Deliverable D5.3: High-Level task planner in relevant environments

Due Date: 30/03/2023

Main Author: ERM

Contributors: HWU, UNITN, CVUT, APHP

Dissemination: Public Deliverable

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

DOCUMENT FACTSHEET

Deliverable	D5.3: High-Level task planner in relevant environments
Responsible Partner	HWU
Work Package	WP5: Multi-User Spoken Conversations with Robots
Task	T5.2: High-level Robot Task Planner
Version & Date	30/03/2023
Dissemination	Public Deliverable

CONTRIBUTORS AND HISTORY

Version	Editor	Date	Change Log
1.0	ERM	10/01/2023	First Draft
1.1	HWU	20/01/2023	Second Draft
1.2	ERM	22/02/2023	Third Draft
2.0	HWU	24/03/2023	Final Draft

APPROVALS

Authors/editors	ERM, HWU
Task Leader	HWU
WP Leader	HWU

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

Contents

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

Abbreviations

Abbreviation	Meaning
ARI	Social assistive robot used by the SPRING project
AP-HP	Assistance Publique – Hôpitaux de Paris (SPRING Partner)
BIU	Bar-Ilan University (SPRING Partner)
CVUT	Czech technical university in Prague (SPRING Partner)
ERM	ERM Automatismes Industriels (SPRING Partner)
HRI	Human Robot Interaction
HWU	Heriot-Watt University (SPRING Partner)
INRIA	Institut National de Recherche en sciences et technologies du numérique (SPRING Partner)
PAL	PAL Robotics (SPRING Partner)
PNP	Petri-Net Planner
ROS	Robot Operating System
SPRING	Socially Pertinent Robots in Gerontological Healthcare
UNITN	University of Trento (SPRING Partner)
WP	Work Package (of the SPRING project)

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

Executive Summary

Deliverable 5.3 reports on the robot task planner and user applications T5.2. This report delivers software packages developed for the interaction with users of the SPRING robot applications and the task planner in relevant environments. The deliverable describes the design of these packages, interleaving the robot application and the low-level modules through the high-level planner to work in relevant environments, with direct interaction with users, whether these are patients (and their companions) or experimenter clinicians for the SPRING project.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement No. 871245.

1 Introduction

The goal of T5.2 is to develop the robot application and task planner to connect the overall robot goal, with the low-level goals and the current status of the environment. The robot task planner, as constraint by the use cases, will decide which is/are the forthcoming task/s to be performed by the robot, build upon the system in T5.1 and integrate with both the conversational system from T5.3 and the non-verbal behaviour manager from T6.3.

This document reports on the software modules for SPRING robot task planner and user applications for the direct interaction with users in relevant environments. The resulting system is based on data collected from the use-cases and stakeholder interactions of WP1.

Chapter 2 introduces the examined use-cases for SPRING, and define their priorities, constraints, default actions. Chapter 3 presents the network and data infrastructure developed to run experiments within SPRING at the relevant environments in Broca Day Care Hospital. Chapter 4 describes the design and implementation of the robot user applications for direct interaction with users, patients and experimenters for the relevant environments in SPRING. Chapter 5 describes the design and implementation of the robot task planner for the use cases in the relevant environments for SPRING.

The software will be released in the [11, 12] code repositories. As per European Commission requirements, the repository will be available to the public for a duration of at least four years after the end of the SPRING project. People can request access to the software to the project coordinator at `spring-coord@inria.fr`.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

2 Use Cases

SPRING's scenarios are in gerontological healthcare, and we will experimentally validate the developed technology in the Broca Day Care Hospital, relevant environment, that is in the facilities of AP-HP.

To work in good condition in WP1 AP-P and ERM defined basic key points for every use-case, and define priorities, constraints, default actions. The operation of the robot for SPRING use cases was refined through the user feedback from the preliminary validation collected in deliverable D1.4 [3].

The robot will be in only one hospital (Broca Hospital at Paris), only one ward (day care hospital specialized in gerontology). The interactions will take place in France, the patients will be French-speaking. The robot will have to be French-speaking as well. For the GDPR (General Data Protection Regulation) and PPC (Protection of Persons Committee), patients will be filtered. Only volunteers, interested and with some time, will be able to interact with the robot. To avoid any interaction with non-voluntary persons, the robot will be placed in an isolated room, where only patients willing to interact with it will be seated.

During the experiments/interactions, the robot will always be accompanied by, at least, one person from the hospital who can execute some procedures to handle the robot. A crucial aspect for the acceptability of the technology by the medical personnel is that they should be able to easily command the robot. The experimenters will have access, for instance through a tablet, to configuration and high-level commands (see section 4.2). It is very important for them to be able to easily: start/end the interaction, reboot certain functionalities, pause the robot, push the interaction towards certain scenarios/cases, etc. Researchers will be allowed to configure their modules if needed. Nevertheless, the goal is for the robot to be as autonomous as possible. If necessary, some features can be disabled during the experimentation.

2.1 Scenarios

The scenarios are conceived under two main guidelines:

(i) to evaluate the technology developed in the SPRING project;

(ii) to have a positive impact in the every-day life of the Broca Day Care Hospital staff.

The main SPRING use cases were initially defined as follows:

- Welcome:
  welcoming people arriving at the Broca Day Care Hospital.
- Safe Social Interactions:
  encourage safe social interactions in the public spaces in the Broca Day Care Hospital.
- Active Telepresence:
  provide a social telepresence service for people in the Broca Day Care Hospital.
- Patient Guidance:
  helping people navigate and travel to their appointments in the Broca Day Care Hospital.
- Entertainment:
  provide entertainment, with social interactions, to the patients and accompanying people visiting the Broca Day Care Hospital.

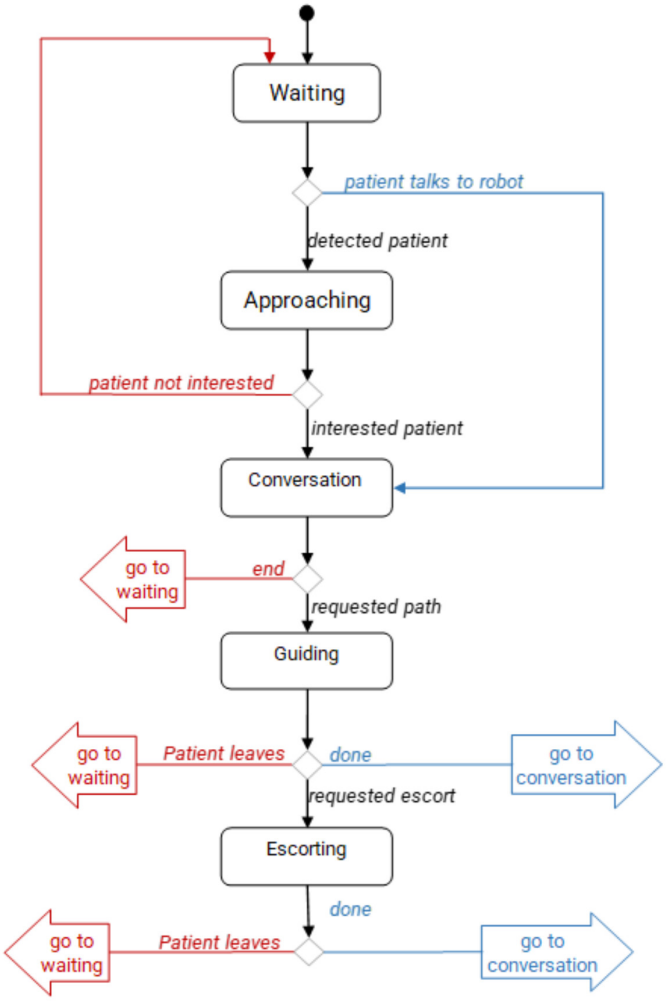

Figure 2.1: Operation Phases Diagram

2.2 Operating phases

Regardless of the SPRING scenario the robot is operating on, every scenarios must follow the same operating phases, depicted in the diagram of Figure 2.1, as follows:

- Maintenance (not in diagram): start-up (before and after use), setting and positioning.
- Waiting: the robot waits for a user, detects contact with user (patient) to start approach or conversation.
- Approaching: the robot has detected a patient, it approaches the patient and introduces itself.
- Conversation: the robot assist and chat with a patient.
- Guiding: the robot indicates a path.
- Escorting: the robot moves with a patient to a requested location.

