ECHORD++

Deliverable D25.8

Technical manual

Antonio J. Bandera Rubio (UNIVERSITY OF MÁLAGA)
Juan Pedro Bandera Rubio (UNIVERSITY OF MÁLAGA)
Rebeca Marfil Robles (UNIVERSITY OF MÁLAGA)
Adrián Romero-Garcés (UNIVERSITY OF MÁLAGA)
Alberto (need surnames and affiliation)

Version 1
Delivery date: **11.03.2019**

Use the table below as an internal changelog. Delete before submitting the deliverable.

CONTENTS

Glossary of Terms

CGA: Comprehensive Geriatric Assessment

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

1 General information

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 **CGA-med**, 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. CLARC provides all hardware items and do not require any specific constraint to be deployed.

The different components of th CLARC framework are detailed below.

1.1 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 another table in this Section.
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.

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.

The table below describes the standard hardware in the CLARA robot

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	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	

1.2 The CGAmed server

The table below describes the webs in an CGAmed station

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 Hardware)
CGAmed 192.168.0.70/cga-med	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.

1.3 User access

The table below provides the default user/password data needed to access to the modules in the system.

Module	Access
Linux based PC (CLARA)	Password: scitos
Windows based PC (CLARA)	Accessible from the Linux based PC using the Remmina remote desktop app
CGAmed embedded PC	User / password: isis / grupoisis
CGAmed	URL (CGAmed) 192.168.0.70/cgamed user / password: adminWeb / adminSecure URL (Administration) 192.168.0.70 user / password: admin / adminRobot

Note: All CGAmed stations share currently the same IP Address. Contact us if you need to change this address, as this change implies internal updates on the modules on CORTEX architecture.

2 Setting up the CLARC framework

This Section provides details about how to set up the robot, the server and the environment to use the CLARC system.

2.1 Turning the robot on

The CLARA robot is turned on using the key in its right side. When the robot is activated, the two internal PCs are automatically turned on. The screen of the robot is connected to the Linux PC. Login to this PC using the username: SCITOS Demo user (the one by default) with password: scitos. Then, connect the Linux PC to the local WiFi network provided by the router. Finally, start Yakuake application.

The Windows PC can be accessed from the Linux PC using the Remmina application. A direct access to this application is located in the Desktop of the Linux PC.

2.2 Turning the robot off

1. Power down the Windows PC from the Remmina application.

2. Power down the Linux PC.
3. Switch off the key.

2.3 Capturing the map of the environment

The first time that the robot is going to be deployed in a new place, it is necessary to build a map of the environment. This will be addressed using MIRA and the CogniDrive application from MLAB. Briefly, the following steps have to be followed (please refer to MetraLabs documentation about use of MIRA and CogniDrive for more detailed descriptions[1]).

+ **Opening the MIRA mapping application:** run the following command in a shell:

```
Miracenter SCITOSConfigs:etc/SCITOS-mapping.xml
```

+ **Setting the origin of coordinates:** the initial position of the robot will be stored in MIRA as the origin of coordinates of the environment. It is **recommended to mark this initial position on the floor**, because it will be used many times to localize the robot, as it will be explained later in this manual

+ **Recording a map:** in the MIRA top menu, click the "Window" menu and then the "Show view" tag. Select the "Simple Mapper" view. Click on the "Record" button and move the robot manually around the place to allow MIRA catching the information to build the map. The robot can be moved by pushing it, using the keyboard arrows, or the arrows of MIRA application (to do it, you must put the focus on the arrows section of MIRA by clicking there). The movement of the robot around the place must to finish at the same position where it started (the initial position). After that, you can stop the recording by clicking the "Finish" button of the Simple Mapper.

N When the robot is moved to record the map, please, be careful not step in front of the robot, to avoid your legs to be mapped as obstacles!!

+ **Saving the map:** the result of the previous step is a map that must be saved to use it later. You have to save it as a MCF file using the "Save MCF" button in the "Simple mapper" view.

2.4 Editing the map of the environment

Once the map has been built, you can edit it for erasing noise and adding areas of NOGO (an area which the robot must no enter). Please refer to MetraLabs documentation about use of MIRA and CogniDrive to edit maps[2].

[1] https://www.mira-project.org/MIRA-doc/toolboxes/MapBuilder/MappingIntroduction.html, https://www.mira-project.org/MIRA-doc/toolboxes/MapBuilder/SimpleMapperPage.html
[2] https://www.mira-project.org/MIRA-doc/toolboxes/MapBuilder/SimpleMapEditorPage.html, https://www.mira-project.org/MIRA-doc/toolboxes/MapBuilder/MCFReference.html

2.5 Localizing the robot in the map

When a new map of the environment is built, it is necessary to localize the robot in that map as a previous step to define the goal positions for the tests. This step is also necessary when an error in the localization of the robot is detected. Localization is achieved using the steps detailed below:

Note: A localization error can occur if the robot is moved by hand by pushing it. Because of that, it is recommended don't move the robot by pushing it. It can be moved using the keyboard or the arrows in the MIRA application.

1. The first step is to turn on MIRA using the previously built map. The easiest way to do it is making a shell script (.sh file) such as:

```
#!/bin/bash

cd ~

source .bashrc

miracenter MiraNavigation:etc/SCITOS-application.xml MiraNavigation:etc/MiraNavigation.xml -
v MCFFile=<map_name>.mcf -p 123
```

   Example scripts are provided in the robot, so a new script can be easily generated just changing the name of the loaded map.

2. When MIRA starts and load the map, we could see the position of the origin of coordinates (the initial position of the robot in the mapping process) marked with a big coordinate axes, and the position estimated by MIRA for the robot, marked with a small coordinate axes (Figure 1).

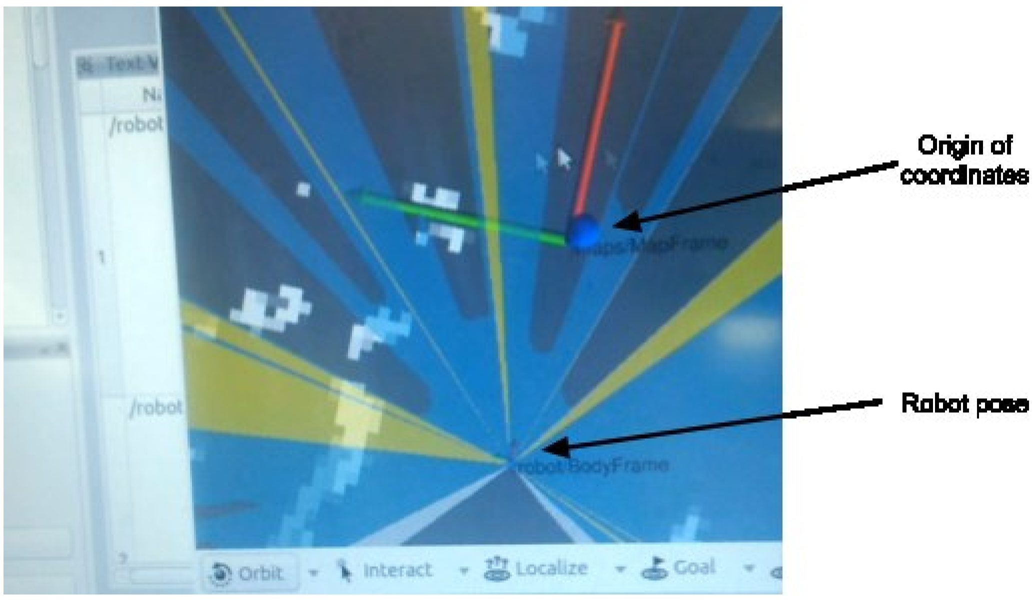


Figure 1: Detail of the origin of coordinates and robot pose in the MIRA itnterface

