CARESSES Software Handbook

ENGLISH

USER GUIDE & TECHNICAL GUIDE

Contributors: Roberto Menicatti UNIGE
Carmine Recchiuto UNIGE

Contents

Chapter 1

User Guide

This Chapter is meant to be a guide for the User who wants to interact with the CARESSES Robot without knowing what is going on "beneath the surface".

1.1 Interaction Modes

The interaction between the User and the Robot can be schematized as in Figure 1.1, where the circles represent the four *modes* or *states* that the Robot can assume and the rectangles represent the situations which make the interaction switch from one mode to the other.

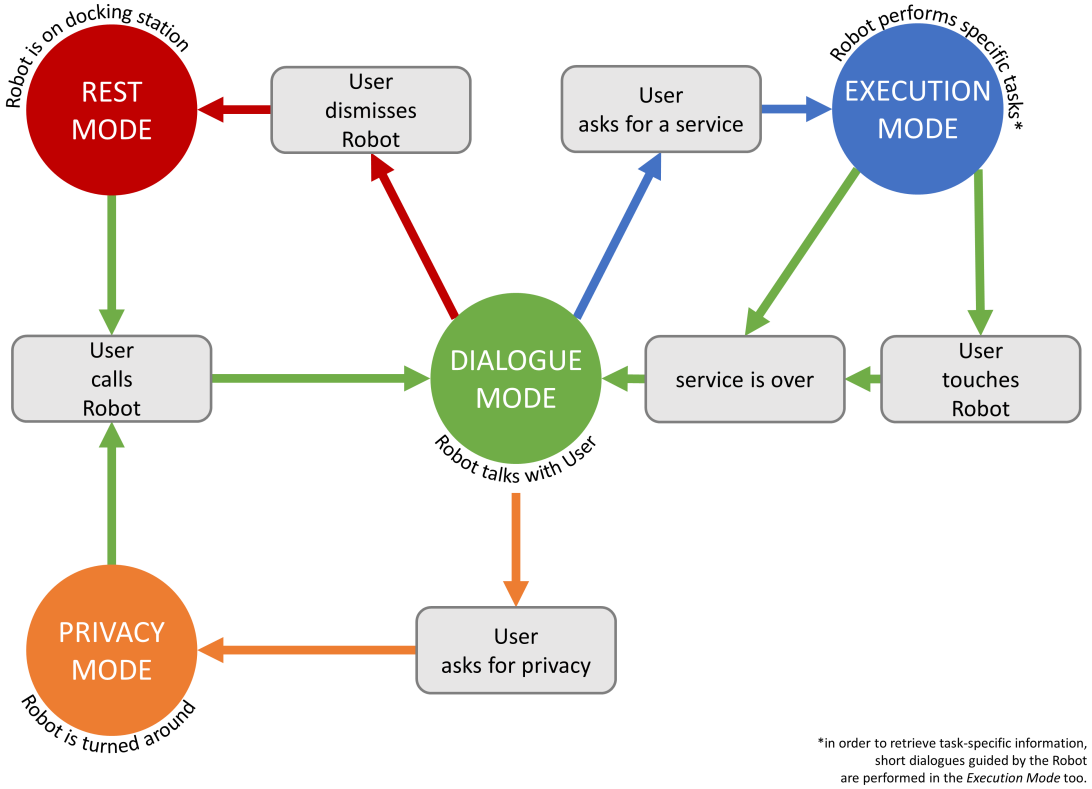
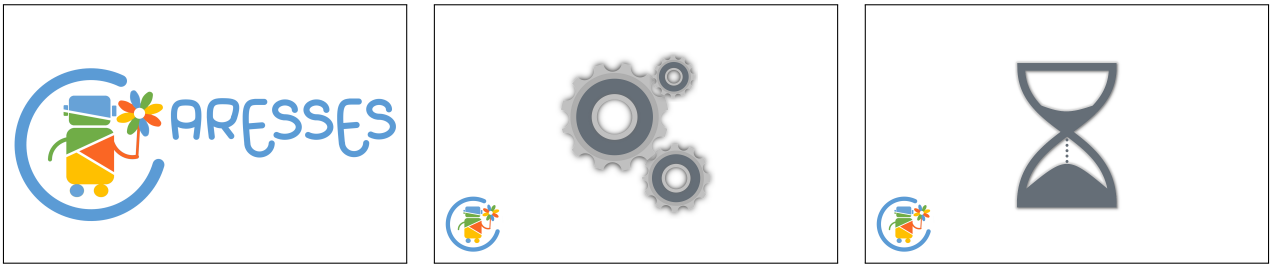


Figure 1.1: Scheme of the User-Robot interaction



(a) Dialogue Mode screen (b) Execution Mode screen (c) Rest/Privacy Mode screen

Figure 1.2: Different tablet screens

Rest. In the ***Rest*** Mode the Robot does not interact with the User. It stays in a predefined place of the environment (normally its *docking station*) waiting for the User to call its attention. If the User calls the Robot, the Robot switches to the *Dialogue* Mode.
The Robot's tablet displays Figure 1.2(c) on the screen.

Dialogue. In the ***Dialogue*** Mode the Robot verbally interacts with the User and tries to detect *triggering keywords* in what the User says. The *triggering keywords* can either activate the execution of a specific service (*Execution* mode) or lead the conversation to a specific topic. The Robot can switch back to the *Rest* Mode either if the User dismisses the Robot on purpose or if the User ignores it for a long time. The Robot can also switch to the *Privacy* Mode if the User explicitely asks for it.
The Robot's tablet displays Figure 1.2(a) on the screen.

Privacy. In the ***Privacy*** Mode the Robot turns its back on the User to give them privacy. The Robot does not interact with the User until the User calls or touches the Robot.
The Robot's tablet displays Figure 1.2(c) on the screen.

Execution. In the ***Execution*** Mode the Robot interacts with the User by executing the requested service. When the service is over, the Robot exits this state and switches to the *Dialogue* Mode. The User can force the Robot to exit the ***Execution*** Mode by touching it or using *Exit Execution-Mode keywords*.
N.B. - During the ***Execution*** Mode it is not possible to start other services or a conversation, unless by first quitting the current task.
The Robot's tablet displays Figure 1.2(b) on the screen.

1.2 How to Interact with the CARESSES Robot

1 - What can I ask to the CARESSES Robot?

You can ask to the CARESSES Robot either to:

- **chat about a topic** or to
- **execute a specific task** among the ones available.

2 - Which *topics* can the CARESSES Robot chat about?

This is a non-exaustive list of the topics you can chat about with the CARESSES Robot, just to give you some suggestions.

- Events
- Family
- Friends
- Food
- Habits
- Health
- Hobbies (books, music, movies, sport...)
- Religion

3 - Which *tasks* can the CARESSES Robot execute?

Please refer to Figure 1.5 for a complete list of the tasks that the CARESSES Robot can execute.

4 - What kind of inputs can I give to the CARESSES Robot?

You can give **five** kinds of input to the CARESSES Robot. It is **very** important to know ***why***, ***how***, and ***when*** to give these inputs. Refer to the corresponding question in brackets to know more.

- **Chat Request Input** (see question 5)
- **Task Request Input** (see question 6)
- **Open Answer Input** (see question 7)
- **Yes/No Input** (see question 8)
- **Parameter Input** (see question 9)

Be sure that the CARESSES Robot has finished talking before giving any kind of input. **While the Robot is talking, it does not listen to you and cannot be interrupted.**

5 - Why, how and when can I give a *Chat Request Input*?

- **Why:** in order to start chatting about a topic.
- **When: after a suggestion by the Robot**, such as the following ones:
  "*Please tell me if I can do something for you.* ***For example I may*** *send a message to your medical staff.*"
  "*Is there anything that I can do for you?* ***As a suggestion, I could*** *show you a video about traditional indian side dishes.*"
  Notice that **a suggestion is present**.
- **How:** simply mention the topic you want to talk about. Two example sentences to start talking about *sport* could be: "*Let's talk about* ***sport*** *now!*" or "*You know? I really like* ***sport***".

6 - Why, how and when can I give a *Task Request Input*?