3 Data and Network Infrastructure

We chose a NAS server from Synology for secure in access and reliability of the stored data resulting from the experiments. Deliverable 10.3 [2] provided guidelines on privacy and ethics informing the experimental validation and data collection security criteria that were followed:

- The use of a backup server separated from the rest of the system allows for autonomy in the evolution of the project.
- For access protection, authentication is required with login and password.
- The hard disks are encrypted to prevent intrusion. Even with direct access to the hardware, the data will remain unreadable, making it necessary to use the authentication system.
- For the reliability of the storage, a redundancy system has been implemented. This means that if one of the hard disks fails, the storage continues to function.

ERM designed and configured the network and data storage. This consists firstly of listing the machines and then assigning the addresses. Then, the network infrastructure is built according to the characteristics of each component. The device with the most constraints is the ARI robot. It needs a WiFi access point and a lot of bandwidth. We dimensioned the WiFi router according to this. Next, we had to find a solution for the Internet connection. The nature of the data in the experiment does not allow us to use the Broca Hospital network. ERM took a 4G mobile connection. To secure the connection, ERM called on a specialised company to set up a private APN (to avoid intrusions), a VPN (for remote access) and a firewall (to protect the system).

At the start of each experiment, ERM re-installs the network in the hospital and checks that it is working properly.

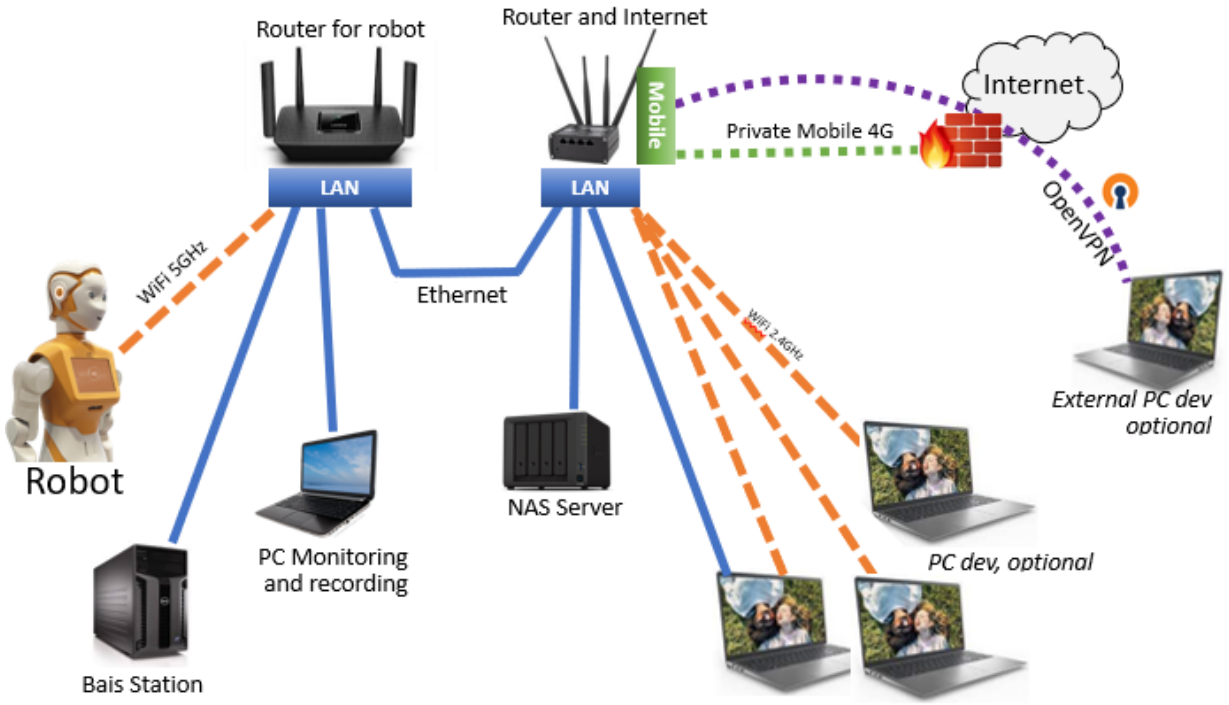

Figure 3.1: SPRING network architecture.

4 User Applications

This section presents the robot applications developed for SPRING providing direct interaction with users. The applications developed are divided into 3 classes relative to the type of user that will be interacting with them: patients, experimenters, and researchers for the data collection applications.

4.1 Interfacing with the Patient

4.1.1 Patient Screen

ERM's role is to develop a display for the patient. This was physically the screen of the robot. The first version was designed to prioritise the display of text and supporting images. For example, in the case of guidance use, the robot could display the hospital map to enhance understanding. From a technical point of view, the screen was divided into four fixed areas (to avoid display shaking), the dialogue module fed the information to be displayed, while the display problem managed fades, renders, timeouts and communication.

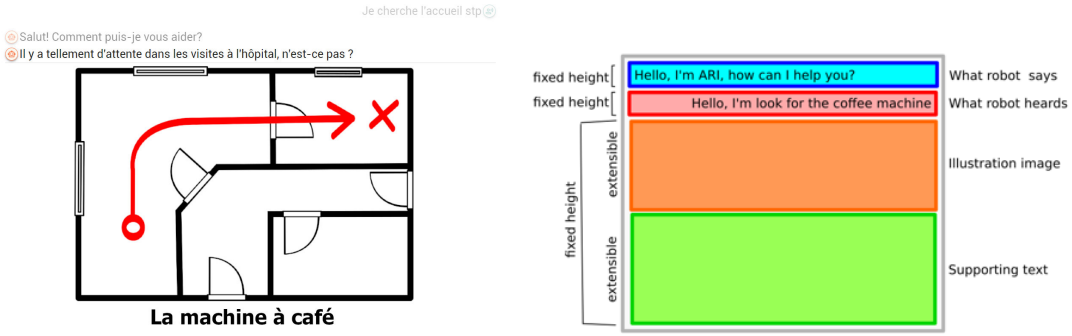

Figure 4.1: Example of the robot display. Display Layout

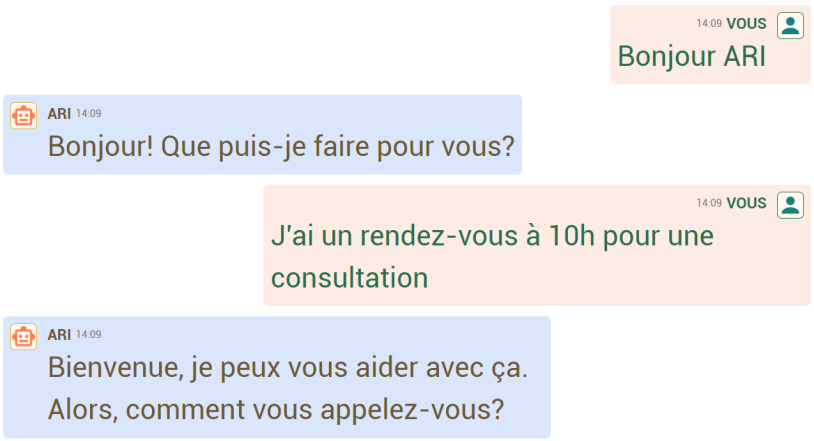

Figure 4.2: Display of the transcription robot

Then, as the project progressed and feedback was received, the project partners agreed to reduce the robot's display to full-screen transcripts. ERM modified its page to display only what the robot hears and what the robot says, in large print for the hearing impaired.