3. The easiest way to localize the robot is to physically place it in the environment position corresponding to the origin of coordinates, and then, set in MIRA that the robot is in the origin of coordinates. To do it, first, in the “Text View” of MIRA look at the “Pose” of the robot odometry. If this pose is different from (0.0,0.0,0.0), click the Reset Odometry button. After that, click in the “localize” button and then, in the origin of coordinates of the map to make a correspondence between the axes of the origin of coordinates and the axes of the robot. Now the robot is localized in the origin of coordinates. After that, it is good to make the robot spin to improve the localization of the robot in the map. Remember, for moving the robot you can use the arrows of the keyboard or the arrows of MIRA application (to do it, you must put the focus on the arrows section of MIRA by clicking there).

Note: It is important, when you turn on the robot, to always check that the robot is well localized. You can do it by starting MIRA using the built map and visually check in the map that the robot is well located with respect to the origin of coordinates.

2.6 Setting goal poses for Barthel and Get Up & Go tests

The first time that the robot is going to be used in a new place, and after the localization step at 2.1.3, you must select the poses (position and orientation) on the map where you want that the robot be for doing the tests. You have to store these positions in two places: in the robot components and in the CGAMed database.

There are three types of goal poses to be defined in the CLARC framework. They are listed in the Table below. The setting of the base_station pose will be described below.

Goal pose	Explanation
getu-pandgo_test	The position on the room from which the robot captures the Get Up & Go test (observing how the patient performs the test)
base_sta-tion	The position from which the robot can autonomously access to the Charging station
habitacion_x	With x ranging from 1 to N, you can define different rooms in the environment. The robot goes to these positions for performing a Barthel test or introducing a Get Up & Go one.

2.6.1 Setting goal poses (CLARA robot)

1. Create a .txt file named “goalPositions.txt” in the cajasvaciasechord/etc folder of the Linux based PC. The structure of this file is the following:

```
getupandgo_test -1.0 0.0 0.0

base_station 0.0 0.0 0.0

habitacion_1 -1.0 0.0 0.0
```

```
habitacion_2 -1.0 0.0 0.0
habitacion_3 -1.0 0.0 0.0
```

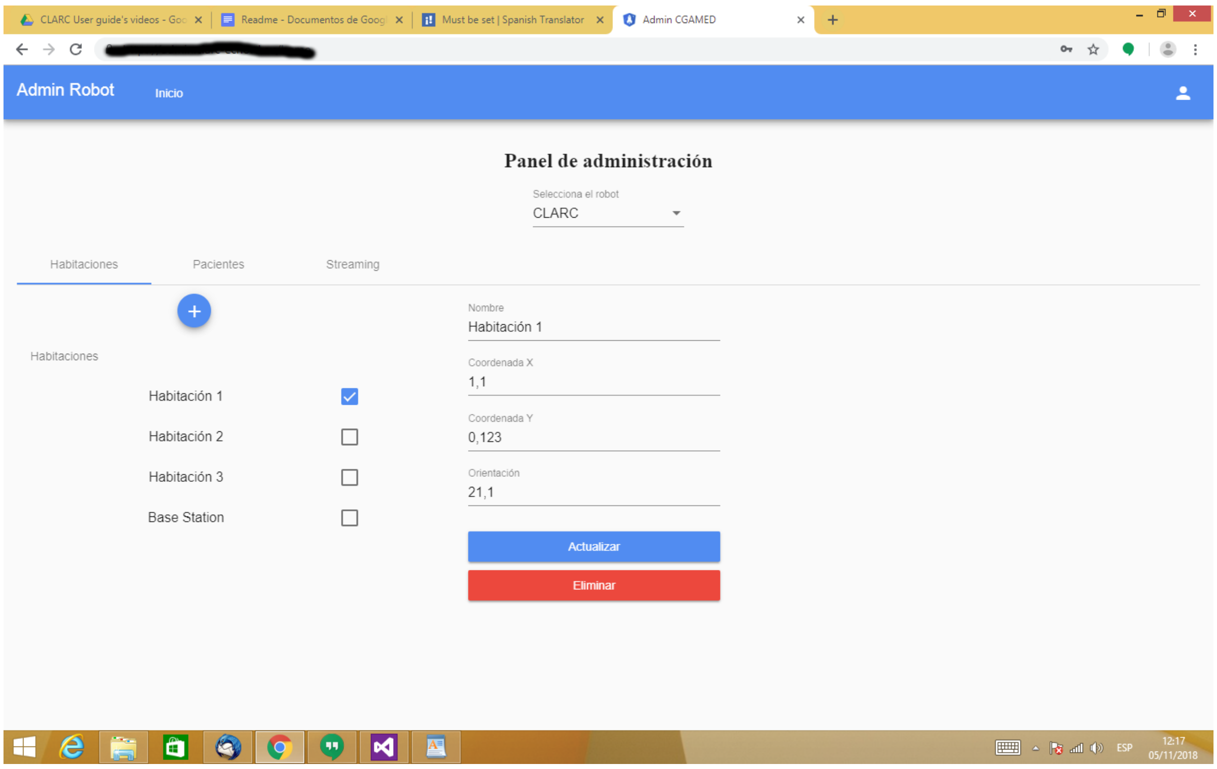
The positions of all goal poses are in the form (X,Y,angle in radians). Please, use the same labels that in this example (you can adapt the number of rooms (habitacion_x) to your environment).

2. Start MIRA with your map.
3. Reset the odometry and localize the robot.
4. Move the robot to the desired position using the MIRA arrows or the keyboard.
5. Look at the robot odometry pose in the text view of MIRA, there you have the X, Y coordinates of the position and also the angle. Be careful, because the angle in MIRA is in degrees and you have to convert it into radians.
6. Write the positions in the. txt file.

2.6.2 Setting goal poses (CGAmed)

1. To configure the positions of the rooms and the base station in the CGAMed you need to access to the administration web (see Section below).

   It is important to highlight that you do not need to write the getupandgo_test pose on the CGAmed.