- **Why:** in order to make the Robot execute a task.
- **When: after a suggestion by the Robot**, such as the following ones:
  "*Please tell me if I can do something for you.* ***For example I may*** *send a message to your medical staff.*"
  "*Is there anything that I can do for you?* ***As a suggestion, I could*** *show you a video about traditional indian side dishes.*"
  Notice that **a suggestion is present**.
- **How:** start your sentence by saying something like: "*I want to...* ***do this task***", "*I would like to...* ***do this task***" or "*Can you...* ***do this task****?*". Each task is triggered by some specific *triggering keywords*. The full list is reported in Figure 1.5, anyway they are quite intuitive. For example, in order to make the robot play some music, you could say "*I would like to* ***listen*** *to some* ***music****!*" or "*Can you* ***play*** *a* ***song*** *for me?*".
  Please notice that you don't necessarily have to give all the details of your request at this stage, the Robot will ask for other information later if necessary (see question 9). For example, you don't have to specify immediately the title of the song that you want to listen to.

7 - Why, how and when can I give an *Open Answer Input*?

- **Why:** in order to say anything you want concerning the current topic of conversation.
- **When: when invited by the Robot** to express your idea on something, for example when the Robot says:
  "*Please,* ***tell me*** *something* ***about*** *your favourite traditional indian outfit if you like*".
  Notice that **this is not a question**.
- **How:** just talk freely, e.g: "*I really like that particular outfit because it reminds me of... etc.*".
  N.B. - When you have finished talking, say one of the *Open Answer keywords* (see question 13) to let the Robot understand that it is now its turn to talk.

8 - Why, how and when can I give a *Yes/No Input*?

- **Why:** in order to answer questions about your habits and preferences or to confirm your request for executing a task.
- **When: when the Robot makes a yes/no question**, such as:
  "*May I ask you a question?* ***Do you*** *make flower arrangement sometimes?*"
  "***Do you*** *really want me to set a reminder for you?*"
  Notice that **these are yes/no questions**.
- **How:** reply clearly by saying "*yes*" or "*no*". It is better to add some words to facilitate speech-recognition for the Robot: e.g. "*yes, sure*", "*no, never*" or "*yes, please*", "*no, thanks*". In some cases, other keywords, such as "*sometimes*", "*always*", "*I do*", "*correct*", *etc.* are accepted too.

9 - Why, how and when can I give a *Parameter Input*?

- **Why:** in order to give more details to the Robot about the task you want to execute.
- **When: when the Robot makes Wh- and How invitations**, such as:
  "*Please tell me* ***what*** *book you would like me to read.*".
  "*Please tell me* ***who*** *you want me to send your message to.*"
  Notice that **these are Wh- invitations**.
- **How:** give a specific answer. The more specific is your answer, the sooner the Robot will accomplish the task: e.g. "Romeo and Juliet by William Shakespeare".

10 - Can I only give *voice* inputs or is there also another way?

Most of the times you can answer ***verbally*** to the CARESSES Robot. However, sometimes you can take advantage of the **Choice Manager** (see question 11) to be sure that the Robot understand your answer.

11 - What is the Choice Manager?

The Choice Manager is a Graphical User Interface, loaded on the Robot's tablet, which shows some or all of the possible answers that you can give to the CARESSES Robot in order to give *Yes/No Inputs* and *Parameter Inputs* (see Figure 1.3).

The answer can be given either by clicking on the corresponding button or by saying the exact text displayed on the button. When present, three *navigation buttons* on the bottom right corner of the screen will help you in showing other possible answers:

- **BACK**: navigate to the previous page;
- **NEXT**: navigate to the following page;
- **EXIT**: exit the Choice Manager and skip the question.

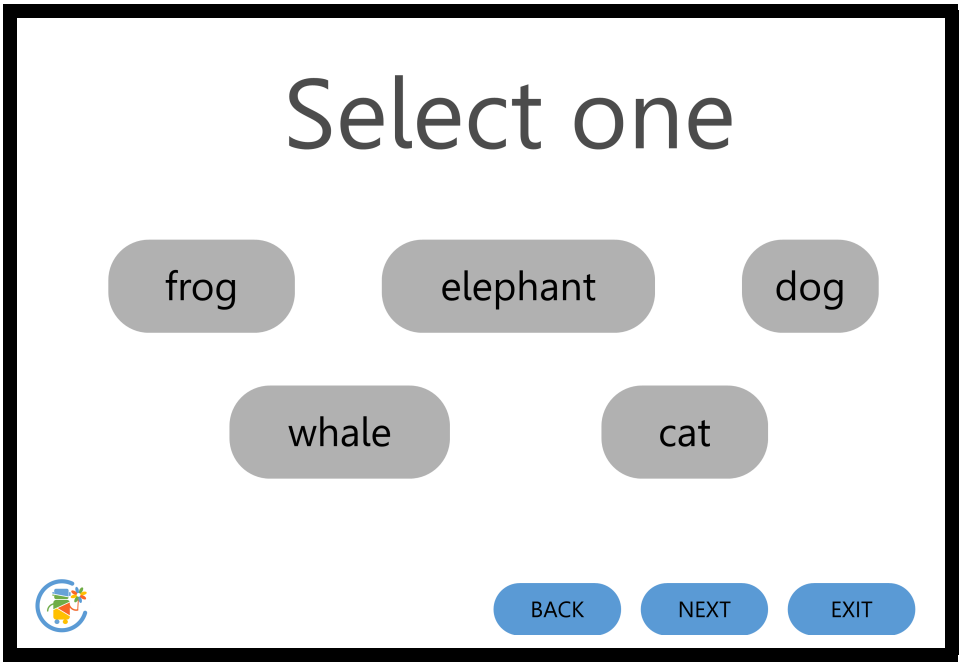


Figure 1.3: Choice Manager layout

12 - How do I launch the Choice Manager?

The Choice Manager is activated:

- **on request** by using *Options keywords* (see question 13);
- **on touch** by touching the tablet;
- **automatically** if you do not reply to the Robot's question within a certain amount of time.

13 - Is there any *keyword* I can say to ease the interaction?

Yes, while interacting with the Robot, you can use some *special keywords* (even within a more structured sentence). **Please notice that the keywords work only when the Robot is waiting for an input from the User and not when it is *doing* something** (e.g. while the Robot is talking, moving, etc.). The keywords are:

- ***Repeat keywords***: repeat, say it again
  Use these keyword to make the Robot repeat its last sentence.
- ***Revise keywords***: revise
  Use this keyword to revise your last *Yes/No Input* to a Robot question about your habits and preferences.
- ***Options keywords***: options
  Use this keyword to launch the Choice Manager.

- ***Open Answer keywords***: what about you?, what do you think?, over and out
  Use these keywords to give back the word to the Robot at the end of an Open Answer.

- ***Initiate Rest-Mode keywords***: bye, rest, sleep, charge, docking station, stop, freeze, block, see you later
  Use these keywords to stop the interaction with the Robot and switch to the *Rest Mode*.

- ***Exit Rest-Mode keywords***: hello Pepper
  Use this keyword to call the Robot, resume the interaction with it and switch from the *Rest Mode* to the *Dialogue Mode*.

- ***Initiate Privacy-Mode keywords***: turn around, privacy, provide privacy
  Use these keywords to make the Robot turn around and switch to the *Privacy Mode*.

- ***Exit Privacy-Mode keywords***: Pepper, hello, hey, you can turn, turn around, turn, look, you can look
  Use these keywords to call the Robot and switch from the *Privacy Mode* to the *Dialogue Mode*.

- ***Initiate Execution-Mode keywords***: *keywords depend on the task you want to activate, refer to Figure 1.5.*
  Use these keywords to start the execution of a task.
  N.B. - In order for the keywords to work you should say at least one keyword for each set. For example: if you just say "*music*" (keywords set 2, row 13), the Robot will simply start talking about *music*; if instead you say "*play* some *music*" (one keyword per each set, row 13), the execution of the corresponding task will be triggered.

- ***Exit Execution-Mode keywords***: I want to quit, quit, abort, exit
  Use these keywords to stop the execution of a task and switch from the *Execution Mode* to the *Dialogue Mode*.

14 - How can I stop the execution of a task?

You can stop the execution of *most* (not *all*) of the tasks in one of the following ways:

- by touching Pepper either on its head, on the back of its hands or on the bumpers (see Figure 1.4);

- by pressing the *EXIT* button on the tablet if the Choice Manager is active;

- by saying one of the *Exit Execution-Mode keywords*.