4.1.2 Other Robot Visual Signals

4.1.3 Eyes

PAL Robotics updated the firmware of the robot, allowing to manage the robot's gaze. Unfortunately, the basic program was too heavy and not adapted to our experiment. ERM redefined the driving program to have a more alive gaze and lower CPU consumption. There is no control over the direction of the eyes because other modules do not implement it and because the lens in front of the eyes physically prevents such a move.

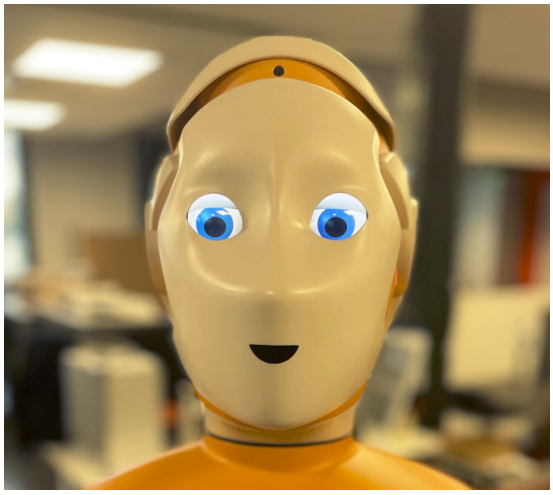

Figure 4.3: The Robot Eyes for the SPRING project

4.1.4 Microphone LED Array

To help the patients and companions interacting in conversation with the ARI robot know when the robot is listening, or waiting for a user reply, and when is talking, or processing speech, and not able to listen to the user utterances, a color messaging system using the LED ring display on the Re-Speaker Microphone array located at the front of the robot was implemented.

4.2 Interface with the Experimenter

4.2.1 Prototype Start-Up Screen

ERM had created a prototype to manage the start-up of the robot. The principle was that the experimenter could set up, test and launch his experiments directly via the robot's screen. There were two types of diagnostics:

- The automatic one where the robot checks its data and performs unit tests on the modules.
- Assisted, where the robot must interact with a human to calibrate itself and to confirm the results.

From a technical point of view, the structure of the diagnostics was modular, making it possible to adapt to the specificities of each module to be tested and to evolve at the same time as the project. The programme was not retained due to lack of time. This required each partner to devote development time to creating communicating test functions. The priority was to finish the module itself first, before thinking about interfacing it with these screens.

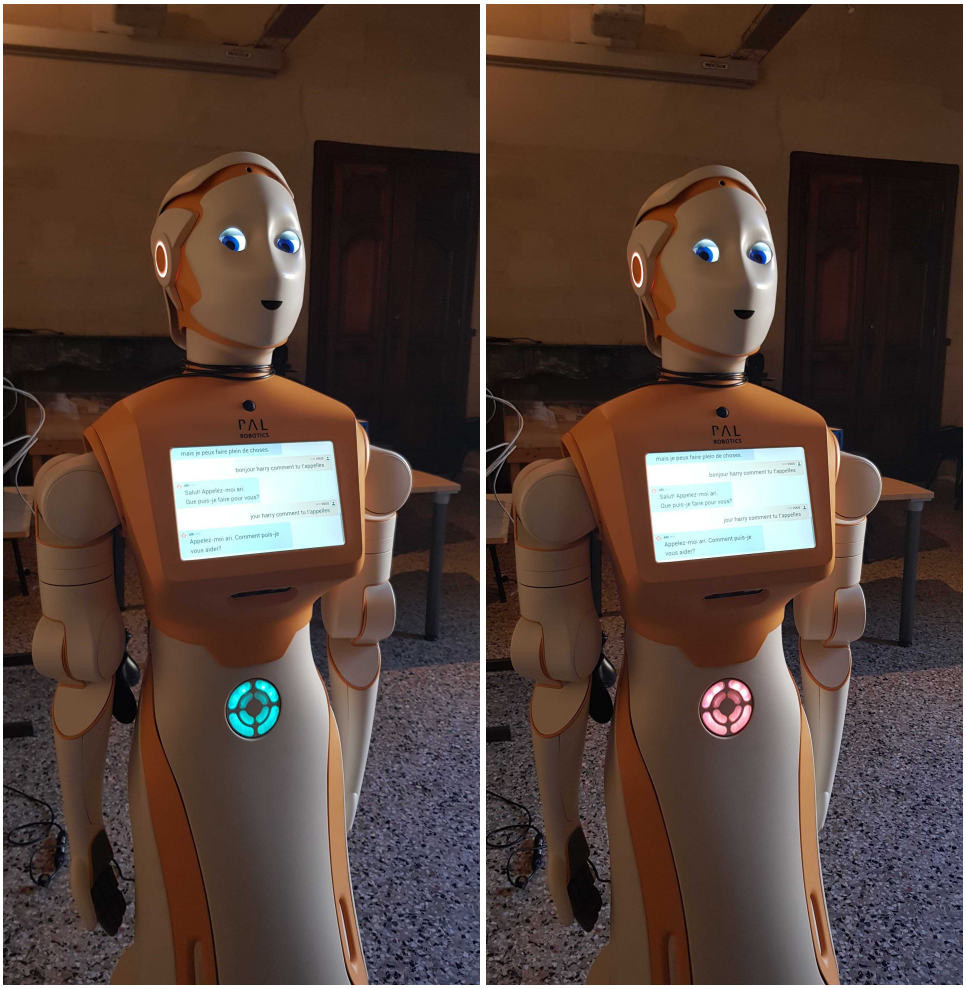

Figure 4.4: Example of the robot LED array signals.

4.2.2 Intermediate Experiment Screen

For the intermediate experimentation, the software architecture became more complex. The web interface and its service were no longer suitable for the recordings. The problem was that there were now two sources of data and wanting to centralise the records would affect the already limited performance of the robot network. As there was not enough time left, ERM created simple recording scripts to be run by hand. Unfortunately, this requires the permanent presence of a technician on the experiment. Then ERM developed a suite of programs to merge recordings from different sources, to merge sequences, to convert files into an accounting format with pre-existing tools and to pre-encode exports. Moreover, ERM also had to develop the encoding tool to adapt it to the new architecture.