CLARC User guide's videos - Goo | Readme - Documentos de Googl | Must be set | Spanish Translator | Admin CGAMED

[redacted]

Admin Robot   Inicio

Panel de administración

Selecciona el robot: CLARC

Habitaciones | Pacientes | Streaming

Habitaciones

Habitación 1	☑
Habitación 2	☐
Habitación 3	☐
Base Station	☐

Nombre: Habitación 1

Coordenada X: 1,1

Coordenada Y: 0,123

Orientación: 21,1

Figure 2: Goal pose definition using the CGAmed

2. Insert in the correct fields the same X, Y and angle values that in the “goalPositions.txt” file.

2.6.3 Setting the goal pose for the charging station

1. Make sure you have a good map of the environment, which also includes the charging station.

2. Start "miracenter SCITOSConfigs:etc/SCITOS-application.xml" and localise the robot correctly (the localisation accuracy must be fine for teaching the charging station).
3. Push the robot onto the charging station, and make sure that it is firmly and centrally on the charging station, and most importantly, make sure that the robot charges! (Yellow LED on the charging station)
4. Use a 3D view with /maps/static/Map and /robot/frontLaser/Laser visualized, and make sure you can see the area around the charging station in the 3D view (i.e. where the laser is).
5. Use the "Station tool" (in the bottom bar of the 3D view) to create a new charging station (remember which ID you give it. We typically just use 0). Leave all the parameters (except for the ID) as they are set by default, then press "Teach".
6. You will now have to mark the area around the docking station in the 3D view using the mouse. Each left-click adds a corner of a polygon. Create a polygon around the outline of the charging station in the laser, and try to include all static and characteristic features of the environment that are in the immediate vicinity of the charging station. E.g. if the charging station is next to a corner, make sure the polygon includes the corner as well, as this will help the robot localise itself correctly when docking on.
7. When you are satisfied with the polygon, finish with a right-clíck. The robot will now start driving backwards and stop four times to record laser templates at different distances. After that he should tell you that he is finished, but I'm not sure. Either way, as soon as he stops for good, he is finished, and you can now dock on to that docking station from the point where the robot is standing right now (this is called the "base point").
8. The docking station will be saved in a file "stations.xml", which needs to be in the directory where you start MIRA from. You might have to exchange "SCITOS-Pilot.xml" for "SCITOS-application.xml" in your startup scripts, as only SCITOS-application.xml includes the docking stuff.

You can read more about the general process and how to dock on in C++ here: http://www.mira-project.org/MIRA-doc/domains/navigation/DockingProcess/index.html

From a procedural point of view, you'll have to drive to the "base point" of the station first using regular navigation. The robot will have to be located in front of the charging station roughly the same as when he finished recording the templates. Only then can you let the DockingProcess dock on to this station. The “base point” is the pose that must be set as base_station (converting the degrees of the angle into radians) in the goalPositions.txt file and the CGAmed tool.

2.7 Updating the map in the start.sh script file

The first time that the robot is going to be used in a new place, and after the map of the environment has been built, the script file in charge of start all the components of the robot (start.sh) must be updated with the new map so that the robot knows its position during the sessions.

1. Within the Linux based PC in the CLARA robot, edit the start.sh script, which is in the “robocomp_clarc/robocomp/components/cajasvaciasechord/” folder.
2. Replace the name of the MCF file in the code line

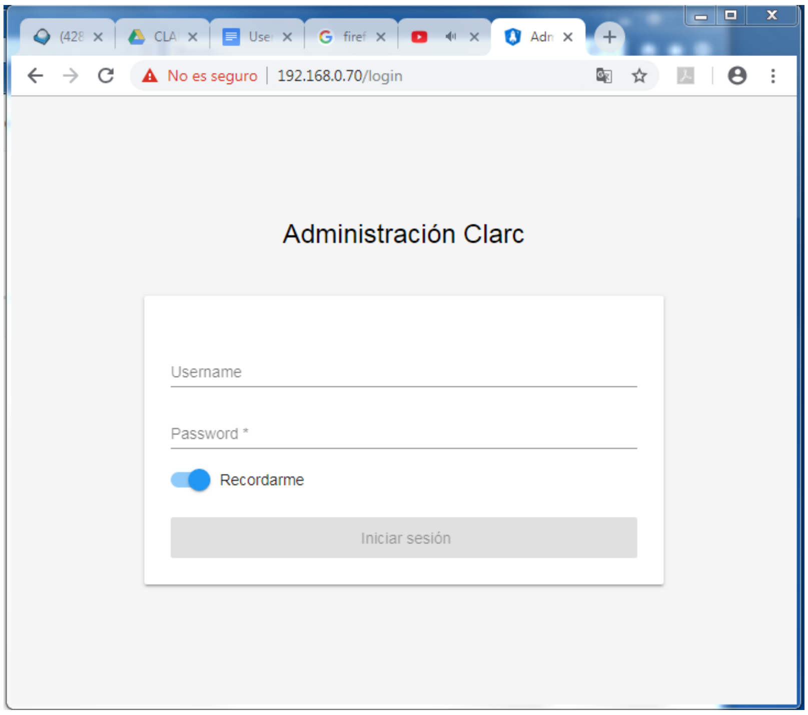
```
qdbus org.kde.yakuake /yakuake/sessions org.kde.yakuake.runCommand 'cd ~ && source
.bashrc && mira MiraNavigation:etc/SCITOS-application.xml MiraNavigation:etc/MiraNa-
vigation.xml -v MCFFile=labPhaseTwo.mcf -p 1234'
```

with the name of your MCF map.

3 Administration facilities in the CGAmed

3.1 Login in the Administration web

When you connect to the URL of the Administration web (http://192.168.0.70), you need to add user and password for entering on the web. This data are provided in Section 1.3.


No es seguro | 192.168.0.70/login

Administración Clarc

Username

Password *

Recordarme

Iniciar sesión

Figure 3: Login screen of the administration web

The administration web runs currently only in Spanish.

3.2 Managing the list of goal poses

Once logged into the Administration web, clicking on the **Habitaciones** tab you have access to the list of goal poses (rooms and base_station).

+ Clicking on the + symbol you can add new poses. When coordinates and angle are added, you should click on the **Añadir habitación** tab.