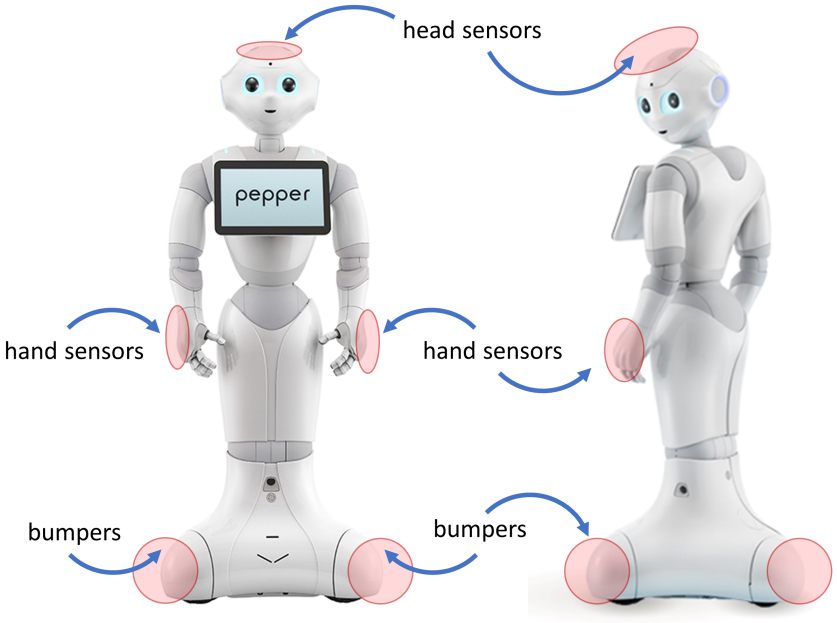


Figure 1.4: Pepper tactile sensors

	GOAL	KEYWORDS SET 1	KEYWORDS SET 2	ACTIONS INVOLVED
1	Control Devices	operate, control	device, smart, ihouse	OperateIoT
2	Date and Time	what, tell	time, date and time, day is today	TellDateTime
3	Go Somewhere	go, come, accompany, need, want, like	[any location of the room]	GoTo
4	Help in Praying	pray	pray	Chitchat
5	Help get Dressed	want, need, help, like, time, will	dress	GoTo, Chitchat
6	Help with Exercises	want, need, help, like, time, will	exercise	Chitchat
7	Help with Meal	help	meal, lunch, breakfast, dinner	Chitchat
8	Make a Call	call, phone	skype, [any contact in the contact list]	SkypeCall
9	Move Object	move, bring	[objects in the room]	GoTo, Load, Unload
10	Play Game	play	game, memory	PlayGameMemory
11	Play Karaoke	sing	karaoke, song	PlayKaraoke
12	Play Movie	play, watch, display	movie, film	PlayVideo
13	Play Music	play, listen	music, song	PlayMusicAndVideo
14	Provide Privacy	provide, need, want	privacy	PrivacyCoverEyes / PrivacyLookDown / PrivacyTurnAround
15	Read Audio-book	play, read	book, audiobook	ReadAudiobook
16	Relax	play, want	relax, relaxing music	PlayMusicAndVideo
17	Read Sensors	check, read	device, sensor, smart	ReadIoT
18	Remind Object Position	remind, where	[objects in the room]	RemindLocation
19	Send Message	send a message	message, [any contact in the contact list]	SendLineMessage / SendTelegramMessage
20	Set Reminder	set	reminder	SetReminder
21	Show Instructions	display, show	video, instructions	PlayMusicAndVideo
22	Show Menu	display, show, want to see, menu	menu, meal, breakfast, lunch, dinner, dessert	ReadMenu
23	Show Photos	display, show	photos, pictures, selfie	ShowPictures
24	Show/Read News	tell, read, show	news	ReadNews
25	Show Weather	want need, show, like, tell, display, what, rainy, sunny	weather, rain, sun, warm	DisplayWeatherReport
26	Take by Hand	take	hand	Accompany
27	Take Photos	take, send	photos, pictures, selfie	TakeAndSendPicture

Figure 1.5: List of tasks that the CARESSES Robot can execute

Chapter 2

Technical Guide

2.1 Introduction

CARESSES software consists of the following three main modules:

- **CAHRIM**: Culture-Aware Human-Robot Interaction Module;
- **CKB**: Cultural Knowledge Base;
- **CSPEM**: Culturally-Sensitive Planning & Execution Module

For the communications between the aforementioned modules two different solutions can be used:

- **uAAL**: universAAL, middleware;
- **caressOS**: CARESSES "operating system" for message exchange.

The software architecture for the two solutions is represented in Figure 2.1 and Figure 2.2. The three modules exchange messages as shown in Figure 2.3.