In IT, there are many cases where data recording can fail (full hard disk, inoperative sensor, communication problem, robot fault, handling error, etc.). ERM has put together a small tool for the technician. It has a non-intrusive view

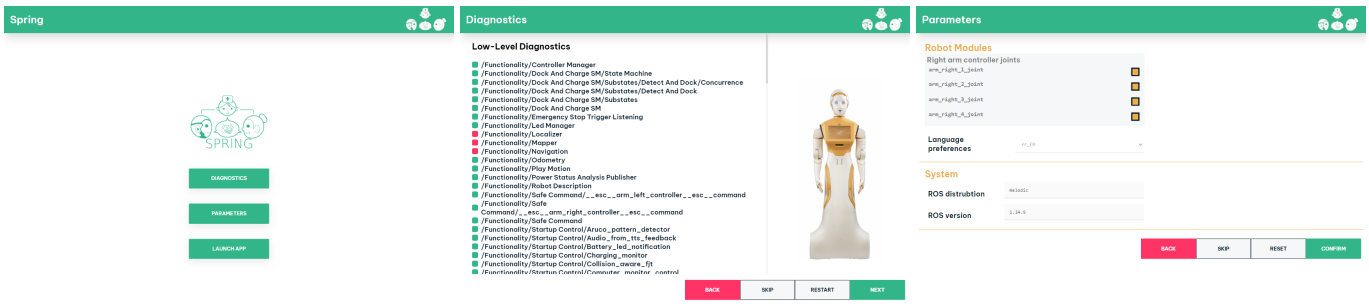

Figure 4.5: Integrated start-up, diagnostic and settings screen

on the status of the storage disks, on the camera feedback, on quick function tests and on the status of the robot.

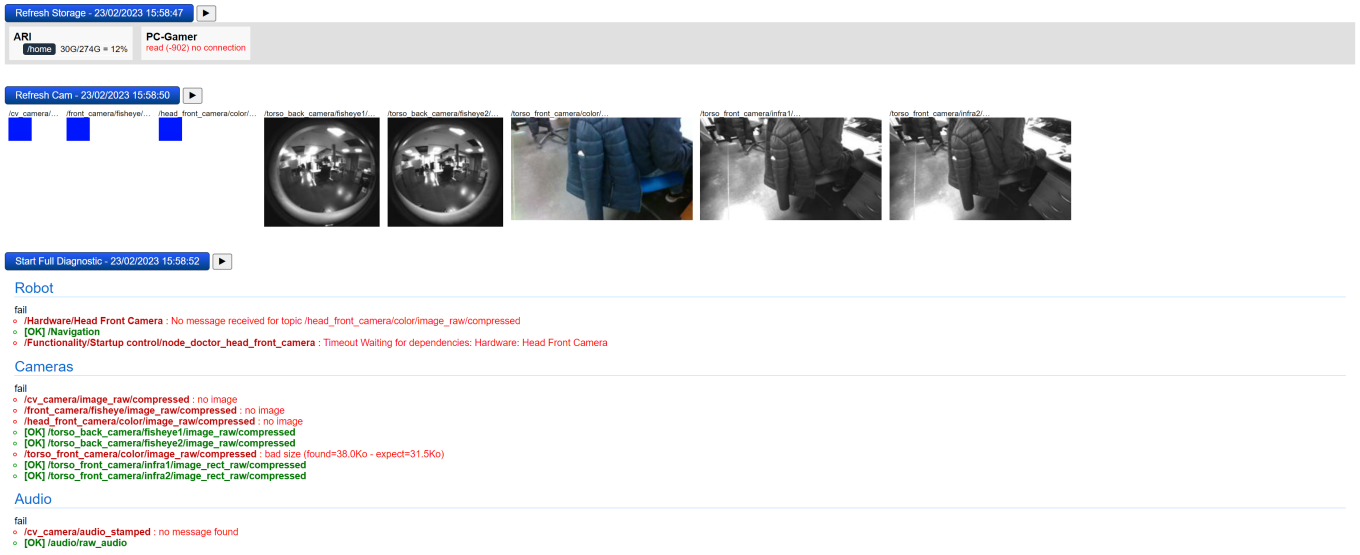

Figure 4.6: Technical view on the status of registrations

4.2.3 Final Experiment Screen

For the final experimentation, the robot will be alone with the APHP experimenters. In order to conduct their sessions, ERM has created a tablet interface to start and stop the robot. From a technical point of view, a server program runs in the calculation server. It communicates regularly with the ROS modules and the dockers and exports its results in HTML. The experimenter opens his web browser on his tablet to get his control screen.

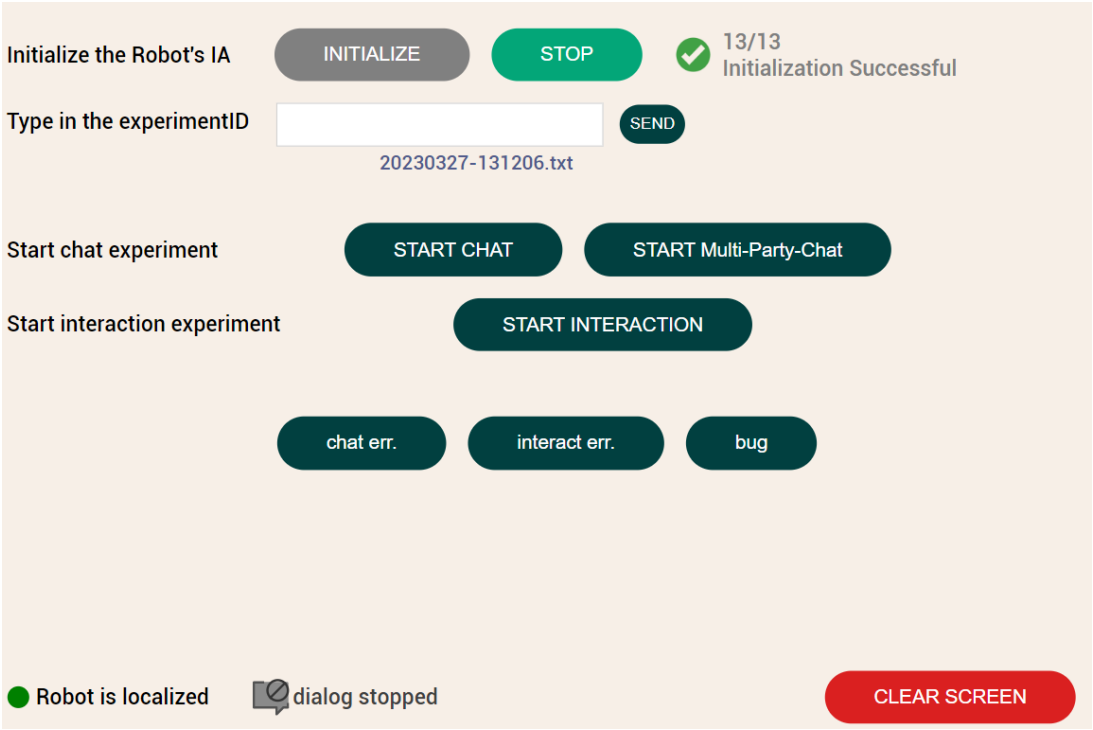

Figure 4.7: Control Screen

It will not be possible to record data over such a long period of time without the presence of technicians. The

compromise made between the project partners is to record only the conversations. ERM has created a program to export the data in a format that can be easily used by APHP and that works automatically. Using the control screen, it is possible to tag the exported file names to sort them and add annotations.

For the final experimentation, ERM provides a computing server, named Bais Station, to run all the modules of all the partners, with the necessary computing power. In addition to providing the hardware, ERM did the server installation and module installation. ERM also created automatic start-up tools so that the robot can be used without the need for technical knowledge. As a result, the server will be usable by APHP staff.

Figure 4.8: APHP Experimenter using the Robot Application to conduct and experiment a Broca Day Care Hospital

4.3 Interface for Data Collection

ERM was to provide the tools to run the initial data collection. In a meeting with the other project members, we agreed on the list of data to be recorded and the recording process. First, ERM wrote scripts to perform performance tests to discover the capabilities of the robot and assess the feasibility of recording. This was a program to be run in a console and display the playback statistics: latency (min, max, avg), rate (min, max, avg), data volume. This step was necessary to check that there were no configuration problems in the robot and for the dimensioning of the network.