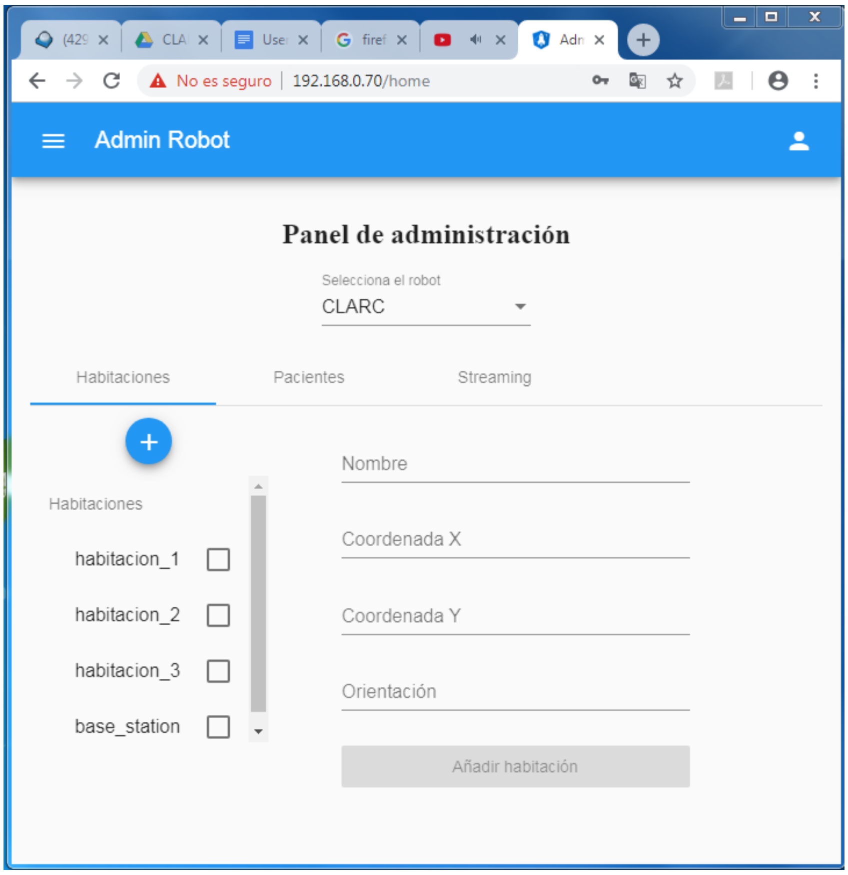
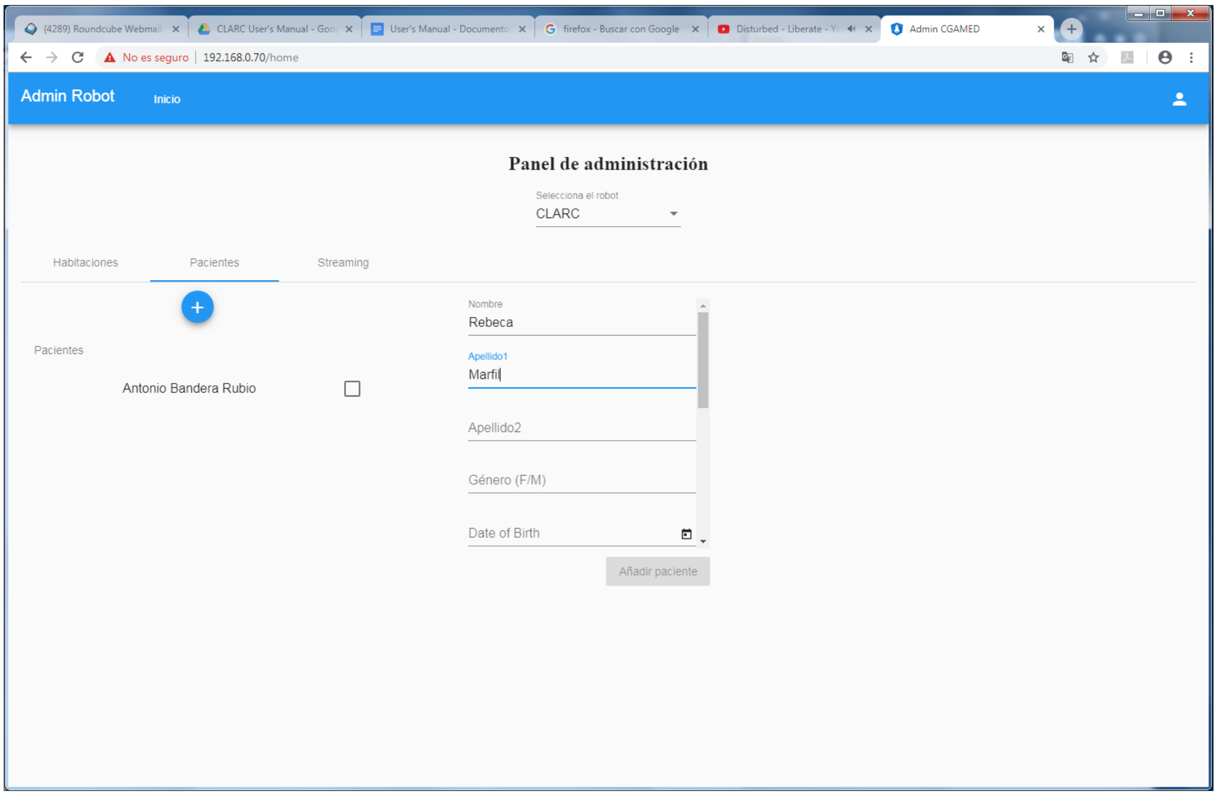


Figure 4: Goals screen of the administration web

3.3 Managing the list of patients

Once logged into the Administration web, clicking on the **Pacientes** tab you have access to the list of patients.

- Clicking on the + symbol you can add new patients. When all data about the patient has been added, you should click on the **Añadir paciente** tab. The system won't allow saving a patient in which there are empty fields, so you have to fill all of them.



Panel de administración

Selecciona el robot: CLARC

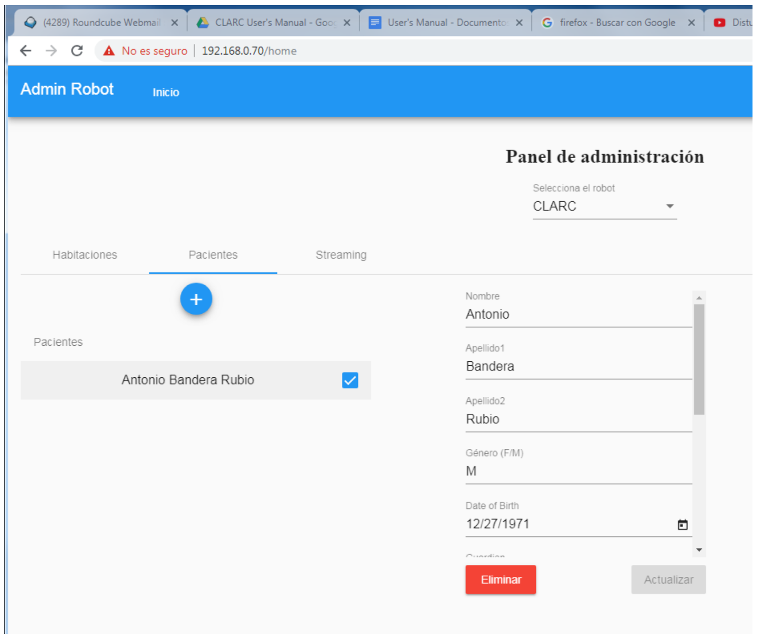
Pacientes

Antonio Bandera Rubio ☐

Nombre: Rebeca
Apellido1: Marfil
Apellido2
Género (F/M)
Date of Birth

Figure 5: Patients screen of the administration web

+ Clicking on the square-shaped box inline with the patient's name on the list, it is possible to edit the data stored about the patient. Clicking on the **Eliminar** tab is also possible to remove her/him from the list.