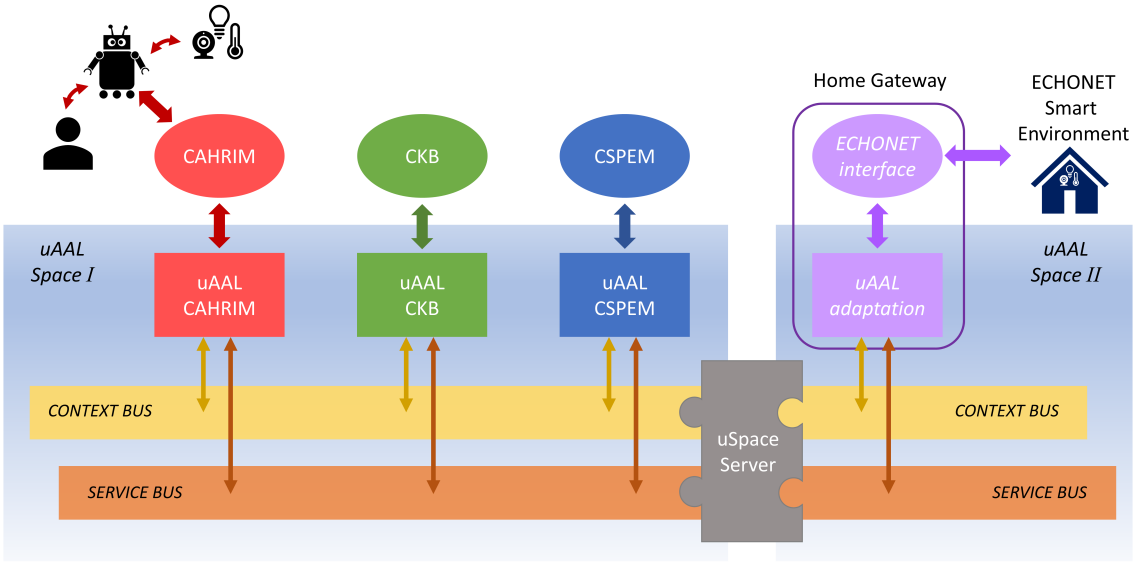


Figure 2.1: CARESSES Software Architecture for the use with uAAL

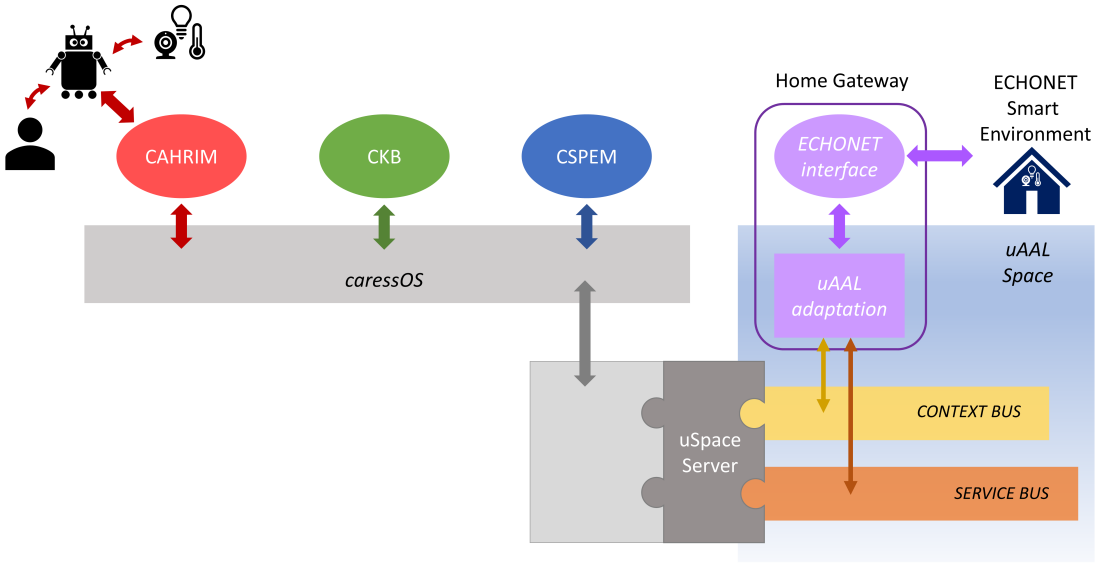


Figure 2.2: CARESSES Software Architecture for the use with caressOS

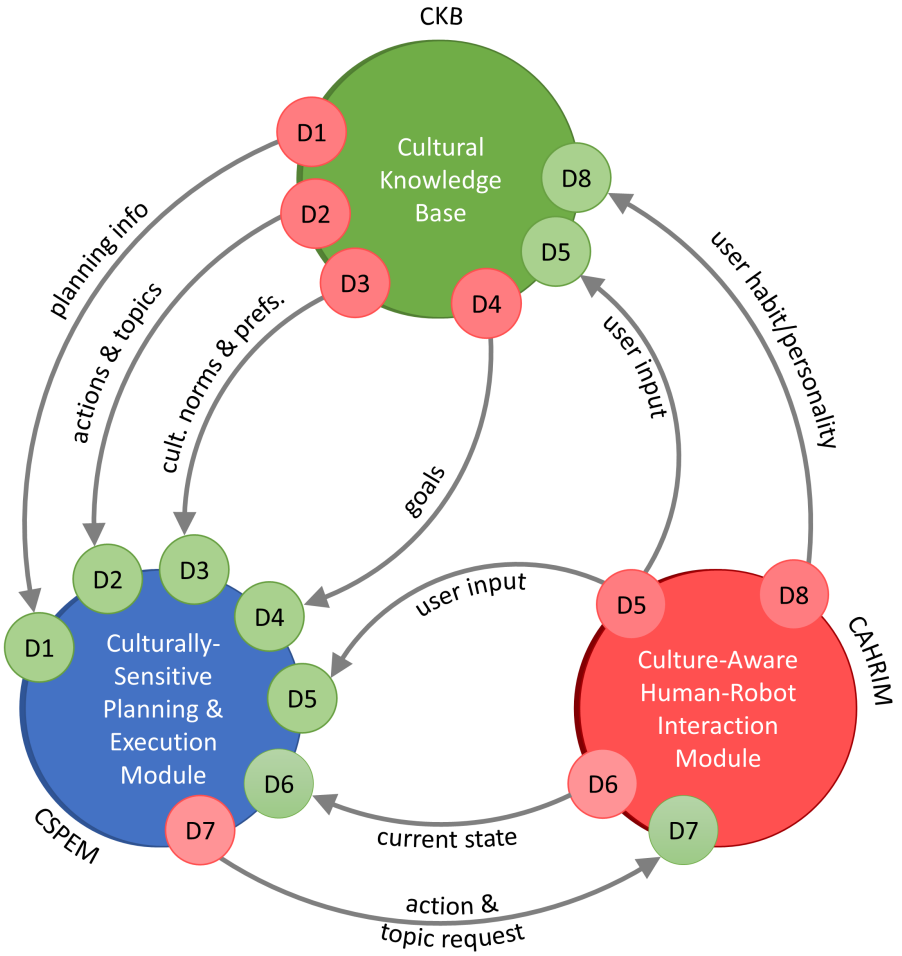


Figure 2.3: CARESSES Functional Architecture

The next Section briefly describes the working principles of the CARESSES Software, while the rest of the Chapter explains how to download, set up and run the different CARESSES components.

2.2 CARESSES Software Working Principles

At startup CKB sends to CSPEM all the planning operators and CARESSES Actions via D1 and D2 messages. The Actions are sent together with their execution parameters and sent again everytime these parameters are updated in the Knowledge Base. At startup CKB also sends to CSPEM the first goal to be achieved via a D4 message. With reference to Figure 1.1, the first goal will either start the robot in *Dialogue Mode* or *Rest Mode*. Every other goal (or most of them) are then triggered by the user through the interaction with the robot and sent to CSPEM by CAHRIM via D5 messages.

The behavior of the robot, handled by CAHRIM, is given by the series of Actions planned by CSPEM to achieve the current goal. Every time an Action must be executed, CSPEM sends a D7 message to CAHRIM specifying the Action name and its execution parameters (previously received by CKB). These parameters are divided into:

- actual parameters;
- cultural parameters.

Cultural parameters are the ones which come directly from the Knowledge Base, such as *volume*, *speed* and *pitch* of the robot's voice, *language*, *user name* and the *suggestions* for the execution of the task. **Actual parameters** are the ones directly related to the task execution and which don't come from the Knowledge Base, e.g. the title of the song to be played, the audiobook to be read. If the actual parameter is not specified by the user when requesting the task to the robot in *Dialogue Mode*, the parameter is passed with the value "n/a" (not available) and the robot will ask for it inside the Action in *Execution Mode*.

Not all the Actions accept any arbitrary input as actual parameter. For example, while the PlayMusic Action can look on the internet for any title asked by the user, the GoTo Action must restrict the choice of destinations to the set of nodes which constitute the predefined map of the environment. Therefore, CKB defines in the ontology a set of available actual parameters for each Action. Their likeliness in the ontology is constantly updated according to the cultural preferences which emerge from the chit-chat dialogue between the user and the robot. The parameters with the highest likeliness are passed to the Action among the *cultural parameters* as *suggestions* (aforementioned). All the parameters, regardless of their likeliness, are also stored in files created at startup by CKB and accessed by CAHRIM at runtime in order to retrieve more information about them (e.g. for a *contact* parameter, the additional information listed in the file may be the phone number, the email address, the Telegram ID...).

2.3 Setup

As first thing, pull from the ***master*** branch of the **caresses-opensource** repository[1] the whole CARESSES software. Then create in the **caresses-opensource** directory the file *caresses-conf.json* by following the template *caresses-conf_TEMPLATE.json* and fill up the fields accordingly.
In the "docking-station" field you must specify whether or not Pepper's docking station is being used. If in use, the docking station should occupy the charger node.
In the "input" and "output" fields you should specify the user input source and the software output device according to the numeric IDs shown in Table 2.1. Please notice that the console output is always present, regardless of which output is specified. The reason for putting 0 as value for the "output" field is to avoid completely audio output (sometimes useful for quick tests and debug).

Table 2.1: Input/Output Configuration Codes

	CONSOLE	PC	SMARTPHONE	ROBOT
	0	1	2	3
INPUT	keyboard	PC mic	phone mic	robot mic
OUTPUT	console output	PC speakers	phone speaker	robot speakers

N.B. - All the fields of the *caressesconf.json* file are compulsory except for:

- "CAHRIM-database": the sub-fields of this voice are needed only if you will be using activity recognition (see section 2.3.1.4);
- "interaction-node": if you want the interaction between the Robot and User to take place in a predefined place of the environment, fill this field with the name of the desired node of the map. If, instead, you want the interaction to take place wherever the User is, leave this field empty.

[1] https://github.com/laboratoriumDIBRIS/caresses-opensource.git

2.3.1 CAHRIM

2.3.1.1 Python

In order to run CAHRIM software you need **Python 2.7 32bit** (`https://www.python.org/`).

The following Python packages must be installed (they can be installed via **pip**): *apiai, imutils, numpy, opencv-python, opencv-contrib-python, python-dateutil, pexpect, glob, beautifulsoup4, requests, isodate, nltk, google.api.python.client, mysql (on Ubuntu: sudo apt-get install python-mysql.connector), colorama, paramiko, matplotlib, feedparser, google-cloud-translate*[2]. Please notice that **nltk** package requires the following additional steps:

- Open a Python shell:

```
$ python
```

- Then type:

```
>>> import nltk
>>> nltk.download()
```

- A window will pop-up: keep all the packages selected and click on "Download".

2.3.1.2 Pepper

- Be sure to have **NAOqi 2.5.5.5** installed on the Robot.
- Install SoftBank Robotics software on your computer, i.e. **Choregraphe 2.5.5** and the **Python SDK 2.5.5**. Follow the instructions at this link[3]. You will be asked to create a developer account.
- Install on the Robot the following Choregraphe apps. To install an application on Pepper, open the **pml** file of the app with Choregraphe, connect to your Pepper, open the *Robot applications* window and click on the first button ("*Package and install current project to the robot*"). The Choregraphe projects are located in the corresponding folders inside **caresses-opensource /CAHRIM/ActionsLib/NAOqi_apps**:
  - *asr2 (more on this later)*;
  - *Caresses Multimedia*;
  - *choice_manager*;
  - *compilation*;
  - *CustomNumber*;

[2] If some other package is missing, when you will run **cahrim.py** for the first time (see section 2.4), you will get a specific import error for any missing package and you can thus fix it on the fly.

[3] http://doc.aldebaran.com/2-5/dev/community_software.html

- *DateSelector*;
- *display_weather_report*;
- *game_memory*;
- *pictures*;
- *play_youtube*;
- *Time12Selector*;
- *Time24Selector*.