Then ERM created an interface to be able to carry out the recordings on site. This involves creating a service to manage the recordings (start, stop, status), with the necessary safeguards (checking that the robot's sensors are operational, avoiding duplicate recordings, ensuring that recordings are always under the control of an operator). In order to conduct the recording sessions, the robot must be made to talk. ERM has taken advantage of the interface to add speech buttons. The programme also includes camera feedback to ensure that the robot is correctly positioned.

During the Data Collection, we realised that the robot lacked vocabulary, which left the patients speechless, hindering the recording. ERM quickly created a second interface consisting solely of speech buttons. The structure of the program made it easy to add new ones between each enrichment session. From a technical point of view, the solution needed to be quickly deployable rather than robust. In addition, it should not conflict with the recorder control interface needed to manage the recordings. The program is a mini web server that emits the same speech commands as the control screen.

Once the data is recorded, it must be checked by APHP before sharing. ERM has created a tool to export recordings in video format for easy review. This same tool also allows for the cutting of unethical footage. As encoding is time consuming and the volume of data to be processed is huge, a non-blocking queue mechanism has been added. Since the program was designed to run in parallel with the experiments, it also includes a tab for monitoring the status of disk space. The Data Collection interfaces were use for the Multi-party Dialogue Data Collection described on Deliverable D5.2 [5].