Panel de administración

Selecciona el robot: CLARC

Pacientes

Antonio Bandera Rubio ☑

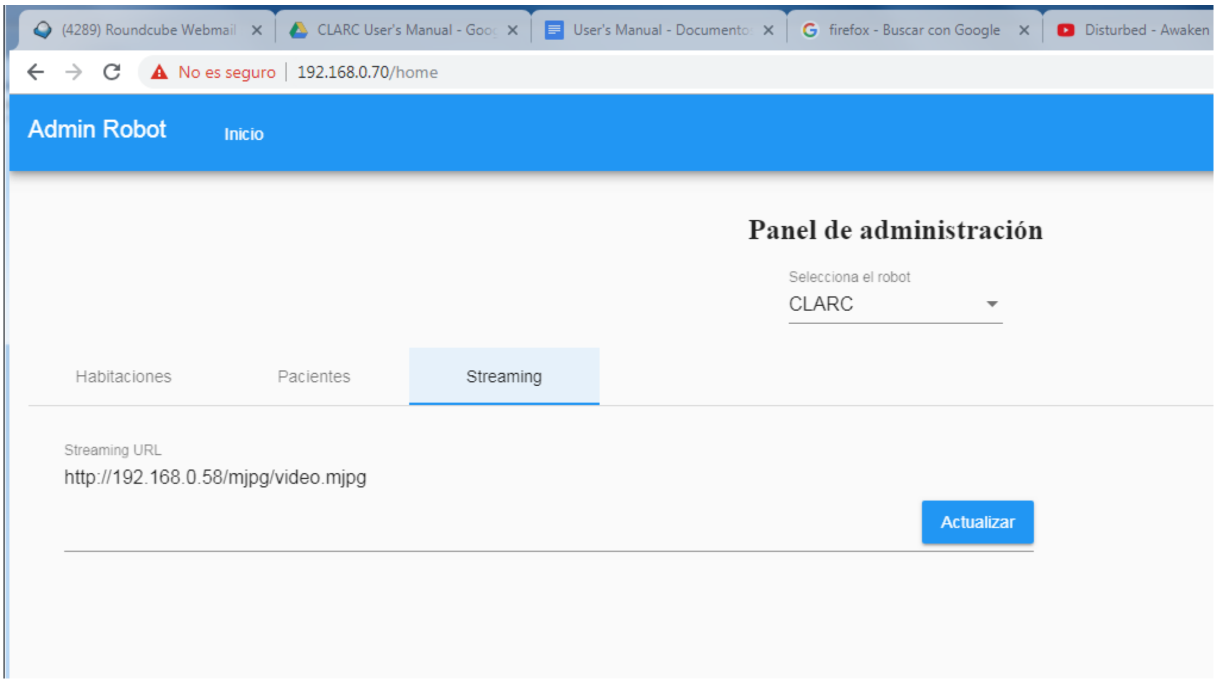
Nombre: Antonio
Apellido1: Bandera
Apellido2: Rubio
Género (F/M): M
Date of Birth: 12/27/1971

Figure 6: Patient deletion

3.4 Video streaming

The CGAmed web offers the physician the possibility of monitoring, through video streaming, the CGA session. This video streaming is provided by a IP camera mounted on the CLARA robot.

To configure this camera, once logged into the Administration web, clicking on the **Streaming** tab you can update the URL address of the IP camera. If you change the URL, click on the **Actualizar** tab for approve the update.



Panel de administración

Selecciona el robot: CLARC

Streaming URL: http://192.168.0.58/mjpg/video.mjpg

Figure 7: IP camera configuration screen

4 Data handling description

This Section describes how CLARA robot captures data, processes them to provide meaningful outputs, and shows these data to the user (clinician or technician). It also describes how data captured by the CLARA robot can be extracted for their evaluation. The information included in this document is restricted to the pilots at the end of Phase III of the CLARC-ECHORD++ project.

4.1 Overview

Figure 8 shows how the data is managed within the CLARC framework.

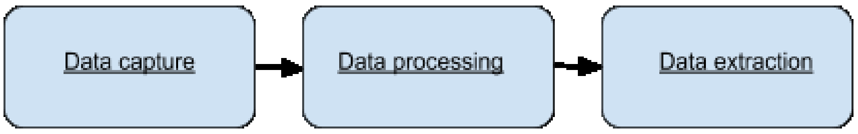


Figure 8: Scheme for handling data within CLARC

Figure 8 shows three major stages on the data capture and analysis flow:

+ **Data capture:** Within the CLARC framework, all the information about the session is captured by the CLARA robot.
+ **Data processing:** All data captured by CLARA robot are processed in order to extract relevant information from raw sources. This information is stored in the cognitive architecture of the robot, and used to extract test results. These results, including additional data such as the videos recorded during the test, are

sent to the CGAmed server once each test finishes. Briefly, each session is encoded in a set of files.

+ **Data extraction:** The responsible of supervising *in situ* how the robot drives the session will be also in charge of updating encrypted versions of all these files in an internal repository. CLARA incorporates tools to ease this process of extracting relevant data.

4.2 Data capture and storing

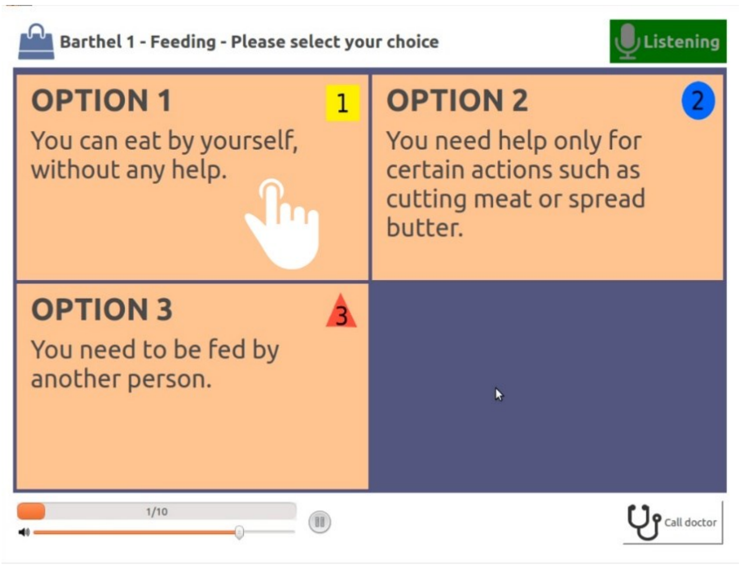
4.2.1 Barthel test

CLARA has three different interfaces which allow her capturing the required information to correctly perform the Barthel tests:

+ Voice interface: This interface allows the patient to answer the Barthel test questions using her/his voice in a natural way. It uses Speech Recognition (SR) techniques to recognize the answer provided by the patient. To do that, the answers must be included in a grammar used by the SR algorithm. If the patient answers the questions using words or sentences that are not included in the grammar, the answers are not recognized. Hence, it is important to note that all dialogue must be conducted using the Language specified on the CGAmed tool when the session was scheduled.

  The voice is captured using a shotgun directional microphone to filter sound sources apart from the speaker in front of the robot. However, in noisy environments, this voice interface still may not work properly.

+ Touch screen interface: This interface allows the patient to answer the questions of the Barthel test by touching the touch screen in the torso of the robot. This interface may be difficult to use for some people, specially elderly people, due to the uncomfortable position in which the arm has to be placed and maintained. Besides, the patient needs to get close to the robot to use this interface. If the patient maintains this position for a while, CLARA stops seeing him/her and it may stop the session because the patient is lost for it.


Barthel 1 - Feeding - Please select your choice

OPTION 1 — You can eat by yourself, without any help.

OPTION 2 — You need help only for certain actions such as cutting meat or spread butter.

OPTION 3 — You need to be fed by another person.

Figure 9: Example of interface shown in the touch screen

+ Remote control interface: the remote control (Figure 10) allows the patient to answer the Barthel questions by pushing its buttons.

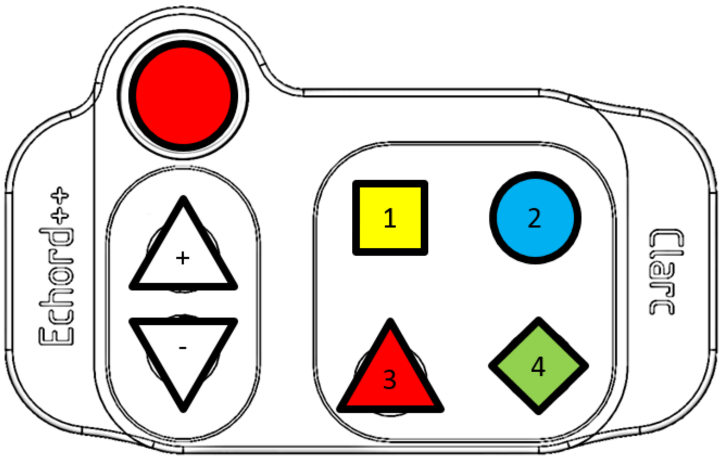


Figure 10: Schematic draw of the Remote Control

4.2.2 Get Up & Go test

To capture data from the Get Up and Go test, the robot uses the Kinect v2 sensor from Microsoft, which provides the skeleton and joint information of the patient. Kinect sensor works properly up to 4.5 meters of distance.