- Install the *follow_me* app by downloading the package from this link[4] (ask SoftBank Robotics for obtaining the access to the site) and uploading it to Pepper through Choregraphe.

- If the app *Force perception* is installed on your robot, uninstall it in order to avoid undesired behaviours in the head pose.

N.B. - Autonomous Speech Recognition app (asr2) requires that several Python libraries are installed on Pepper. As it is not possible to install libraries on the robot, they should be installed on your computer first and then copied into the **asr2/lib/otherlibs/** folder before uploading the *asr2* NAOqi app to Pepper so that the app can later access them in the local path. The libraries are: *cachetools, google-api-core, google-api-python-client, google-auth, google-cloud-speech, googleapis-common-protos, httplib2, oauth2client, pyasn1, rsa, six, SpeechRecognition, uritemplate, urllib3*.

2.3.1.3 Specific Requirements for Single Actions

DisplayWeather DisplayWeather action relies on the API provided by Open Weather Map. Therefore, you need to get first an API key in order to use it. You can get your by following the instructions at *this link*. Then you have to paste your key inside the file *CAHRIM/ActionsLib/NAOqi_apps/display_weather_report_APP/App-DisplayWeatherReport/ html/js/weather.js* assigning it to the *appKey* variable:

```
var appKey = "<openweathermap-API-ID>";
```

Do this before uploading the *display_weather_report* NAOqi app to Pepper.

GoTo GoTo action performs graph-based navigation. You need to create the map of your environment first.

```
$ cd ./CAHRIM/ActionsLib/aux_files/go_to/
$ python draw.py
```

[4] https://softbankroboticseurope.sharefile.eu/home/shared/fo0bfbfa-3127-4ce8-96bb-9b3f84734fc3

The map editor window will pop up. Draw the map *following the tips on the bottom bar of the window*. Use "Node Mode" to draw the nodes of the graph and "Edge Mode" to connect them. Nodes are oriented and by default the orientation is 0 degrees. To change the orientation of a node right-click on it.

Finally save the file with a name of your choice in the directory *Software/CAHRIM/ActionsLib/aux_files/go_to/maps/*. When you will launch CAHRIM you will be asked to select from this folder the map that you want to use. The file that you select will be copied as *map.json* at the parent directory *Software/CAHRIM/ActionsLib/aux_files/go_to/*. If a single map is present in the *maps* folder, it will be automatically selected.

N.B. - Passing-only nodes can be arbitrarily labelled, whereas the nodes meant to be destinations for the GoTo action should be labelled as a subset of the locations stored in the CKB (case sensitive!), which are: ***bed***, ***door***, ***charger***, *fridge, gasCooker, radio, tv, cabinet, drawer, wardrobe, armchair, chair, curtains, precharger, sofa, washSpace, window, kitchenWorktop, table*. **The first three nodes must be necessarily inserted in the map!**

N.B. - Please insert a node between the ***charger*** and the node where the interaction will take place.

ReadIoT / OperateIoT These actions control and read info from IoT devices. This Section explains how to configure a Belkin WeMo Smart Switch:

- plug the Smart Switch to an electricity plug close to the router while keeping pressed the small reset button;
- first, an *orange* LED will blink for a while; then the LED will turn *green* and keep blinking; finally it will blink alternating an *orange* and *green* color;
- at this point the device will create a WeMo Insight WiFi network;
- download the WeMo app on your smartphone but do not launch it yet;
- turn off mobile data and forget known WiFi networks;
- connect your smartphone to the WeMo Insight WiFi network;
- launch the WeMo app and look for the available devices;
- when the Smart Switch is found, give it a name by following the instructions given at the end of this Section;
- select the WiFi network which should be used by the Smart Switch from now on;
- when the LED stops blinking and the WeMo Insight network is no more visible, the Smart Switch is configured. You may need to restart the WeMo app to verify the configuration.
- Set up Pepper by following the instructions of the **doc_bifrost.html** document and by downloading and installing **bifrost.apk** from this link[5].

[5]https://softbankroboticseurope.sharefile.eu/home/shared/fo60c253-d91d-48e3-94a4-1e9ebf011151

N.B. - In order for the action to work you have to rename your device through the procedure explained in the WeMo guide. The device should be named as "<device>_<room>", e.g. "smartLamp_bedroom" being careful to avoid trailing spaces. The name of the device and the room should be in accordance with what is stored in the CKB. Available smart devices are: *smartLamp, smartRadio, smartFan*. Currently, the room can only be *bedroom*.

ReadMenu You need to configure the file *./CAHRIM/ActionsLib/aux_files/meals-conf.json*. Open the file and fill or leave empty each field according to the current day menu. Only the fields which are already present in the file are available. Therefore, try to fit the menu to those voices and leave the onese which are useless as empty arrays. For each available meal insert the dishes inside an array, e.g:

```
"havingLunch": ["lamb cassarole", "mince", "curry and rice"]
"havingSnack": []
```

SendLineMessage In order to use this action, you need to create first your own Line Channel and your bot. To do this, follow the instructions at *this link*. Paste your tokens inside the file *Required_Software/line-bot-server/app.py*:

```
channel_secret = '<LINE_CHANNEL_SECRET>'
channel_access_token = '<LINE_CHANNEL_ACCESS_TOKEN>'
```

You also need to host your bot on a server. You can use Flask. Go to *Required_Software/line-bot-server/* and then:

```
// Getting started
$ export LINE_CHANNEL_SECRET=YOUR_LINE_CHANNEL_SECRET
$ export LINE_CHANNEL_ACCESS_TOKEN=YOUR_LINE_CHANNEL_ACCESS_TOKEN

$ pip install -r requirements.txt
// Run WebhookParser
$ python app.py
// Run WebhookHandler
$ python app_with_handler.py
```

Finally, you have to paste your bot's URL inside *CAHRIM/ActionsLib/send_line_message.py*:

```
LINE_BOT_URL = '<LINE_BOT_SERVER>'
```

Besides, Relatives, friends and caregivers (*recipients*) who might be contacted by the User should go through the following steps:

- Recipients must download the LINE app on their phone;

- Recipients must add the CARESSES bot on LINE app by searching for it on the app or by scanning QR Code at this link[6];
- Recipients must send a text message to the CARESSES bot and get back as a reply their personal LINE id;
- Recipients must provide their id to the person in charge of setting up the software.

The id of each recipient should be added in the CKB (see section 2.3.2).

SendTelegramMessage In order to use this action, you need to create first your own Telegram bot. To do this, follow the instructions at *this link*. Paste your token inside the file *CAHRIM/ActionsLib/send_telegram_message.py*:

```
TELEGRAM_BOT_TOKEN = "<TELEGRAM_BOT_TOKEN>"
```

Relatives, friends and caregivers (*recipients*) who might be contacted by the User should go through the following steps. A more detailed explanation of the following passages is given in Appendix A.

- Recipients must download the Telegram app on their phone;
- Recipients must look for the @myidbot bot from the Telegram app and start a chat with it;
- Recipients must type in the chat `/getid` and get back as a reply their personal Telegram id;
- Recipients must look for the @Caresses_bot bot from the Telegram app and start a chat with it;
- Recipients must provide their id to the person in charge of setting up the software.

The id of each recipient should be added in the CKB (see section 2.3.2).

TakeAndSendPicture If you plan to give the user the possibility to send pictures by email, then you have to fill the file *CAHRIM/ActionsLib/aux_files/email-conf.json* in.

YouTube related actions In order to use YouTube API you need to get first a *developer key* from Google. You can obtain yours following the guide at *this link*. Then you have to paste your key inside the file *CAHRIM/ActionsLib/aux_files/youtube_helper/youtube_helper.py* assigning it to the DEVELOPER_KEY constant:

```
DEVELOPER_KEY = '<YOUTUBE_DEVELOPER_KEY>'
```

Besides, you need to create the sql database which keeps track of the videos watched and the time at which the video was last stopped.

[6]https://qr-official.line.me/M/GQJJzejk_8.png

```
$ cd ./CAHRIM/ActionsLib/aux_files/youtube_helper/
$ python create_db.py
```

2.3.1.4 Activity Recognition

Follow the README at this link[7].

[7] https://github.com/laboratoriumDIBRIS/caresses-opensource.git/tree/master/CAHRIM/CahrimThreads/ActivityAndLocationRecognition

2.3.2 CKB

- Install Java JRE 1.8;
- only on Linux:

```
$ echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/your_path/'
>> ~/.bashrc
$ source ~/.bashrc
```

  where 'your_path' is the path to the *<...>/CKB/NeticaJ_Linux/NeticaJ_504/bin/* folder. If you are using Linux 64 bits then 'your_path' is the path to *<...>/CKB/NeticaJ_Linux/NeticaJ_504/bin/64_bit/*.

