect v2. But other already available alternatives are also being evaluated. The Orbbec Astra Pro[ii] is one promising option in cost and features. Although our first evaluation test reports that light conditions must be more controlled than with the Kinect v2 sensor, it has been successfully employed in other robots in the Consortium. Additional alternatives under evaluation are stereo cameras, such as Karmin2[iii]. Finally, there are systems such as the Blumblee2[iv] that must be discarded due to its high price.
CLARA robot	The use of the Kinect v2 sensor forces our design to maintain two embedded PCs in the robot: one running in Linux and endowed with all the architecture; and one running in Windows, which is in charge of executing the WinKinectComp agent. Removing the Kinect sensor opens the possibility of also eliminating this second PC for the platform. But even if the new sensor and SDK are able to run in Linux, integrating all processes into one computer is not easy: CLARA is currently able to recognize speech or track the person on-line, and this behaviour will need a computational cost that is now provided by one PC. Removing this PC implies to move this workload to the Linux-based PC. Embedded vision and audio can solve this problem, using dedicated AP SoC for managing the audio and video channels.
CLARA robot	CLARA uses the Kinect sensor to detect the presence of the person during the test, but this ability relies on the detection of her full body (upper limbs, torso and head). Sometimes, patients remain quiet or are partially out of the field-of-view. In these cases the skeleton can easily be lost, making the robot thinks that the person has get up of the chair. The full image provided by the

	sensor could be easily processed (face detection and tracking) to reinforce perception and validate this event more robustly.
CORTEX architecture: scene understanding	CORTEX allows the integration of new agents to perceive the scene and detect chairs or other common objects. If the robot gets a deeper understanding of the scene, it could check the position of the chairs and validate some of the goal poses and distances without supervision. Most of the problems when closing tests are currently coming from not fulfilling distance-based constraints (Section 2).
Sensing light conditions	The analysis of the images captured by the cameras on the robot could allow it to automatically check the light conditions, avoiding failures and generating an alert for the doctor. Other alternatives could be built using specific luminosity sensors (e.g. the TSL2561).

[i] https://azure.microsoft.com/es-es/services/kinect-dk/
[ii] https://orbbec3d.com/product-astra-pro/
[iii] https://nerian.com/products/karmin2-3d-stereo-camera/
[iv] https://www.ptgrey.com/bumblebee2-firewire-stereo-vision-camera-systems