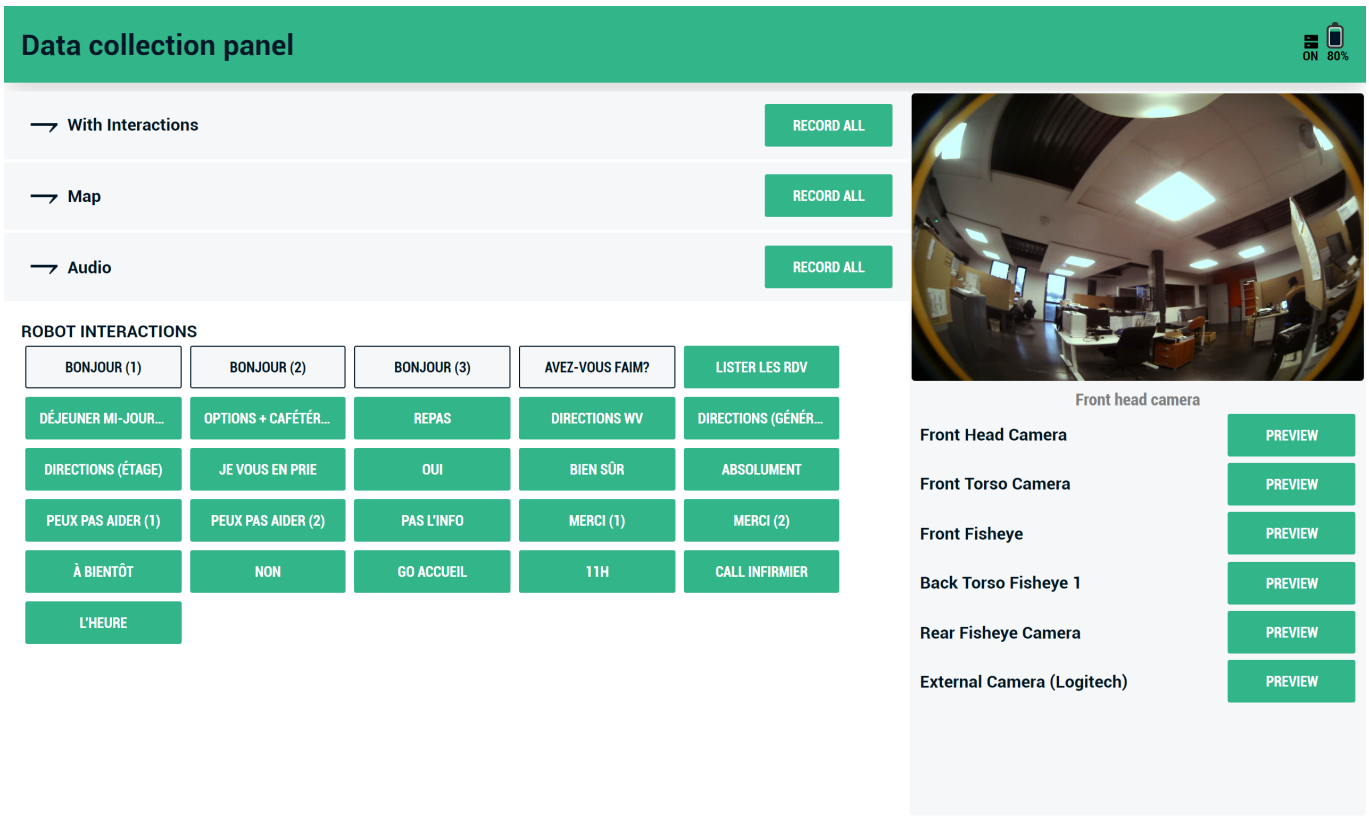

Figure 4.9: Web interface for monitoring recordings and speech

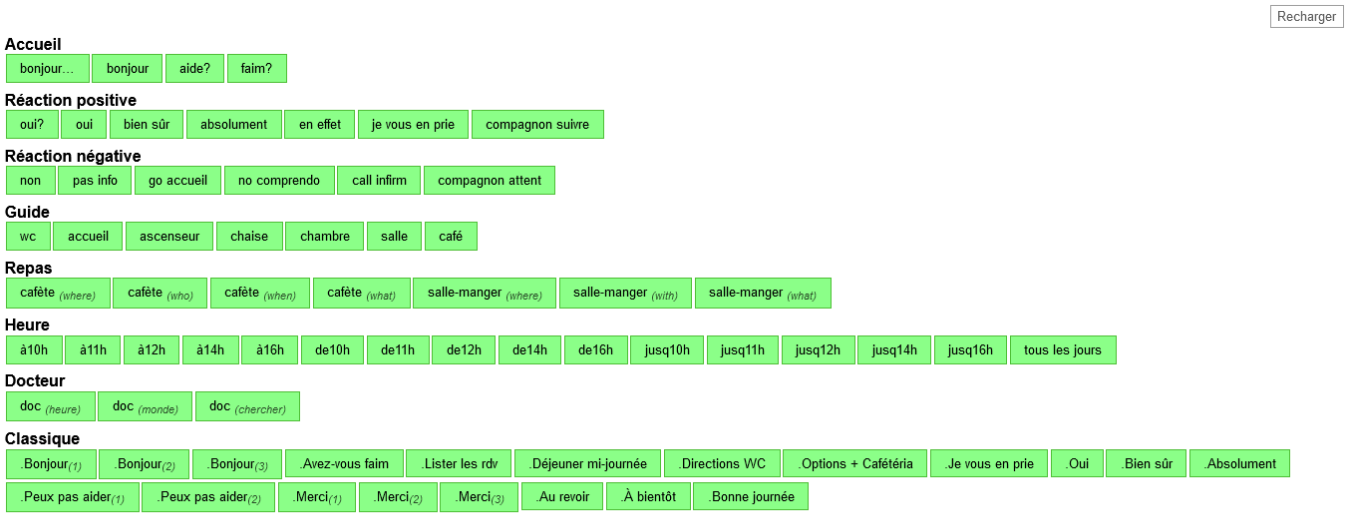

Figure 4.10: Web interface for speech button complements

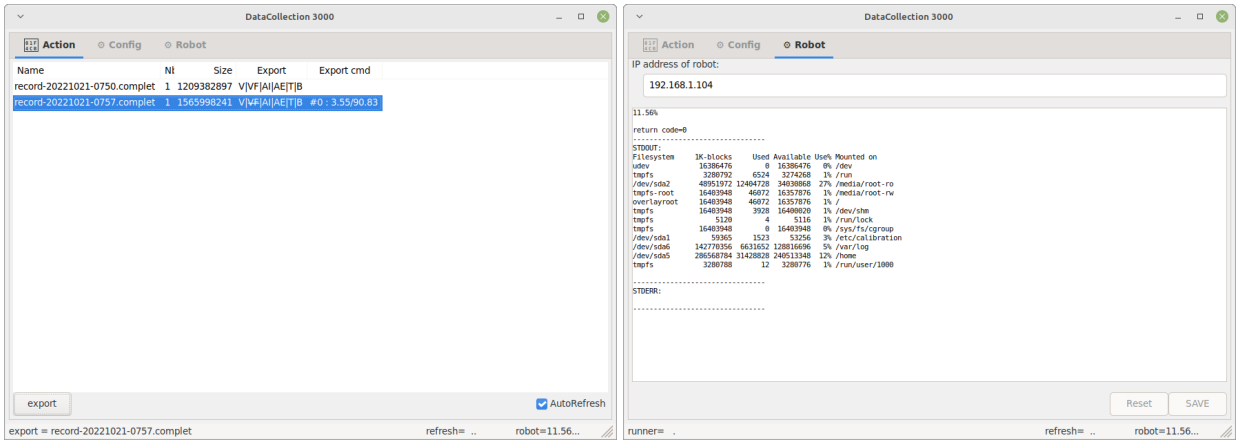

Figure 4.11: Video encoding and export tool for recordings

5 Hierarchical High-Level Social Planner

In SPRING the High-Level Task Planner to connects the overall robot's goal, with the low-level goals and the current status of the environment.

At the heart of the SPRING Robot Behaviour architecture for controlling the non-verbal behaviour system and the conversational system lies the High-Level Planner, which allows for multi-threaded, and concurrent, execution of dialogue and non-verbal task actions by the robot. To endow robots with the necessary skills to engage/disengage and participate in conversations the High-Level Planner enables sensor-based and knowledge-based robot actions for multi-modal multi-person interaction and communication.

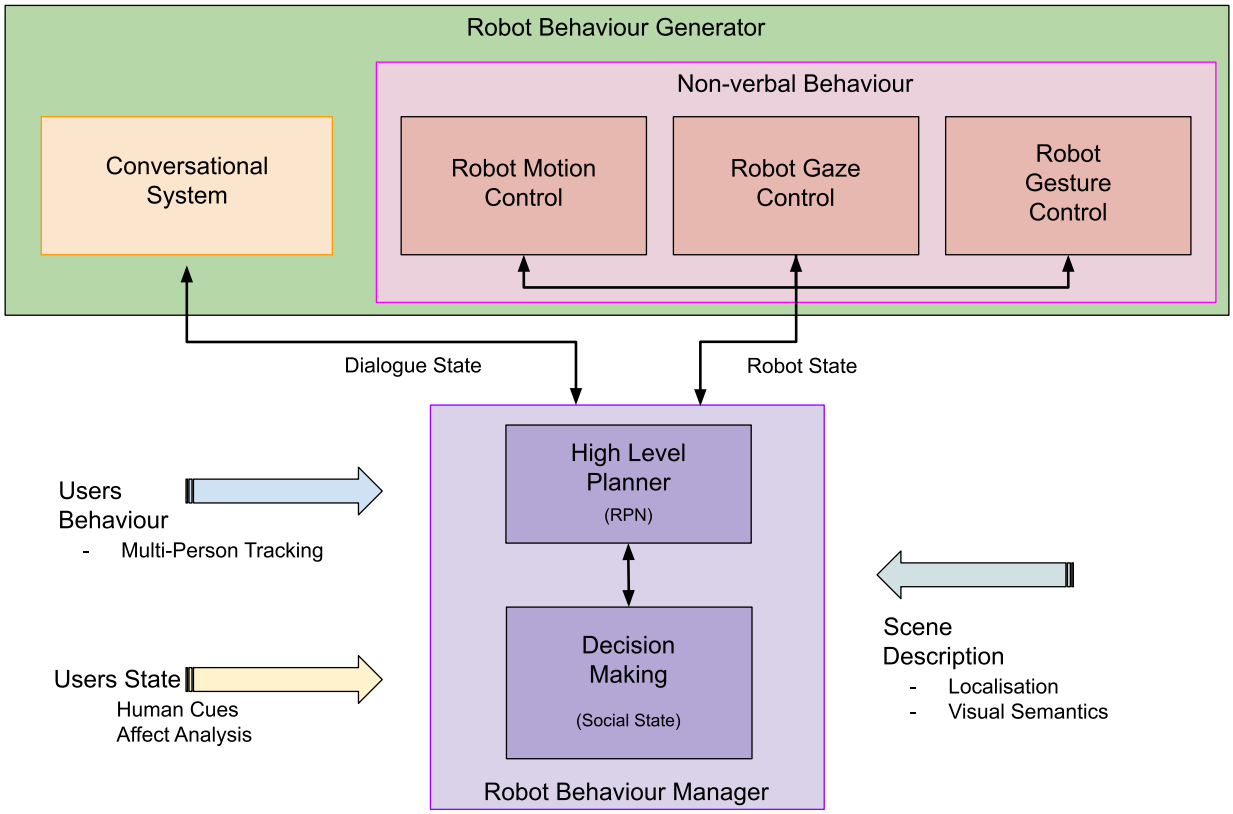

Figure 5.1: Diagram for SPRING Robot High-level Planner System

The high-level planner interfaces the dialogue system and the sensors and physical actions of the robot. We have implemented and tested the high-level planner in order to satisfy the requirements of the specified use-cases and the operation phases 2.2

To make decisions such as which person to interact with, where to go, and what task to execute at what time the high-level task planner will decide which is/are the forthcoming task/s to be performed by the robot. Thus, based on the detected user's intent, it executes a plan combining dialogue, physical, and perception actions.

Normally, robot control and dialogue are separate. Physical actions take time, dialogue actions don't and is hard to combine both in the same planning domain. During conversation, people might switch topics, starting tasks with the robot or start new tasks and forgetting about the old ones, etc.

These requirements demand mechanism that is able to keep track of the current tasks, pause and resume tasks depending on the interaction, reprompt users about open tasks that have been paused and allow for interleaving of tasks, mixing dialogue and physical actions.

Table 5.1: Robot High-level Hierarchical Plans

- Wait (monitor)
- Start Interaction (approach)
- Interact (conversation)
- End Interaction (return to wait)

We have developed a framework for distributed dialogue management, executing plans and running ROS actions concurrently or in sequence; integrating the conversational system, the robot behavior manager, with Petri Net planner and the robot ARI system.

5.1 Petri-Net Machines Planner

The High-Level Task Planner is implemented by a Petri Net Planner (PNP), following [1].

A Petri-Net is a mathematical model for state automata, used to model state machines. It defines places, transitions, arcs, execution tokens. With no dead-ends and that the goal is reachable from every state.

[1] introduces Petri Net Machines, a formal definition for state machines based on Petri Nets that are able to execute concurrent actions reliably, execute and interleave several plans at the same time, and provide an easy to use modelling language. The implementation of the PNP supports automatic generation of Petri-Net Machines, handle concurrent execution of multiple Petri-Net Machines, and naively exploits ROS infrastructure.