- Customize the ontology with the user's information: run the customization tool from the CKB folder:

```
$ cd ./CKB
$ java -jar ./tool.jar
```

  The customization software will now start and you will be required to enter the following data:

  - user's nationality: it could be **en** (English), **in** (Indian) or **jp** (Japanese)
  - user's gender: **m** (Male) or **f** (Female)
  - language: **english** or **japanese**
  - user's name
  - user's surname
  - user's contacts (i.e., relatives, friends and caregivers that can be contacted by the user): while the previous four points are mandatory, you may not want to enter any contacts.

    ```
    $ Do you want to insert the user contacts? (y-n)
    ```

    If you do not have any information about relatives, friends and caregivers, answer **n** to this question. Otherwise, enter **y**. On the terminal, you will get:

    ```
    $ Insert the contact (Sister, Brother, Wife, Husband,
       Daughter, Son, Grandchild, Friend, MedicalStaff, Doctor
       , Nurse) or write exit when done
    ```

    Just follow the instructions on screen. You will be required to enter the contact, and the available related info (Name, Skype, E-mail, Telegram, Line, Phone). When you have entered all available information, type **exit**. In the same way, enter **exit** when you have added all contacts.

N.B. - If the user has more than one daughter (or brother, grandchild, friend, ...) please add information only for one of them.

For each family relationship whose contact has not been added, the software will ask if it would be ok for the robot to talk about that topic. For instance, if you haven't added any information about a brother, you will get:

```
$ You haven't inserted any data for a brother. Is it ok
   if the robot talks about this topic? (y-n)
```

Unless you are totally sure that it would be inappropriate for the robot to talk about this topic, please answer **y**.

– sensitive topics that should be avoided by the robot.

```
$ Are there sensitive topics that should be avoided by
   the robot (y-n)?
```

Enter **n** if there are no topics that needs to be avoided. Otherwise please enter **y** (as before, answer **y** only if you are sure that it is stricly necessary to remove that topic from the dialogue tree).

```
$ Please select one of the following topics:
   BeliefAndValue, ReceivingVisit, PrayingAndMeditation,
   Health, Family, UserMarriage. Write exit when done
```

Possible sentitive topics are: *Belief and Value*, *Receiving Visits*, *Praying and Meditation*, *Health*, *Family*, *Marriage*. Enter **exit** when finished.

– optional nodes of the map. The map of the environment will be composed by three fixed nodes, Charger, Door and Bed, and possibly by a certain number of optional nodes, that should be chosen from the following list: Armchair, Chair, Cabinet, Curtains, Drawer, Fridge, Radio, Sofa, Table, Tv, Wardrobe, WashSpace, Window. Enter *exit* if you do not need to add any optional node to the ontology. Otherwise, enter the name of the node. You will be then asked to insert the likeliness of the node: please always use 0.5, as in the example below.

```
$ Let's prepare the map. Fixed nodes are: Charger - Door
   - Bed
$ Optional nodes are: Armchair, Chair, Cabinet, Curtains,
    Drawer, Fridge, Radio, Sofa, Table, Tv, Wardrobe,
   WashSpace, Window.
$ Insert one of the optional nodes, or write exit to
   continue:
$ Armchair
```

When you have entered all optional nodes, write **exit**.

– smart devices.

```
$ Last step, the smart devices. Are we using them? (y-n)
```

If you are not using smart devices in the experiments, enter **n**. Otherwise, write **y**. In both cases you will be then asked to specify what kind of device are you using (SmartLamp, SmartFan or SmartRadio).

```
$ Please tell me which device we are using. SmartLamp,
   SmartFan or SmartRadio. Write exit when done
```

If you are not using smart devices, just enter **exit**. Otherwise, select one of the possible devices. You may also add more than one device. When you have added all devices, enter **exit**.

The ontology is now customized for the user.

2.3.3 CSPEM

- Install Java JRE 1.8;
- Get the latest version of SpiderPlan from the *Required_Software* folder of the **caresses-opensource** repository and execute the following commands on a terminal:

```
$ cd ./Required_Software/SpiderPlan
$ ./gradlew install
```

2.3.4 uAAL Skeleton

- Install Java JRE 1.8;
- Get **karaf_CARESSES.zip** zipped folder from the *Required_Software* folder of the **caresses-opensource** repository. Copy the folder to a directory of your choice outside your local copy of the Software repository and decompress it.
- Get the universAAL skeleton files from the *uAAL_Skeleton* folder of the **caresses-opensource** repository. Copy the files **uAAL-CAHRIM-1.1.4.jar**, **uAAL-CKB-1.1.4.jar**, **uAAL-CSPEM-1.1.4.jar** to the *deploy* folder of the **Karaf** folder you previously decompressed.

2.3.5 caressOS

- Install Java JRE 1.8.

2.4 Execution

2.4.1 Execution of Standalone CAHRIM Actions

Even without running the whole CARESSES Software, CAHRIM Actions can be executed standalone.

In order to run a CAHRIM Action, simply run the corresponding python script by passing the IP of the robot as argument. If you have already configured the **caresses-conf.json** file, the IP argument is not required.

```
$ cd ./CAHRIM/ActionsLib
$ python ./<action_name>.py --ip <Pepper_IP>
```

When the CARESSES Software is executed as a whole, the suggestions for the actual parameters of an Action come from the CKB according to their likeliness (see Section 2.2); when the Action is executed standalone, the suggestions used are the ones hardcoded at the bottom of the code. You're free to change them as you want (if more than one, they should be separated by a double ampersand symbol), as long as what you insert as suggestion is also listed in the parameter file used by that Action.

2.4.2 Execution of the Whole Software

Please notice that the software will run with Pepper in a "semi-autonomous" mode. CAHRIM will automatically handle this at start-up. There is no need to turn on Pepper's *autonomous life*. However, if this modality is already on, CAHRIM will automatically turn it off and will turn on only some of the robot's autonomous abilities.

- Switch on the robot;
- Check that the **caresses-conf.json** file in the **caresses-opensource** directory is correctly configured and in compliance with **caresses-conf_TEMPLATE.json** (*and keep an eye on the IPs. They like to change*). Do not remove the "*CAHRIM-database*" fields; if you did not set up the sql database for activity recognition (not compulsory) they are simply ignored. Remember to fill the "interaction-node" field only if you want the interaction to take place in a predefined node of the map, otherwise leave it empty.
- **N.B.** - **If you are using uAAL** start Karaf and the uAAL Skeleton:

  ```
  $ cd <PATH_TO_KARAF>/bin
  $ ./karaf
  ```

  and then execute the other modules in the following order:

  - CAHRIM;
  - CKB; finally, when the message "CKBready" appears on the screen, run
  - CSPEM.

- **N.B.** - **If you are using caressOS** start caressOS:

```
$ cd ./uAAL_alternative
$ java -jar ./caressOS.jar
```

and then execute the other modules in any order.

- Start CAHRIM:
  - Start Bifröst app:
    - access Android home page of Pepper's tablet (swipe upwards from the bottom edge and tap on the circle icon);
    - open Bifröst app and wait for the WeMo device to be displayed.
  - Be sure to have configured the file *./CAHRIM/ActionsLib/aux_files/meals-conf.json* as explained in Section 2.3.1.3.
  - Be sure to have created the map for your environment as explained in Section 2.3.1.3.
  - Put the robot in a node of the map you are going to use with the correct orientation;
  - Finally run cahrim.py by specifying the starting node:

```
$ cd ./CAHRIM
$ python ./cahrim.py <node>
```

  - You will be prompted to confirm or select which map you want to use among the ones present in the folder *./CAHRIM/ActionsLib/aux_files/go_to/maps/*.

- Start CKB:

```
$ cd ./CKB
```

Then, run one of the following:

```
$ java -jar ./CKB.jar -nationality English
```

```
$ java -jar ./CKB.jar -nationality Indian
```

```
$ java -jar ./CKB.jar -nationality Japanese
```

There are also some optional flags that can be used in order to configure the CKB. These are:

- *virtual*: 0 or 1 (default value is 0). If 1, the CKB is started in the virtual mode, i.e. without the robot. It could be useful for testing the system.