4.2.3 Additional sources of information

Along with the necessary information required to evaluate the tests, CLARA uses a webcam to record the patients while they are performing the tests. The video recording of each test is sent to the CGAMed database just after finishing it. The clinicians can check these videos through the CGAmed web interface.

4.3 Automated data processing & analysis

4.3.1 Answers recognition for the Barthel test

The user provides answers for the Barthel test using three different interfaces: voice, the tactile screen mounted on the robot, or the remote control. These interfaces activate when the user is requested to provide a response, and the first interface registering an input sets the answer of the user for that question.

The components in charge of these behaviours into the cognitive architecture are currently working fine, and we don't expect any change for them in the future.

4.3.2 Gait analysis for the Get Up & Go test

The Get Up & Go test is automatically evaluated by the robot following a parametric Human Motion Analysis approach, in which the complete gait of the person performing the test is firstly divided into a set of actions. Then, the algorithm evaluates each action separately, and it finally combines obtained results to provide an integrated score for the complete motion. Expert knowledge is used to perform the motion segmentation and evaluation processes.

The employed algorithm is able to select the actions in which a motion will be segmented from a library of actions. This approach allows the medical specialist to (i) create new motion exercises by selecting different actions from the library; or (ii) autonomously search for particular actions in a perceived motion. In the particular case of the Get Up & Go test, the complete detected motion will be segmented into these sequential actions:

+ Seated
+ Standing up
+ Standing
+ Walking straight
+ Turning
+ Walking straight
+ Seating

Once the gait of the person performing the Get Up & Go test has been recorded, the analysis algorithm searches for these actions in the gait. For each of the actions, the starting and ending times are detected following actions order (i.e. the ‘Standing Up’ starting point will be looked forward from the point in which the ‘Seated’ action ends).

Segmented actions are then evaluated separately. Evaluation is based on expert knowledge: the algorithm detects different clues, defined by physiotherapists, that may increase the risk of falling, and provide a 0/10 score for the action depending on those clues. These clues are extracted from different parameters, such as time employed to perform certain parts of the action, spine angle, number of steps, hip trajectories, etc.

The total elapsed time for the action is also stored. Hence, being C({ai})the set of actions of the gait (1<i<N, being N the number of actions to be detected in the gait, N = 7 in our case), for each action aia score si, an action starting time tis, and an action ending time tif, are obtained.

Once individual actions have been evaluated, the total score s_T and the motion total time t_T are computed as follows:

$$s_T = \frac{\sum_{i=1}^{N}(w_i \cdot s_i)}{\sum_{i=1}^{N} w_i}$$

$$t_T = max_i(t_i^f) - min(t_i^s)$$

where w_i are a priori weights also set via the empirical assessment of human experts.

In this situation, all actions contribute to the score and the total time. It is possible to exclude certain actions from the computation of s_T and t_T (i.e. the action is detected but its score is not considered to compute total score, or its elapsed time is not considered to compute motion total time). In the Get Up & Go use case, the first action -’Seated’- is excluded from the computation of s_T and t_T. The rest of actions contribute uniformly to s_T .

Time Up And Go

It is interesting to highlight that the Get up & Go test is usually replaced by a simpler test, named Time Up And Go. This test is based on the analysis of the same gait, but it only considers the time employed by the person to execute the motion. The proposed algorithm, that evaluates the Get Up & Go test, also computes the motion total time t_T, hence it provides the result of the Time Up And Go test

$$t_T < 20 \text{ secs} \rightarrow \text{no risk of falling,}$$

$$t_T > 20 \text{ secs} \rightarrow \text{risk of falling}$$

Evaluation

The algorithm has been tested in real scenarios, involving senior people with no stability issues, and also patients in different conditions from the rehabilitation units of the Hospital Civil de Málaga. The following tables show the obtained results, where the Get Up & Go score is provided on a five-point scale: 1 = normal; 2 = very slightly abnormal; 3 = mildly abnormal; 4 = moderately abnormal; 5 = severely abnormal. A person with a score of 3+ is at risk for falling.

Results of the Get Up & Go test for healthy people. Scores are provided in the five-level scale of the test. System's scores are also provided in a 0-10 range for the complete gait and each action. Minimum action scores are highlighted.

Id	Time (secs)	Total Score	Action scores (0-10)						
			TOTAL	StndUp	Standing	Walk. St.	Turning	Walk. St.	Seating
1	8.6	1	**9.67**	9.74	**9.07**	9.89	10	9.35	10
2	15.7	2	**7.73**	6.31	7.41	8.84	10	8.82	**5**
3	10.1	1	**8.26**	10	7.06	9.80	10	7.73	**5**
4	9.5	1	**8.22**	**5.65**	7.41	9.69	10	6.61	10
5	11.4	1	**8.55**	10	8.77	9.79	10	7.76	**5**
6	14.9	1	**8.87**	10	**5.60**	8.80	10	8.87	10
7	9.3	1/2	**7.87**	**5**	8.88	9.70	10	8.68	5
8	9.8	1	**9.04**	10	**6.67**	8.75	10	8.84	10
9	5.6	1	**9.71**	9.96	**8.67**	9.90	10	9.78	10
10	14.3	1	**8.59**	10	8.68	8.78	10	**4.13**	10
11	6.2	1/2	**7.91**	8.99	**-**	8.89	10	9.63	10
12	12.2	1	**8.65**	10	9.39	7.67	10	**4.86**	10
13	9.2	2	**7.07**	5.60	7.40	8.37	10	6.09	**5**
14	18.1	2	**6.91**	10	7.92	7.83	10	6.16	**0**
15	10.7	2	**7.62**	10	7.18	8.85	10	**4.73**	5
16	14.6	2	**6.24**	10	7.33	7.46	5	7.67	**0**
17	14.1	1	**8.32**	10	9.43	5.85	10	9.66	**5**

Autonomous evaluation of the Get Up & Go test for patients in the rehabilitation units of Hospital Civil (Málaga). Both a physiotherapist and the proposed system have provided scores for each patient in the five-level scale. System's scores are also provided in a 0-10 range for the complete gait and each action. Minimum action scores are highlighted.