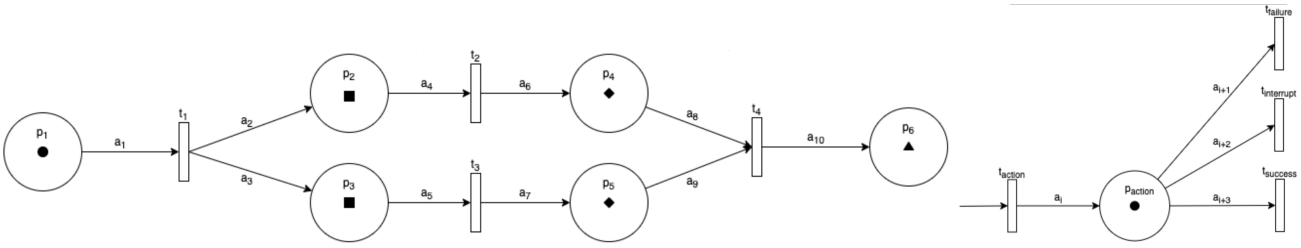

Figure 5.2: Example of Petri-Net with concurrent states. Example of a ROS action with the 3 outcomes of a typical ROS action server (From [1]).

In order to plan interactions, the user can define domains and plans manually. The Yet Another Markup Language (YAML) is use as markup language for defining plans and domains. The PNP uses PDDL like domains for action definition. Actions are defined with a name, a list of parameters, preconditions, and effects. They are then converted to ROS Action Servers, dynamically created from a recipe description. Actions execute based on their name, with no need to create special actions. The Petri-Net planner handles passing of data between states automatically, enabling the plans to to communicate with the conversation and robot non-verbal behaviour systems. It also provide automatic checks and recovery behaviours.

The Petri-Net planner is fully implemented in ROS [13]. It consist of a the Petri-Net Plan server, the knowledge-base (KB), a set of ROS action servers from the petrit-net plans (recipes). The Petri-Net planner start tasks and keeps track of them, and is able to manage the currently available and running Petri-Nets and to provide functionality to send and receive information from/to a specific net. The other major functionality of the controller interface is to exchange data between the different plans and the social state representation.

The High-level Planner is highly interweave with the Interaction Arbiter and the Dialogue Arbiter. Based on the social belief state and localization/map, current dialogue, plans execution state these modules will decide which is/are the forthcoming task/s to be performed by the robot. The ROS Action servers are thus executed according to the plan, and send actions or task commands for the robot behaviour generation modules as needed for execution.

More details about the distinct components that comprised the High-Level Planner, i.e. the planner, the Petri-Net Plan server, the knowledge-base (KB), the set of ROS action servers, where given on deliverable D5.1 [4] and deliverable D6.3 [6].

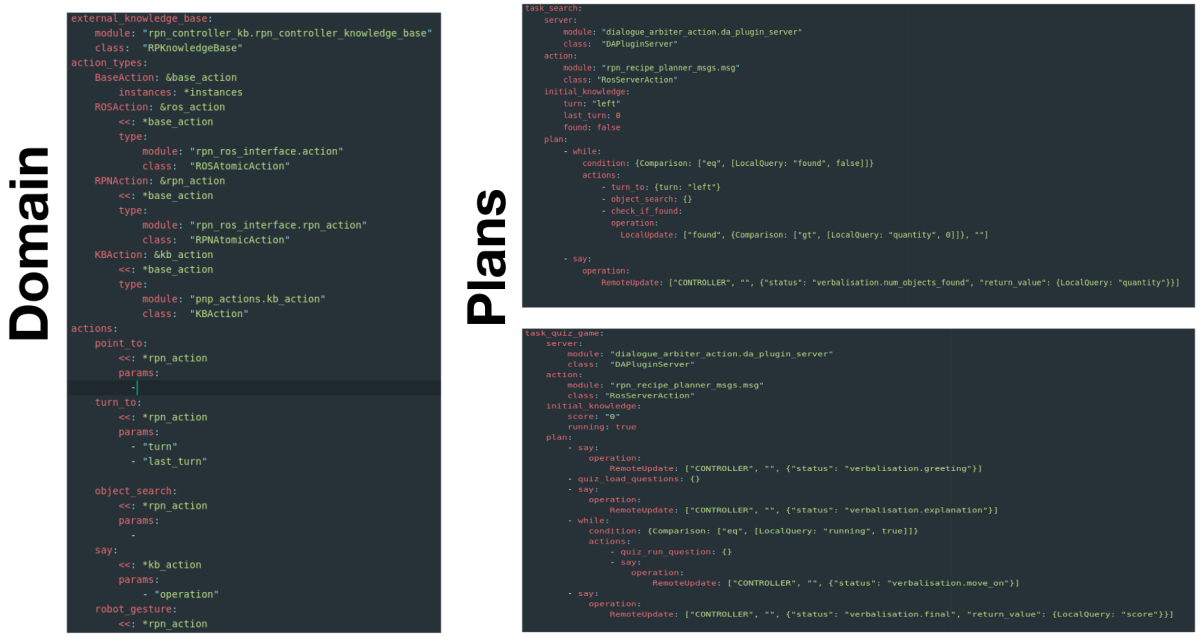

Figure 5.3: Example of PNP domain and plan recipes. PDDL style syntax is used for the domain file of the PNM while the plan file follows its own unique syntax.

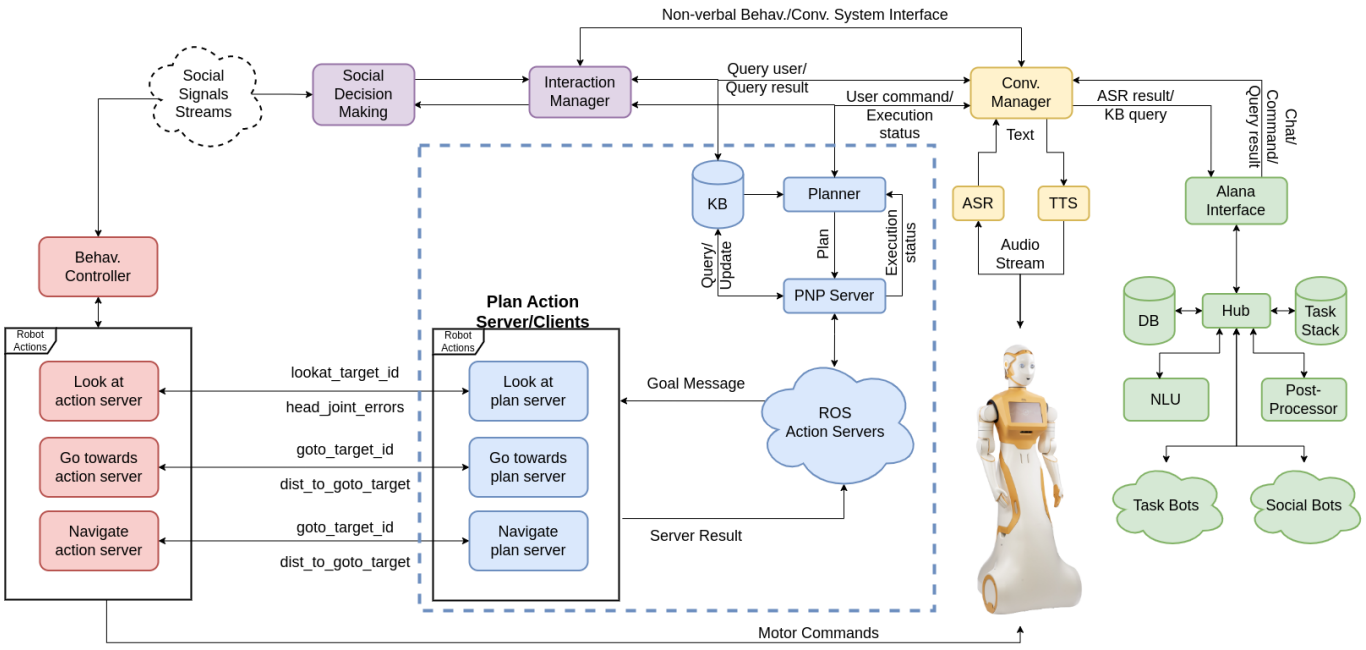

Figure 5.4: SPRING architecture for the Robot Non-verbal Behaviour System and Conversational System (from SPRING Deliverable 6.3 [6]), with the High-Level Planning framework, highlighted in blue. The diagram shows the non-verbal behaviour manager (in purple), the conversation manager interface (in yellow) and the conversational system on the right (in green) with the robot non-verbal behaviour manager components (in red) on the left.