- *language*: English or Japanese (default value is English). It selects the language of the robot.
- *objects*: 0 or 1 (default value is 0). If 1, the CKB will send to CSPEM information about the location of some objects that are relevant for the user, if these information are encoded in the ontology.
- *update*: 0 or 1 (defualt value is 1). If 1, the ontology is updated with the user's feedback.
- *cloud*: 0 or 1 (default value is 0). If 1, the CKB is started in the *cloud* mode.
- *options*: 0 or 1 (default value is 0). If 1, you will be required to enter some additional information after having executed the software (i.e. location, time of the day).
- *cultural*: 0 or 1 (default value is 1). If 1, the robot will be *culturally-competent*
- *goal*: *react* or *greet* (default value is *greet*). If *greet*, the robot will start the interaction, moving towards the user; if *react*, the robot will be in the status *React To Sound*, waiting to be called by the user.

Finally, there are two flags that will be used only during the CARESSES experiments (*group*, that could be control or experimental, and *session*, a number between 1 and 6). Their usage will be shown in Chapter 3.

N.B. - On Windows, if you get the following error

```
$  <PATH_TO_CKB>/NeticaJ.dlll: Can't find dependent libraries
```

add <PATH_TO_CKB> to the Windows environmental variable PATH[8]

- Start CSPEM:

```
$ cd ./CSPEM
```

If on Windows, use this command:

```
$ while(1){.\CSPEM.bat}
```

otherwise, use this:

```
$ while true; do ./CSPEM.sh; done
```

[8]For further information about how to set the Windows environmental variable PATH, go to https://www.java.com/en/download/help/path.xml

2.5 Terminate the Software

- To terminate the three CARESSES modules simply press CTRL + C
  N.B. - when killing cahrim.py, press it only once!
- To terminate uAAL press CTRL + D.
- To terminate caressOS simply press CTRL + C.

2.6 Saving, Loading and Deleting a User Session

We use the expression *User Session* to indicate the ensemble of the files used in and created by the CARESSES software before/during/after the interaction between the Robot and the User which contain data related to that specific User or that specific interaction. Therefore, a *User Session* consists of:

- the Cultural Knowledge Base containing User-specific data (CKB.owl);
- the map (graph) of the environment (map.json);
- the transcription of the dialogs (speechLog.log);
- the log of CAHRIM module (cahrimLog.log);
- the log of CARESSES exchanged messages (uAAL_log.text);
- the database of the videos already seen by the User (youtube-history-db.sqlite);
- the custom reminders verbally dictated by the User to the robot (reminder_generic.json);
- the long term goals which could/should be achieved during a future interaction (stored-goals.uddl);
- the pictures stored on the robot.

The Python script **user_data.py** inside the **caresses-opensource** directory allows for:

- **saving** a User Session before starting a new one;
- **loading** a past User Session which has been previously saved;
- **clearing** a User Session which is not meant to be saved before starting a new one.

User sessions are saved to and loaded from the *./Experiments* directory.

2.6.1 How to Save Current User Session

In order to save the current User Session:

- terminate CARESSES software;
- run the following command from the **caresses-opensource** directory:

```
$ python user_data.py -s
```

- enter the required information when prompted (User ID).

A folder named as 'userID_YYYY-MM-DD' (e.g. U002_2018-10-23) will be created inside the *./Experiments* folder. If a folder with this name already exists, a number is appended at the end of the folder name. The aforementioned files are moved to the newly created folder and new 'default' files are created where necessary for a new session.

2.6.2 How to Load a Previously Saved User Session

In order to load a past session which has been previously saved:

- terminate CARESSES software;
- run the following command from the **caresses-opensource** directory:

```
$ python user_data.py -l
```

- enter the required information when prompted (User ID);
- if a single saved session is present for this User, it is automatically selected as session to be loaded. If more than one saved session is present for this User, all the available sessions are listed. Enter the index corresponding to the session that you want to load when prompted.
  N.B. - On different operating systems sessions may be listed in a different order, select carefully the index of the session you want to load.

The files of the selected session are moved to the their corresponding "execution paths".

2.6.3 How to Clear Current Session

In order to clear the current session and prepare the files for a new one without saving anything:

- terminate CARESSES software;
- run the following command from the **caresses-opensource** directory:

```
$ python user_data.py -c
```

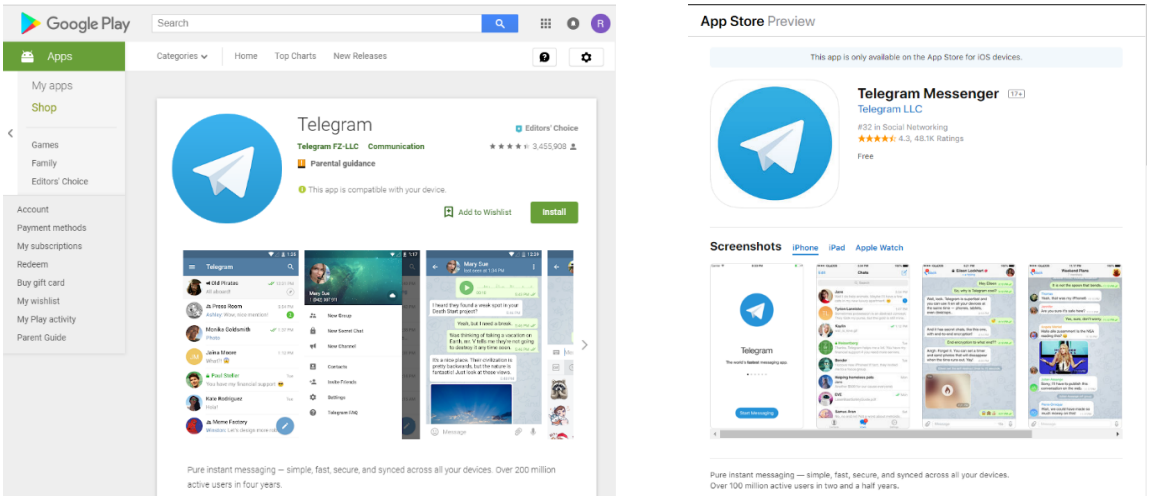
The files of the current session are deleted and new 'default' files are created where necessary for a new session.

Appendices

Appendix A

Instructions for Receiving Telegram Messages from the User Interacting with the CARESSES Robot

1. Download the **Telegram** messaging app on your phone from the Google Play Store (Android – Figure A.1(a)) or the Apple App Store (iOS – Figure A.1(b));



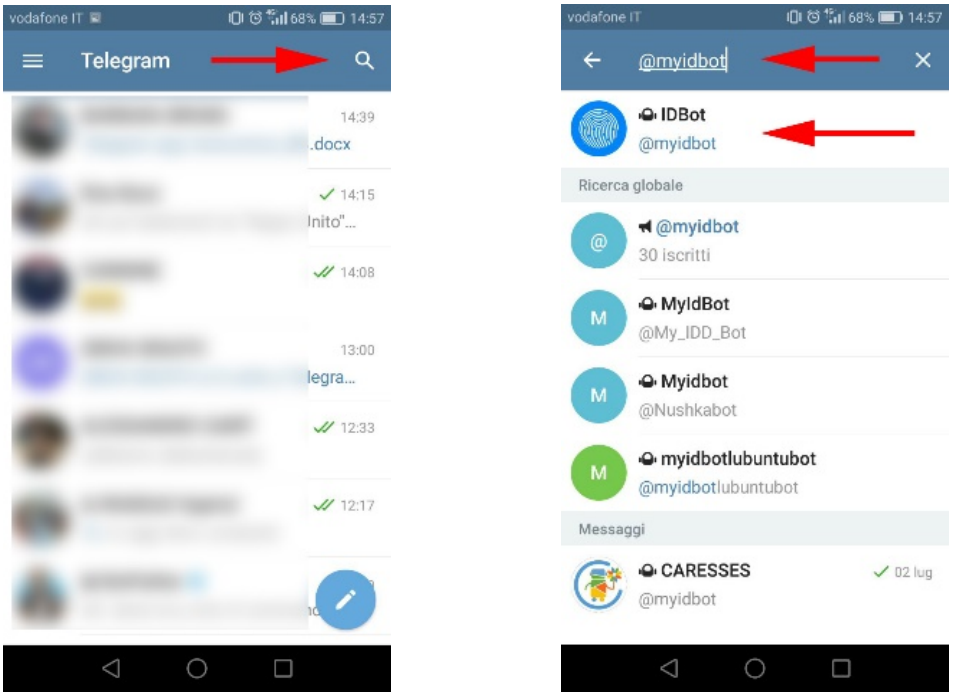
(a) (b)

Figure A.1

2. Install the app;

3. Launch the app and Join Telegram by providing your phone number;

4. Press the magnifying glass icon on the top right corner of the app screen (Figure A.2(a)) and type **@myidbot** to search the "IDBot" bot (Figure A.2(b));