Id	Age	Gender	Physiot. Score	System Score	Time (secs)	Action scores (0-10)							Diagnosis
						TOTAL	StndUp	Standing	Walk. St.	Turning	Walk. St.	Seating	
1	61	Male	3	2	12.1	**7.28**	10	6.84	9.83	10	7.06	**0**	Knee lesion, Stroke
2	64	Female	2	2	13.1	**6.93**	7.74	8.85	7.63	**5**	7.36	**5**	Stroke
3	47	Male	2	2	13.2	**7.85**	7.38	8.35	7.90	10	8.51	**5**	Cervical myelopathy
4	68	Female	1	1	9.3	**9.21**	10	10	**7.51**	10	7.74	10	Disc herniation
6	72	Male	1	2	13.8	**7.16**	7.74	8.19	9.59	**5**	7.41	**5**	Cauda equina
7	82	Female	4	3	27.9	**5.62**	10	6.6	9.63	**0**	7.5	**0**	Elbow fracture
8	47	Female	1	2	9.8	**7.42**	**4.75**	7.39	9.73	10	7.67	**5**	-
9	67	Female	1	1	7.1	**9.33**	10	8.35	9.87	10	7.76	10	Left arm issues
11	37	Male	1	2	11.4	**7.62**	9.91	7.44	9.59	**5**	8.83	**5**	Meniscus lesion
12	62	Female	2	2	13.9	**7.87**	10	8.25	9.33	**5**	9.67	**5**	Osteoporosis
15	75	Female	2	2	13.5	**6.89**	**2.75**	-	9.91	10	8.7	10	Fibromyalgia
17	62	Male	1	1	7.4	**8.36**	9.99	7.85	9.91	10	7.42	**5**	Osteoporosis
19	85	Male	1	1	11	**8.39**	10	7.82	9.81	10	7.71	**5**	-

Regarding the Get Up & Go score, the results show that for healthy people the algorithm correctly provided a 'no risk of falling' result for all performers (scores 1 or 2). On the other hand, for the patients there were only one case (patient ID #1) in which the system should have detected a certain risk of falling but it doesn't. The results for the Time Up And Go test (motion total time), on the other hand, were accurate for all performances.

Analysis of the results and further improvement actions

Results show that the proposed HMA system is able to correctly evaluate human motion. The system requires the complete gait to be perceived before evaluating it, but once the gait is captured the analytic nature of the algorithm allows producing fast responses. The algorithm is autonomous and it does not impose any constraints on the performer nor the environment. Experiments have involved successful autonomous evaluation of human gait in the Get Up & Go test.

The splitting approach seems correct to evaluate human motion. The use of modular actions to represent a complete motion facilitates generalization and adaptability for different scenarios. Encoding these actions requires expert knowledge to manually tune their conditions and evaluation functions. While this is a drawback if this knowledge is difficult to encode, it also offers a high degree of control over the evaluation criteria.

The proposal, based on sequential detection, is not robust against errors that affect one of the actions: if an action is not correctly detected, it can invalidate the complete gait analysis. That’s not a drawback, but a positive feature, for the considered use case, where the system has to provide a robust score for a clinical test. However, this issue should be addressed if the algorithm has to be adapted for more general activity recognition and evaluation scenarios.

Regarding the evaluation system, further expert assessment may modify evaluation criteria: for the Get Up & Go test, the total score for the gait is obtained by averaging action scores. But a very low value in a particular action may indicate a high risk of falling, even if the rest of the gait scores are good. Result tables mark the minimum action score of each gait, showing that some of them are far below the averaged, total score. A non-averaged evaluation, based on different weights or even discriminant thresholds, will most probably better suit this test. More evaluations of gaits of frail elderly people, assessed by clinicians, will be conducted in the CLARC project, to determine which actions should trigger an alert of falling risk, regardless of the rest of the gait.

Experiments involving patients have revealed so far that the proposed HMA system is able to provide robust, coherent and accurate results for the Time Up & Go test. However, the current ability of the system to evaluate the Get Up & Go test can be described as a limited success. While results are promising and mostly coherent with the ones provided by a human expert, it seems clear that any gait classified as 'abnormal' should be reviewed by a physiotherapist. On the other hand, all performances classified by the system as 'normal' (i.e. the ones achieving 1, the highest score) can reliably be associated to a person with no particular risk of falling. Hence, the current implementation of the system can be considered an interesting tool for screening and monitoring. It may not be precise enough so as to autonomously provide a definitive score for a medical test, but it can provide a rough diagnostic. In the Get Up & Go test, it allows discarding some performers as having a risk of falling, or alert an expert supervisor if any issue is detected in the gait.

4.3.3 Behaviour execution: action planning

This section describes how the data collected for the planning system during the tests is processed for three purposes: to control and monitor the execution of the entire system, to generate information of the execution of the tests for the clinicians, and to generate information of the planning system itself for the technicians.

The planning system PELEA is depicted in the Figure 11.

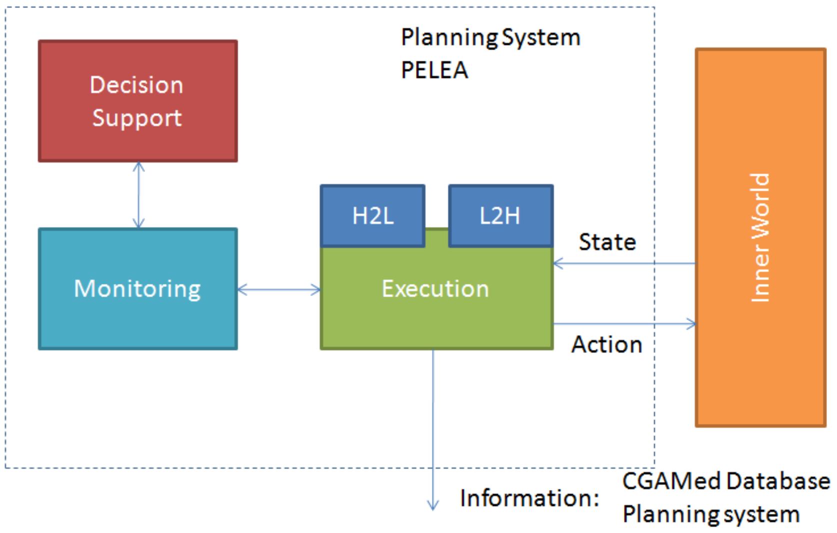


Figure 11: Schematic view of the PELEA framework

When the clinician selects the test(s) to be performed, the current state of the robot and of the environment are recovered by PELEA from the Inner World. The Inner World is represented by a graph in which the main concepts of the environment and the robot are represented (e.g., the robot has the level of battery high, the patient has answered a question). Once PELEA recovers this external information from the Inner World, it sends it to the Execution module, which in turn redirects it to the LowToHigh (L2H) module. 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 Execution. This complete high-level state is sent to Monitoring 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 (e.g., it detects the patient has not answered a question or the patient is not facing the robot), Monitoring retrieves a plan from Decision Support, which encapsulates a high-level planner. This plan is stored by Monitoring, which returns the next action to Execution module. 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. Then, Execution module transforms this high-level action into a set of low-level actions with the help of HighToLow (H2L), and inserts them into the Inner World. Other *compoNets* scanning the Inner World will notice the changes introduced by PELEA and act consequently. Then, PELEA 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 and the cycle starts again. In this scheme, the Execution module has full control of the execution, timing the maximum duration of an action, pauses, etc. to control the pace of the social interaction with the patients.

Note that during the previous execution, PELEA generates two kinds of information. The first type of information refers to the performance of the test by the patient, and the second is obtained to generate statistics on the operation of the planning system itself. The first type of information is sent by PELEA to the **CGAMed database**, while the second type is stored in the robot. In the following paragraphs, we describe these two types of information.