5.2 Interaction Plan Monitor Application

The High-Level Planner, Interaction Manager and Dialogue Arbiter provide a number of "Interaction State" messages to allow monitoring the status and execution of the robot's task, plans and interaction.

The Interaction Plan Monitoring Application provides a web based interface for SPRING researchers to monitor messages from the High-level Planner (High-Level Planner View display on the left of the screen) execution state of the task plans and dialogue state and conversation messages from the Conversational System (Conversation View

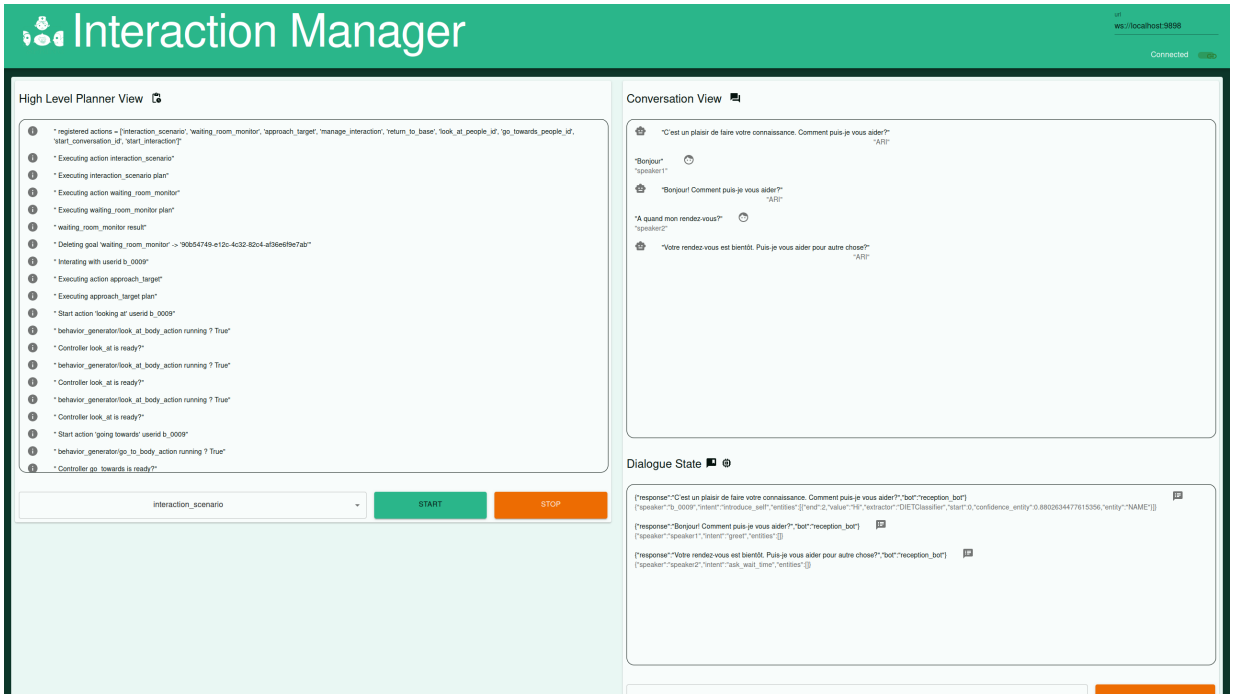

Figure 5.5: Screenshot for the Interaction Plan Monitoring Application. The Planner View display status messages from the high-level planner action. The Conversation View display the state of the conversation dialogue.

display on the right of the screen).

The application also provides researchers with minimal overrides control to select and start/stop a robot's High-level Hierarchical Plans (Table 5.1) or start an Interaction/Conversation with users.

This project has received funding from the European Union's Horizon 2020
Research and Innovation Programme under Grant Agreement No. 871245.

6 Outputs

Various software modules for direct interaction with users, data collection, and the high-level task planner, have been implemented and integrated with the ARI robot system.

The patient's robot messaging screen application can be found in [8]

The experimenter GUI application can be found in [7].

The ROS Petri-Net Planner repository can be found in [10].

The monitor application for the interaction manager can be found in [9].

As per European Commission requirements, the repositories will be available to the public for a duration of at least four years after the end of the SPRING project. People can request access to the software to the project coordinator at `spring-coord@inria.fr`. The software packages use ROS (Robotics Operating System) [13] to communicate with each other and with the modules developed in the other workpackages.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 871245.

Bibliography

[1] Christian Dondrup, Ioannis Papaioannou, and Oliver Lemon. Petri Net Machines for Human-Agent Interaction, 2019.

[2] SPRING Project. D10.3: Privacy and ethics guidelines for experimental validation and data collection. https://spring-h2020.eu/wp-content/uploads/2020/05/SPRING_D10.3_Privacy-and-Ethics-Guidelines_VFinal_30.04.2020.pdf.

[3] SPRING Project. D1.4: User feedback from the preliminary validation (realistic environments). https://spring-h2020.eu/wp-content/uploads/2022/02/SPRING_D1.4_Preliminary-Experimental-Validation_VFinal_31-01-2022.pdf.

[4] SPRING Project. D5.1: Initial high-level task planner and conversational system prototype for realistic environments. https://spring-h2020.eu/wp-content/uploads/2021/06/SPRING_D5.1_Initial_High-level_Task_Planner_and_Conversational_System_Prototype_for_Realistic_Environments_vFinal_31.05.2021.pdf.

[5] SPRING Project. D5.2: Multi-party asr and conversational system in realistic environments. https://spring-h2020.eu/results/.

[6] SPRING Project. D6.3: Robot non-verbal behaviour system in realistic environments. https://spring-h2020.eu/wp-content/uploads/2022/02/SPRING_D6.3_Robot_non-verbal_behaviour_system_in_realistic_environments_VFinal_25.01.2022.pdf.

[7] SPRING Project. Spring-wp1-repository-experimenter-gui. https://gitlab.inria.fr/spring/wp1_user_application/exp-gui.

[8] SPRING Project. Spring-wp1-repository-user-application. https://gitlab.inria.fr/spring/wp1_user_application/user_application/-/tags/sms.

[9] SPRING Project. Spring-wp5-repository-interaction-manager-application. https://gitlab.inria.fr/spring/wp5_spoken_conversations/manager_app/-/tree/v1.

[10] SPRING Project. Spring-wp5-repository-petri-net-planner. https://gitlab.inria.fr/spring/wp5_spoken_conversations/ros_petri_net_planner.

[11] SPRING Project. Wp1: User application. https://gitlab.inria.fr/spring/wp1_user_application.

[12] SPRING Project. Wp5: Spoken conversations. https://gitlab.inria.fr/spring/wp5_spoken_conversations.

[13] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. https://www.ros.org.