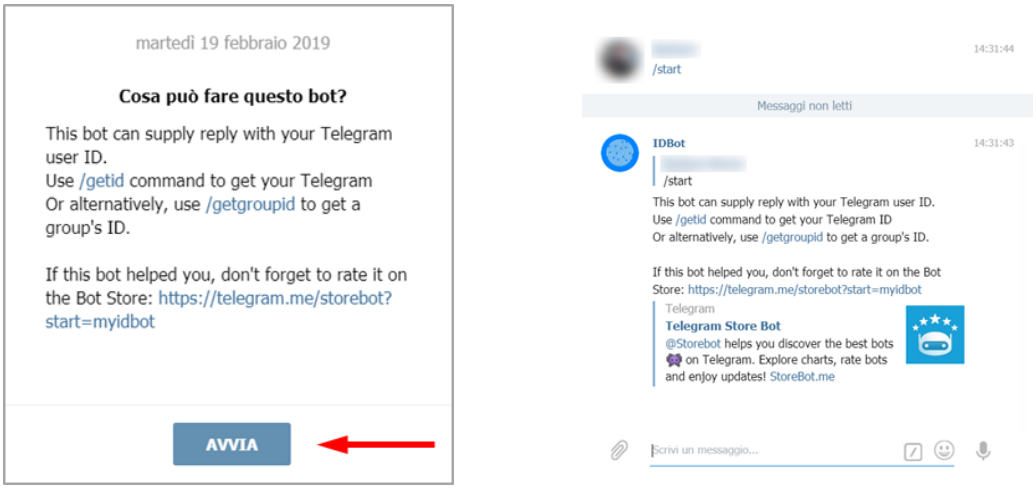
(a) (b)

Figure A.2

5. Press the first result, which looks like Figure A.3;

Figure A.3

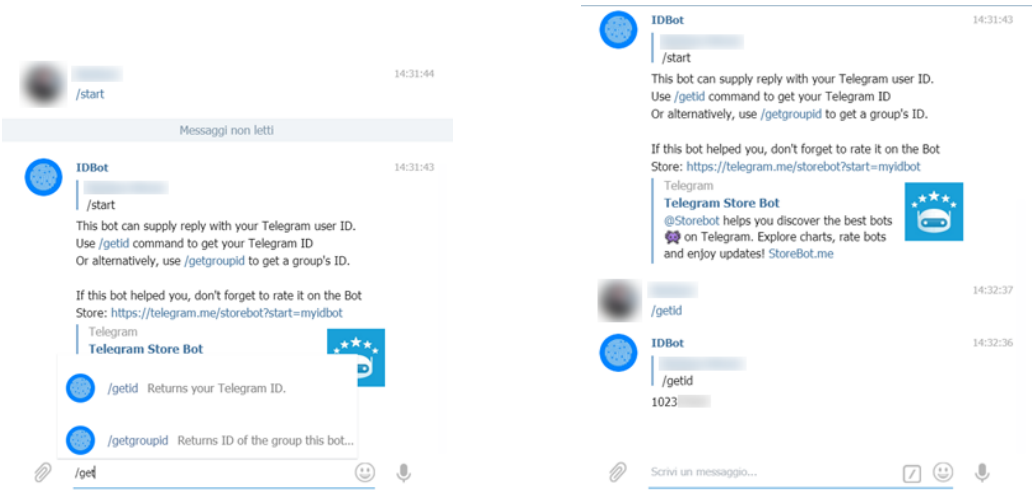
6. An information message will pop up: press the “Start” button on the bottom (Figure A.4(a)) and a “chat” will start (Figure A.4(b))



(a) (b)

Figure A.4

7. Type in the chat **/getid** (Figure A.5(a)) and get back as a reply your personal 8-digits Telegram ID (Figure A.5(b) – some digits are blurred in the figure for privacy reasons);



(a) (b)

Figure A.5

8. ***Provide your 8-digits Telegram ID to the CARESSES staff*** so that it can be added to the temporary contact list of the custom-specific CARESSES robot;
9. Press the backward arrow on the top left corner of the screen;
10. Repeat steps 4 and 5 but this time type **@Caresses_bot** to search the "CARESSES" bot (Figure A.6);

Figure A.6

11. Finally, repeat step 6 to start a “chat” with the CARESSES bot. Whenever the User sends you a message through the CARESSES robot, you will receive it on this chat. *At the moment it is not possible to send messages to the User or to reply to incoming messages.*

Appendix B

Exercise Example

In order to fully test the entire CARESSES software, you can practice by simulating two complete sessions.

Two different scenarios are proposed. Use the given information to set up the system and customize the Cultural Knowledge Base, by carefully following all the steps described in this Handbook. For each scenario, run the software and interact with the robot for at least one hour. Try to test all the actions across the different tests. During the interaction, try also to send Pepper to the docking station (Rest Mode) and call it back after a while.

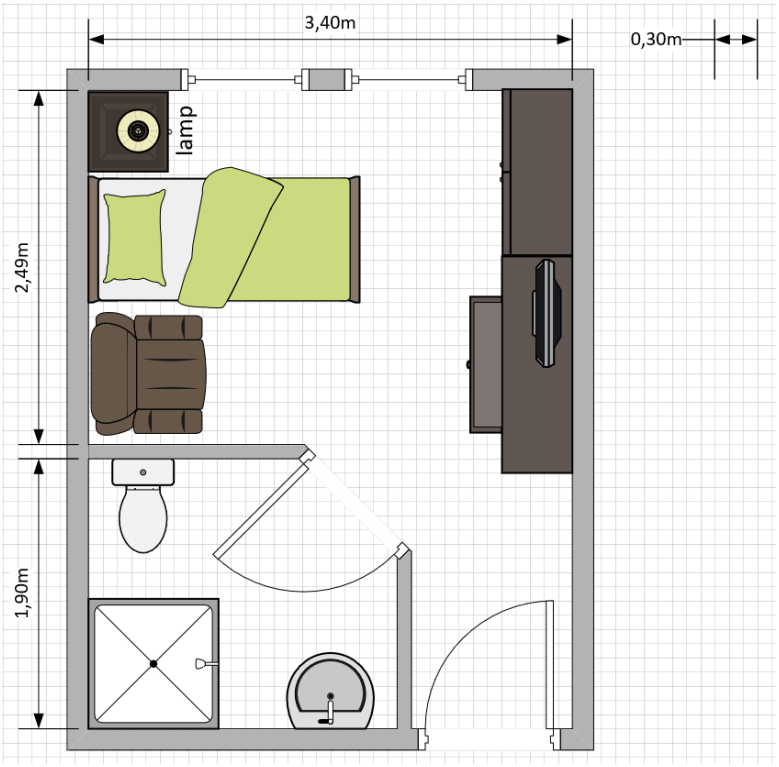
The information about two fictional Users are given in Tables B.1 and B.2. The layout of the room is given in Figure B.1(a) and an example of carehome menu is given in Figure B.1(b).

Table B.1: User Information for Scenario 1

RESIDENT ID	T1
GROUP	Experimental
CULTURAL BACKGROUND	Japanese
GENDER	male
FIRST NAME and FAMILY NAME	Tsubasa Ozora
CONTACT 1	daughter Aoba, Skype-ID: PepperCaressesAoba, phone 1234
CONTACT 2	wife Nakazawa, email: PepperCaressesNakazawa@soccer.jp, Line-ID: PepperCaressesNakazawaLine
CONTACT 3	Care home staff, phone: 5678
NOTES	Tsubasa said explicitly that he doesn't want to talk about religion.

Table B.2: User Information for Scenario 2

RESIDENT ID	T2
GROUP	Control
CULTURAL BACKGROUND	Japanese
GENDER	female
FIRST NAME and FAMILY NAME	Mila Hazuki
CONTACT 1	friend Shiro, phone: 1111, email: PepperCaressesShiro@volleyball.jp
CONTACT 2	friend Kaori
CONTACT 3	—
NOTES	Mila doesn't want to talk about volleyball anymore.



(a)

Week 4 - Thursday

BREAKFAST

CHOICE OF CEREAL OR PORRIDGE, BACON, SAUSAGE
EGGS (FRIED, SCRAMBLED or BOILED) AND TOAST

LUNCH

LAMB CASSAROLE
MINCE
VEGETABLE SPRING ROLL
VEGETABLE CURRY & RICE

DESSERT

YOGHURT
FRUIT SALAD
ICE CREAM

SUPPER

SOUP, BREAD AND BUTTER
MACCARONI CHEESE
ASSORTED SANDWICHES
FRESH FRUIT, CHEESE & BISCUITS

(b)

Figure B.1