Information generated by the planning system for the CGAMed

This information refers to the performance of the test by the patient. In particular, PELEA collects the following information:

+ *Patient's answers*. For each question in the Barthel test, PELEA collects if the question has been answered, and what option has been selected. From this option PELEA computes the score of each question. The following equation is used to compute the score of each question in the Barthel test:
+ score=(numberOptions * 5) -(selectedOption * 5)

  where *numberOptions* is the total number of options of the question, and *selectedOption* is the option selected by the patient. Then, if the question has four options, the option number one has a score of 15, the option two a score of 10, the option three a score of 5, and, finally, the option four has a score of 0. Note that this way of computing the score is the common way in the Barthel test. At the end of the Barthel test, PELEA computes the total score of the test by adding the partial scores of each question.
+ *Time that the patient takes to answer.* PELEA also computes the time required for the patient to answer each question. This time is measured in seconds, from the moment in which the planning system sends the action that enables the system the reception of the response, to the moment in which the planning system detects that the response has been received by some mean: audio, screen or tablet.
+ *Video information.* PELEA is also responsible for sending the actions to start/finish the recording of the sessions.
+ *Control information.* Finally, PELEA also collects control information on the status of a particular session, i.e., start/end time of the session, whether the clinician stops/resumes the session, if the robot does not detect the patient, etc.

Once the tests finishes, PELEA sends all this information to the **CGAMed database**. Such information is displayed by the CGAMed web server to the clinicians.

Information of the planning system itself

In addition to the previous information, PELEA also generates information on the execution of the planning system itself. In particular, this information refers to:

+ *Number of actions*. PELEA collects information on the number of actions required to perform each tests. From an Automated Planning point of view, this number refers to the cost of the plan.
+ *Starting time and duration of each action.* The starting time and the duration for each action is also stored. The duration is computed from the moment in which the action is *written* in the inner world, to the moment in which the planning system detects in the inner world that the action has been correctly fulfilled.
+ *Number of replanning situations.* PELEA also collects information on the number of replanning situations, i.e., on the number of times the high-level planner is invoked. Each time the *perceived* situation in the environment is not equal to the *desired* situation, PELEA replans.
+ *Duration of the total plan.* The duration of the total plan required to perform each of the tests.
+ *Log information of the PELEA modules.* Finally, the execution traces of the different modules of the planning system (*Execution, Decision Support* and *Monitoring*) are also stored. It is important to be aware of the fact that the previous information (number of actions, starting time and duration of each action, number of replanning situations) is generated by the analysis of these execution traces as shown in Figure 12.

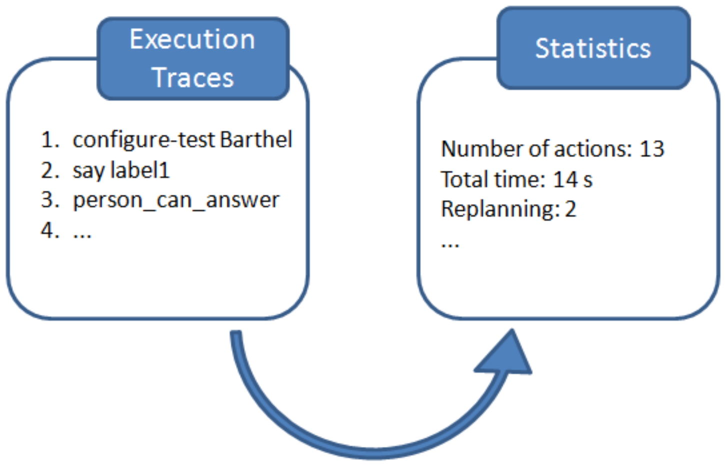


Figure 12: Analysis of execution traces in PELEA

For each session, a folder with the name session_YYYY_MM_DD_HH_mm_ss is created within the folder output of the planning system. In that folder, the previous information is stored. The purpose of all this information is to collect statistics about the execution of the planning system. Such statistics are analyzed by technicians in order to evaluate the performance of the system.

Evaluation

It should be interested to report here whether the planning capabilities of the system supports and/or affect to the human-robot interaction. The experiments in this section has been conducted in Seville, Spain, in a retirement home the 7th of November, 2017. The following table present the statistics collected from PELEA in two different tests, Barthel and GetUp&Go, in that retirement home.

Test	Duration (s.)	Cost	Replan	Planning (s.)	Response (ms.)
Barthel	716.5 64.5	102 5.3	5.8 3.3	0.33 0.01	277.7 6.8
GetUp&Go	166.7 9.2	20.0 1.8	2.2 0.7	0.05 0.03	236.1 5.6

The results in the previous table are analyzed across five dimensions: the accumulated time (seconds) used to solve the test (we label as *Duration* the column in the table), the accumulated cost measured as the number of actions in the executed plan (*Cost*), the number of times it is necessary to invoke the high-level planner (*Replan*), the average time (seconds) spent by the high-level planner to build a plan (Planning), and the average response time (milliseconds) per action (*Response*). The previous table shows the means and standard deviations computed from five different patients for each domain and test.

Patients interacted with CLARC robot a mean of 716.5 seconds to complete the Barthel Test and 166.7 seconds (dimension Duration in Table). The standard deviation is 64.5 seconds for the Barthel test, and 9.2 for the GetUp&Go. Thus, we demonstrate that the planning system is able to manage large social interactions without human intervention. The mean number of actions required to solve the Barthel test is of 102 5.3 and for the GetUp&Go test 20.0 1.8. Therefore , the Barthel test requires at around five times more actions than the GetUp&Go test. The number of replanning situations is of 5.8 3.3 for the Barthel test and 2.2 0.7 for the GetUp&Go. Finally, the mean time required for the high-level planner to build plans is 0.33 0.01 seconds for the Barthel test and 0.05 0.03 for the GetUp&Go, and the mean response time is of 277.7 6.8 milliseconds for the Barthel test and 236.1 5.6 for the GetUp&Go. Both times are low enough to allow a correct patient-robot interaction.

Analysis of the results and further improvement actions

The previous experiments demonstrate that user interaction is fluid even when many re-planning actions are required due to the high uncertainty of the interaction. However, more evaluations will be conducted in the CLARC project in order to get a greater amount of information that allows us to obtain more conclusive results.

4.4 Extracting data from the CGAmed server and the robot

The protocol for capturing the information coming from the **Planning system** (within the software architecture CORTEX, driving the CLARA robot) and the **CGAmed database**, as well as all **videos** available, consists on the following steps:

1. Within the Linux based PC in the CGAmed server, open a term window and run:

   ```
   >> backupClarc.sh
   ```

   This script takes all the information and compress it in three files, which are saved in the folder *backupClarc.* These three files are

   + YYYY_MM_DD_HH_mm_ss_PeleaLogs.tar.gz, which contains the information of the planning system.
   + YYYY_MM_DD_HH_mm_ss_database.sql.gz, which contains the information of the CGAmed database.
   + YYYY_MM_DD_HH_mm_ss_videos.7z, which contains the videos. This encrypted file has been compressed using a password (nBRTj6JuUc6b7ZnGXqLLFqmZ).

   Once the files are saved into the aforementioned folder, the script removes the original files.
2. Go to the folder *backupClarc* and copy the previous files into a pendrive or an external hard drive.

4.5 Privacy issues - how do we handle privacy

Personal data are not stored in the robot, but in the CGAmed server. This server is a computer that controls access via username/password. It is connected only to the robot through a local network. Hence, only registered users can access the data in CGAmed server, and they can access it only through physical operation of the CGAmed server.

People extracting data from the CGAmed server and the robot are responsible of guaranteeing the privacy of these data from the moment they copy these data out of the folder *backupClarc*.